

# NFIELD REFERENCE



24 September 2025

Copyright © 2025 NIPO

All rights reserved.

This document contains proprietary information of NIPO. This product is protected by copyright law. Reverse engineering of the software is prohibited.

Due to continued product development this information may change without notice. The information and intellectual property contained herein is confidential between NIPO and the customer and remains the exclusive property of NIPO.

If you find any problems in the documentation, please report them to us in email. NIPO does not warrant that this document is error-free. In cases where the documentation significantly differs from the software implementation, the end user is encouraged to contact NIPO. However, the information in this document can not be used to grant the end user of the product any rights with regard to updates or fixes, demanding a match with the existing documentation.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise without the prior written permission of NIPO. You are not allowed to share the software with individuals outside your company.



# Table of Contents

<b>1.</b>	<b>Introduction.....</b>	<b>9</b>
1.1	Terminology .....	9
1.1.1	General .....	9
1.1.2	Questions and Answers.....	9
1.1.3	Files and Tables.....	10
<b>2.</b>	<b>Using the NIPO ODIN Developer .....</b>	<b>11</b>
2.1	The Script Editor Window .....	11
2.1.1	Line Size.....	11
2.1.2	Pop-up Menu Options.....	11
2.1.2.1	Inserting Question Definitions.....	11
2.1.2.2	Changing Question Options .....	12
2.2	Syntax Checking .....	13
2.2.1	Warning Message for *CODES Question Without Code Categories.....	15
2.2.2	Warning Message for Unfixed Questions .....	15
2.2.3	Error on *FORM Question Language Section Mismatch.....	15
2.2.4	Undeclared Variables are Created when Fixing a Questionnaire .....	15
2.2.5	Results Window Keeps Previous Syntax Checks .....	16
2.3	Test Run Questionnaire .....	16
2.3.1	Question Highlight in NIPO CATI Preview.....	16
2.3.2	Check Routing for Stratification .....	17
2.3.3	Show Variables .....	17
2.4	Menu Options .....	17
2.4.1	Open Data Files .....	17
2.4.2	Remove Commands .....	17
2.4.3	Comment / Uncomment .....	17
2.4.4	Create Questionnaires in Unicode .....	18
2.4.5	Find Function with Regular Expressions and Mark All.....	18
2.4.6	Start Editor in Workbook Mode .....	18
2.4.7	Create Code Numbers Before Code Labels and Fields .....	18
2.5	The NIPO ODIN Developer .....	20
2.5.1	NIPO Diana/DSC Export.....	21
2.5.1.1	Code Label Exports in a *FORM Question .....	23
2.5.1.2	Exports of *FORM Questions with Codes.....	23
2.5.2	Triple-S XML .....	24
2.5.3	SPSS Portable .....	26
2.5.4	SPSS-PC.....	28
2.5.5	Ascribe .....	31

2.5.6	Quantime.....	32
2.6	Imports .....	33
2.6.1	Ascribe .....	34
2.7	NIPO ODIN Developer Configuration .....	34
2.7.1	Syntax Highlighting.....	35
2.7.2	Application Configuration Options .....	35
2.7.2.1	ODIN Options.....	35
2.7.2.2	Interview System Options.....	36
2.7.2.3	CATI.....	37
2.7.2.4	How to setup the Nfield interview system.....	37
2.7.2.5	Coding System Options.....	38
2.7.2.6	Check Options.....	38
2.7.2.7	Dummy Data Options .....	38
2.7.2.8	Print Options.....	39
2.7.2.9	Autosave Options.....	40
2.7.2.10	Files Options.....	40
<b>3.</b>	<b>Using the NIPO ODIN Script Language .....</b>	<b>43</b>
3.1	Naming Conventions .....	43
3.1.1	Questions .....	43
3.1.2	Data Fields.....	44
3.2	Variables.....	44
3.3	System Variables.....	45
3.3.1	Test Mode Aware Scripting.....	45
3.3.2	Repeat Number .....	45
3.3.3	Timer .....	46
3.3.4	Elapsed Time Function.....	47
3.3.5	Language .....	48
3.4	Expressions.....	48
3.4.1	Expression Operators.....	49
3.4.2	Examples of Expressions .....	51
3.4.3	Common Mistakes in Expressions.....	52
3.5	String manipulation routines.....	53
3.5.1	STRFINDMATCH .....	54
3.5.2	STRHASMATCH .....	55
3.5.3	STRINDEX.....	55
3.5.4	STRLENGTH.....	56
3.5.5	STRLOWER .....	56
3.5.6	STRREPLACE .....	57
3.5.7	STRSUBSTR.....	58
3.5.8	STRTRIM .....	58
3.5.9	STRUPPER.....	59

3.6	Form Field References.....	59
3.7	Custom Properties .....	61
3.7.1	Overview .....	61
3.7.2	Custom Properties in Nfield.....	61
3.7.3	property() Function.....	61
3.7.4	In *USELIST command.....	63
3.8	Date Functions.....	65
3.8.1	Overview .....	65
3.8.2	Adding Seconds to a Date.....	65
3.8.3	Calculating the Time Difference Between Two Dates .....	65
3.8.4	Obtaining the Day Number of the Week .....	66
3.8.5	Obtaining the Week Number.....	66
<b>4.</b>	<b>Quota .....</b>	<b>67</b>
4.1	NIPO Academy Sessions on Quota .....	67
4.2	What is Quota? .....	67
4.2.1	Steps to Create Quota Frame.....	68
4.2.2	How to Define Quota Variables and Levels .....	68
4.2.3	How to Organize and Order the Quota Variables .....	71
4.2.4	How to Add Quota Targets.....	73
4.2.5	Max Overshoot Option .....	74
4.2.6	Changing Quota Frames.....	75
4.2.7	Uploading and Downloading Quota Frames.....	77
4.2.8	Quota in ODIN Script .....	77
4.2.9	Quota Out in Paradata .....	77
4.3	Minimum and Maximum Targets .....	77
4.3.1	Only a Total Target Entered .....	78
4.3.2	Total Target and Max Targets for Levels are Entered .....	78
4.3.3	If the Total Target is Not the Same as the Sum of Maxes .....	78
4.3.4	Total Target and Minimum Level Targets are Entered.....	79
4.3.5	Total Target Smaller than the Sum of Level Minimums .....	80
4.3.6	Total Target Greater than the Sum of Level Minimums.....	80
4.3.7	Combining Maximum and Minimum Level Targets .....	81
4.3.8	Overshoot.....	82
4.4	Least-filled Quota .....	83
4.5	Multi Quotas.....	90
4.5.1	Linking Multi Quota with Other Quotas.....	93
4.6	Quota frame validation.....	94
4.7	Counting Quota Level More than Once per Interview.....	95
<b>5.</b>	<b>Suspend and Resume Interview.....</b>	<b>97</b>
5.1	Ways to Suspend an Interview.....	97
5.2	Resume Interview .....	98

<b>6.</b>	<b>Command Index .....</b>	<b>99</b>
6.1	*? .....	99
6.2	*ALPHA .....	104
6.3	*BACK .....	106
6.4	*BLOCK .....	108
6.5	*BUT .....	111
6.6	*CODES .....	114
6.7	*CONTROL .....	116
6.8	*COPY .....	118
6.9	*COUNT .....	120
6.10	*DATE .....	122
6.11	*DUMMY .....	125
6.12	*END .....	126
6.13	*ENDNGB .....	128
6.14	*ENDPAGE .....	129
6.15	*ENDST .....	131
6.16	*EXCLUDE .....	132
6.17	*FIELD .....	134
6.18	*FONT (definition) .....	136
6.19	*FONT (switching) .....	138
6.20	*FORM .....	141
6.21	*GETLFQLIST .....	145
6.22	*GOSUB .....	151
6.23	*GOTO .....	153
6.24	*GROUP .....	155
6.25	*HEADING .....	157
6.26	*ID .....	160
6.27	*IF (condition), *ELSEIF, *ELSE, *ENDIF .....	162
6.28	*IF (question option) .....	163
6.29	*INCLUDE .....	165
6.31	*INIT .....	167
6.32	*LABEL .....	168
6.33	*LANGUAGE .....	169
6.34	*LIST (definition) and *ENDLIST .....	176
6.35	*LIST (question option) .....	179
6.36	*MATRIX .....	182
6.37	*MAX .....	188
6.38	*MERGE .....	190
6.39	*MIN .....	193
6.40	*MULTI .....	195
6.41	*NMUL .....	197

6.42	*NOCON.....	198
6.43	*NON.....	200
6.44	*NUMBER .....	201
6.45	*OPEN (question type) .....	203
6.46	*OPEN (codes option).....	205
6.47	*ORDER.....	206
6.48	*PAGE.....	209
6.49	*PICT (question option).....	210
6.50	*PICT (codes option).....	212
6.51	*PROPERTIES.....	214
6.52	*PUT.....	217
6.53	*QUESTION .....	220
6.55	*QUOTA .....	222
6.56	*RANDOM.....	224
6.57	*RANGE.....	226
6.58	*REC.....	228
6.59	*REPEAT ... *ENDREP.....	230
6.60	*REQUEST.....	233
6.61	*RETURN .....	243
6.62	*ROT.....	244
6.63	*SAMPLEDATA.....	246
6.64	*SAVE (question option).....	247
6.65	*SAVE (codes option).....	249
6.66	*SHOWDOCUMENT.....	250
6.67	*SORT and *STOPSORT .....	252
6.68	*SPLITSTRING.....	255
6.69	*STOPRANDOM.....	258
6.70	*STRAT.....	259
6.71	*SUBROUTINE ... *ENDSUB.....	260
6.72	*SWILANG.....	262
6.73	*TABLE.....	264
6.74	*TEMPLATE.....	266
6.75	*TEXTVARS .....	268
6.76	*UIOPTIONS (command).....	270
6.77	*UIOPTIONS (question option).....	272
6.78	*UIRENDER (command) .....	274
6.79	*UIRENDER (question option) .....	276
6.80	*USEBUTTONS .....	277
6.81	*USELIST .....	282
6.82	*VAR.....	284
6.83	*VARS.....	286

<b>7.</b>	<b>File Structures and Database Tables .....</b>	<b>289</b>
7.1	Data Files.....	289
7.1.1	Closed Answers File (DAT-file) .....	289
7.1.2	Open answers file (O-file).....	289
7.2	Paradata .....	291
7.2.1	Introduction to Paradata.....	291
7.2.1.1	What is Paradata .....	291
7.2.1.2	What is Paradata Used For .....	291
7.2.1.3	How to Create and Download Paradata.....	291
7.2.1.4	How to Set Paradata to Auto Sync after Each Interview .....	293
7.2.2	Paradata in CAPI Surveys.....	293
7.2.2.1	CAPI survey without Sampling Points.....	294
7.2.2.2	CAPI survey with Sampling Points and Quota.....	299
7.2.2.3	CAPI survey with Sampling Points and Addresses.....	301
7.2.2.4	CAPI survey with Sampling Points, Quota and Addresses.....	304
7.2.3	Paradata in Online Surveys.....	305
7.2.4	Paradata for the Quota Out (both Online and CAPI Surveys).....	306
7.3	Audit Trail.....	307
7.3.1	Introduction to Audit Trail.....	307
7.3.2	Overview of an Audit Trail file .....	308
7.3.3	Suspending an Interview .....	310
7.3.4	Killing the Nfield CAPI app During Interview .....	312
7.4	Sample Table .....	312
7.4.1	Introduction to Sample Table.....	312
7.4.2	Rules for Sample Table Headers.....	315
7.4.3	Dealing with Customer Personally Identifiable Information (PII) data .....	316
7.4.4	Blacklist .....	317
7.4.5	Locations of Nfield Services.....	317
7.4.6	Local Data Storage Possibilities .....	318
<b>8.</b>	<b>Interview Simulator.....</b>	<b>320</b>
8.1	Running an interview simulation .....	320
8.2	Hints.....	322
8.2.1	What are hints for interview simulation? .....	322
8.2.2	When to use Hints?.....	322
8.2.3	Example .....	322
8.2.4	Properties.....	324
	For more information.....	325
	For more information on this feature, please watch our NIPO Academy session 46.....	325
8.2.5	Excluding specific response codes .....	326
8.2.6	Total sum value for the question .....	327
<b>9.</b>	<b>Default Template for Nfield System Rendering Options .....</b>	<b>329</b>

9.1	Capture Photo.....	329
9.2	Capture Audio.....	331
9.3	Play Media.....	333
<b>10.</b>	<b>Only Relevant for Online Surveys.....</b>	<b>335</b>
10.1	Exit Links.....	335
10.1.1	Placing the Exit Links.....	335
10.1.2	Types of Exit Links .....	336
10.1.3	Generic Exit Links Example.....	336
10.1.4	Variables in Exit Links Examples .....	337
10.2	NIPO Status Page .....	339
<b>11.</b>	<b>Only Relevant for CAPI Surveys.....</b>	<b>341</b>
11.1	Silent Recording.....	341
11.2	GPS Location Fix from Script.....	347
11.3	Adding custom fields to Sampling Points.....	349
<b>12.</b>	<b>Appendix .....</b>	<b>351</b>
12.1	Nfield Acceptable Use Policy .....	351
12.2	Maximum Number of Respondents To Upload .....	351
12.3	Maximum Length for Sample Fields.....	351
12.4	Response Codes.....	351
12.5	TTStartLink .....	353



# 1. Introduction

This is the command reference for the NIPO ODIN scripting language.

Please make sure to always use the latest ODIN Developer version. It can be downloaded from our [support](#) website. Ask the NIPO support team to allow you to download the latest version of ODIN (currently, the latest version is 5.18.021).

In this reference we are only showing the features of ODIN Developer relevant to Nfield, even though ODIN Developer also supports commands for NFS.

Please also note that all examples in this reference are based on the default NIPO template (Nfield Chicago). If you use a different template, the \*UIOPTIONS and the \*UIRENDER might be different for your template than in the examples shown.

## 1.1 Terminology

This section provides a brief overview of the terminology used within this document.

### 1.1.1 General

#### Question type

Defines the type of answer which is expected for a question. Question types are for example closed, open, numerical, etc.

#### Question option

Defines what special properties an answer must have or how answer categories will be displayed. There are question options to allow multiple answers, set a maximum value to be entered, show answer categories in random order, etc.

#### Answer option

Defines the behavior of the program when an answer category is chosen. There are answer options to prevent that an answer is combined with other answer categories, to prompt for an open answer, etc.

### 1.1.2 Questions and Answers

#### Closed question

A question where the answer is expected to be a choice from a fixed number of answer categories.

#### Answer code category

One of the possible answers defined for a (semi-)closed question.

**Open question**

A question where the answer is expected to be entered literally as text.

**Semi-closed question**

A question where the answer is expected to be a choice from a fixed number of answer categories and where certain answer categories have a box to enter an alternative answers as text.

**Open-ended answer**

The literal text of an answer entered for an open or semi-closed or open question.

**Numerical question**

A question where an answer is expected to be a numerical value.

**Text question**

A question where an answer is expected to be text of limited length.

**Answer code**

A numeric value which is stored in the DAT-file when an answer category is chosen.

**Answer field**

A (series of) positions in the DAT-file where answer codes, values or texts are stored.

### **1.1.3 Files and Tables**

**Q-file (questionnaire)**

Unicode file containing the question text and all the NIPO ODIN commands for routing, et cetera.

**Sample table**

Table in the database containing the gross sample. Each record in the sample table contains information (telephone number, address, name, company size, etc.) about one (future) respondent.

**DAT-file**

Unicode text file containing answer codes that refer to chosen answer categories of closed questions, values entered for numerical questions and texts entered for text questions.

**O-file**

Unicode text file containing all open answers entered by keyboard.

## 2. Using the NIPO ODIN Developer

The NIPO ODIN Developer is the NIPO ODIN script author's tool. It can be used for the following purposes:

- Create, edit and syntax-check questionnaires for Nfield.
- Run a preview of the questionnaire.
- Generate dummy (test) data to verify questionnaire integrity.
- Export survey data for statistical analysis in various packages.

This section briefly discusses a number of features of the NIPO ODIN Developer.

### 2.1 The Script Editor Window

#### 2.1.1 Line Size

A line in the NIPO ODIN Developer may be up to 4000 characters long. Scrolling in a very long line might be slow - use **Edit > Goto position** to easily place the cursor on a certain position. To open data files, select **File > Open data file...** from the menu.

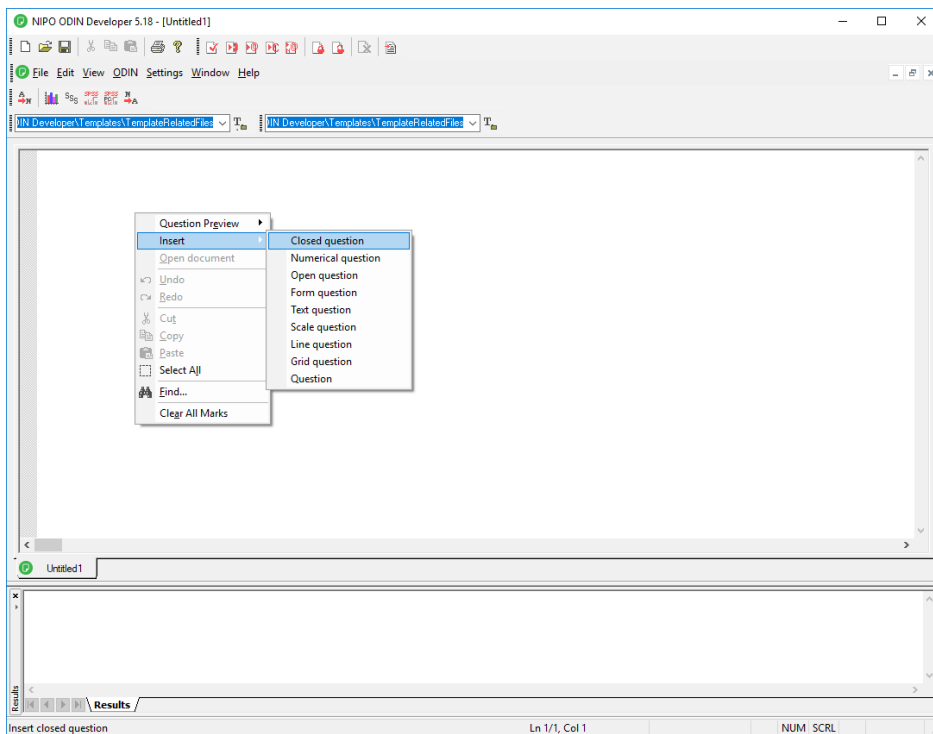
#### 2.1.2 Pop-up Menu Options

For any open file in the script editor window, right-click anywhere to open the pop-up menu. This section describes a number of those options.

##### 2.1.2.1 Inserting Question Definitions

To simplify creating NIPO ODIN Questionnaire, right-click the editor window and select Insert to choose from the available question options.

## Inserting a question



### 2.1.2.2 Changing Question Options

Right-click on a question line to change the question options. Depending on the type of question, one of the following dialog boxes appears.

#### Changing the general question options

The 'Question Properties' dialog box is shown with the 'General' tab selected. The 'Closed' sub-tab is also visible. The 'Display' section contains checkboxes for 'Allow no answer', 'Don't clear screen', and 'Display picture'. Below these are input fields for 'Name', 'Type' (set to 'String'), 'Library name', and a 'Filter' checkbox. The 'Storage' section has input fields for 'Position' and 'Field' (set to '1'), and checkboxes for 'Use sample file' and 'Save answer in variable'. The 'Diana variables' section has checkboxes for 'Attach label' and 'Variable'. The dialog has 'OK', 'Cancel', and 'Apply' buttons at the bottom.

### Changing the question options for a \*CODES question

Question Properties

General Closed

☐ Multiple answers

☐ Auto format

☐ Control by question  ☐ Include ☐ Mark

☐ Code list  Type: String

☐ Min

☐ Max

Answer order

☒ Normal

☐ Random

☐ Inverted

☐ Rotated

OK Cancel Apply

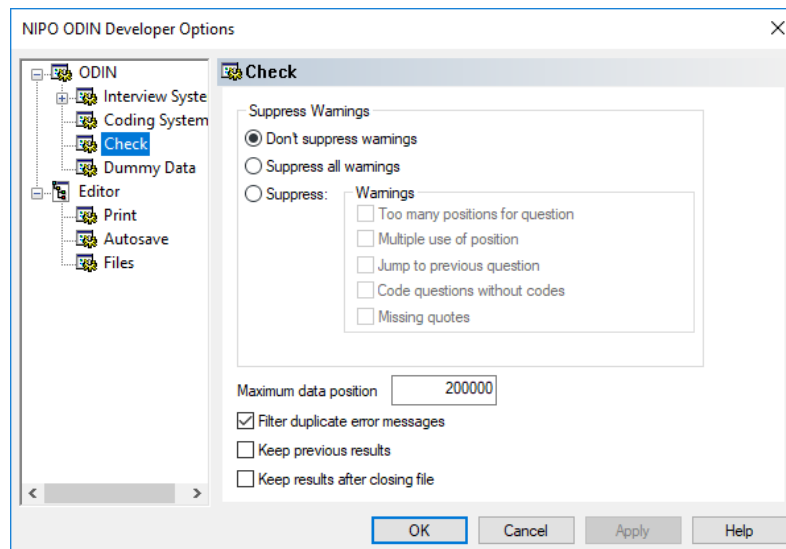
The options you see depend on the question types.

## 2.2 Syntax Checking

You can syntax check your questionnaire by selecting **ODIN > Check...** from the menu. A dialog appears in which warning options may be configured. Warnings are issues that may lead to unwanted behavior and / or loss of data, but do not prevent the questionnaire from running.

**Note:** By clicking on individual error or warning messages, the line in the code to which that error or warning pertains becomes highlighted and comes into focus.

### NIPO ODIN Developer check options



You may select the following options:

- **Don't suppress warnings.** All warnings are reported.
- **Suppress all warnings.** None of the warnings are reported.
- **Suppress.** You can select which warnings to suppress.

The following warnings may optionally be suppressed:

- **Too many positions for question.** Applies if more positions are defined than strictly required to store the input data. The length definition of a `*CODES` question reserves more space than required to store the answer codes. This may sometimes be the case for any question where `*MULTI` was accidentally omitted.
- **MAX positions for Diana.** Applies if more than 200,000 positions are used.
- **Multiple use of position.** Applies if a particular position is accessed to store information from more than one question. This may cause the data to be overwritten. This warning is ignored for positions in a `*DUMMY` question.
- **Jump to previous question.** Applies if a `*GOTO` is used to jump back instead of forward in the questionnaire, which may cause an infinite loop. It is recommended to use `*BACK` instead.
- **Code question without codes.** Code questions without codes are skipped, therefore a warning can be issued if a code question misses codes. This warning is ignored for `*CODES` questions in combination with `*DUMMY`.
- **Missing quotes.** Applies to quoted text where a closing quote is missing.

#### Maximum data positions

Sets the maximum amount of positions allowed for a single interview. This is useful to match NIPO ODIN to the limit of your data processing system. Nfield limit is 200,000 positions.

**Filter duplicate error messages**

This limits the number of reports per error message. For example, duplicate use of a position in the DAT-file may only need to be reported once.

**Keep previous results**

Results of a previous syntax check are not cleared for a new syntax check; instead, a new window is opened so that results may be compared.

**Keep results after closing file**

Keeps the **Results** window open once the questionnaire file is closed in the editor.

**2.2.1 Warning Message for \*CODES Question Without Code Categories**

If no code categories are defined for a \*CODES question a warning is displayed upon a syntax check. No warning is supplied if a question text is also missing.

**2.2.2 Warning Message for Unfixed Questions**

You can test-run a questionnaire within the NIPO ODIN Developer NIPO ODIN Developer with unfixed questions, but can not run the questionnaire in Nfield with unfixed questions.

The NIPO ODIN Developer gives a warning message upon syntax check if any unfixed questions (length definitions without position definition) have been defined in the NIPO ODIN Questionnaire.

Only one warning is ever generated for the first line where an unfixed question was found.

Select **ODIN > Fix** or click the **Fix** button to fix the questions.

**2.2.3 Error on \*FORM Question Language Section Mismatch**

The syntax check reports an error message when a \*FORM question in the language section does not contain any fields (\*NUMBER or \*ALPHA). The syntax check also reports an error message when a field (\*NUMBER or \*ALPHA) is specified in a question that is not defined as \*FORM.


**2.2.4 Undeclared Variables are Created when Fixing a Questionnaire**

If a variable is not declared before it is being used, a warning message appears. When fixing the data positions or renumbering a questionnaire, these variable name declarations are automatically added to the questionnaire as \*TEXTVARS variables. Make sure no mistakes are made in the variable name, and verify if \*TEXTVARS is an appropriate type.

## 2.2.5 Results Window Keeps Previous Syntax Checks

The **Results** window of a syntax check keeps previous syntax checks. Right-click the window and select **Remove tab** to close the result window. Click on the cross on the upper-left corner or use the **View > Results** from the menu to toggle showing the docking window.

## 2.3 Test Run Questionnaire

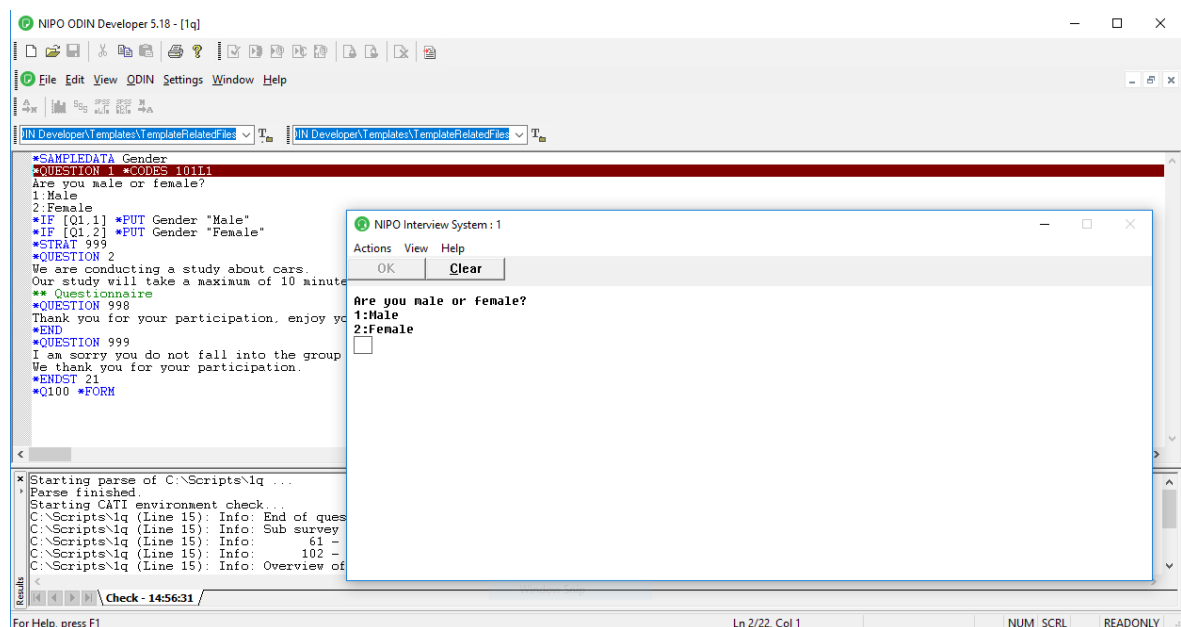
Save your questionnaire and select **ODIN > Run** from the menu and select to preview the questionnaire in NIPO **CATI** (or just click on the **Run** icon , which automatically starts a preview in CATI). You could also do a preview in Nfield. For that, please refer to our [NIPO ODIN Scripter Basic](#) course.

### 2.3.1 Question Highlight in NIPO CATI Preview

For a NIPO CATI preview, the NIPO ODIN Developer highlights the question currently in focus in the script window. This helps in reviewing the question in focus during a preview run.

#### Note:

Special characters in the file path (underscores, ampersands et cetera) are not supported.



#### Question highlight

The script cannot be edited during the preview. When exiting the preview run, the cursor remains at the last highlighted question.

This feature requires that the NIPO CATI Client used for preview is version 5.11 or newer. If the highlight does not work, verify that the correct NIPO CATI Client version is in use (see "Interview System Options" on page 36). In addition, enable **Adjustable screen size** to see the script window.

---

**Note:**

This feature is only available for a NIPO CATI preview.

---

### 2.3.2 Check Routing for Stratification

When using the command `*STRAT` in a questionnaire, for each occurrence a message dialog 'stratification filled? Yes/No' pops up. Select the appropriate answer to test your routing. Note that no telephone file or sample table is used.

### 2.3.3 Show Variables

For NIPO CATI preview, you can show the contents of the currently known variables by selecting **View > View variables...** from the menu. Note that system variables that appear are also NFS system variables, and not only the Nfield ones. Other variables have to be declared with `*VARS`, `*TEXTVARS` or `*SAMPLEDATA` first.

## 2.4 Menu Options

### 2.4.1 Open Data Files

The regular script editor is capable of managing up to 4,000 horizontal positions. To open larger (data) files, select **File > Open Data File...** from the menu. This opens files up to 99,999 horizontal positions, and cancels syntax highlighting.

### 2.4.2 Remove Commands

To be able to create a client-friendly questionnaire as well as to translate your questionnaire, select **ODIN > Remove Commands...** from the menu. This option creates a translatable `*LANGUAGE` section without NIPO ODIN commands, except for `*QUESTION`, `*FONT` and `*?varname`, that may be linked to the original questionnaire using the `*LANGUAGE` command.

### 2.4.3 Comment / Uncomment

Select a block or a couple of lines and select **Edit > Comment** to make comments of these line. This inserts a `**` in front of all selected lines. Select **Edit > Uncomment** to remove the `**` from the selected lines.

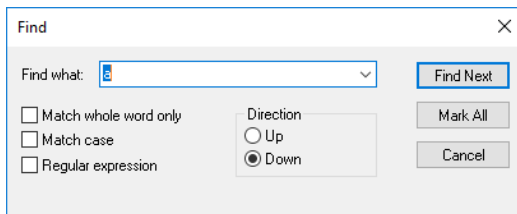
### 2.4.4 Create Questionnaires in Unicode

This means full support for all non-western languages, like Hebrew, Arabic, Chinese, Japanese, et cetera. Unicode is enforced by saving the file as type UTF-16. This is required for all extended characters in questionnaires for Nfield.

### 2.4.5 Find Function with Regular Expressions and Mark All

Select **Edit > Find** to search for specific text in your questionnaire. Click **Mark All** to mark entries found in the side bar.

Find function



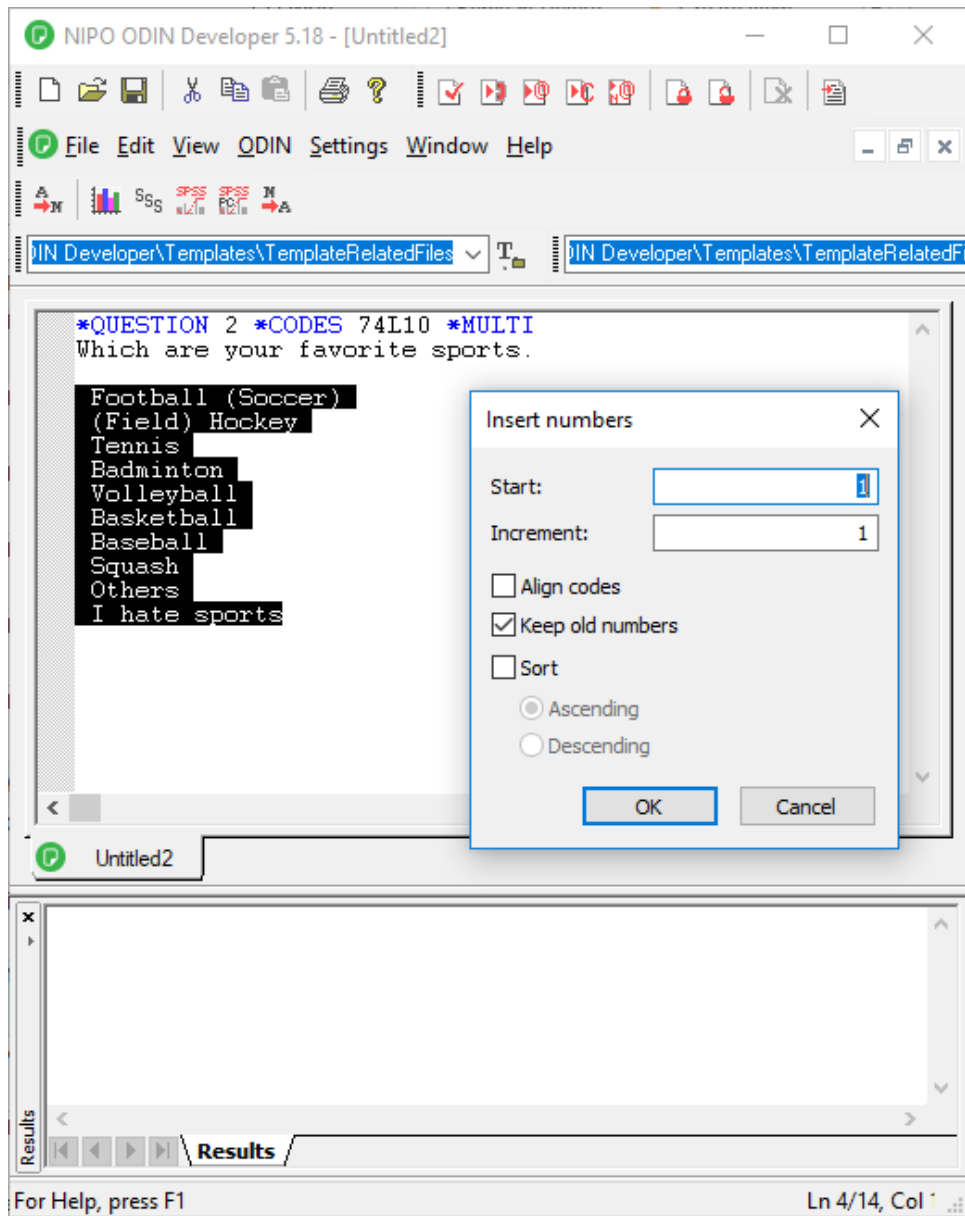
### 2.4.6 Start Editor in Workbook Mode

Select **View > Workbook** to enable or disable showing windows in a workbook mode. The workbook mode makes switching windows easier.

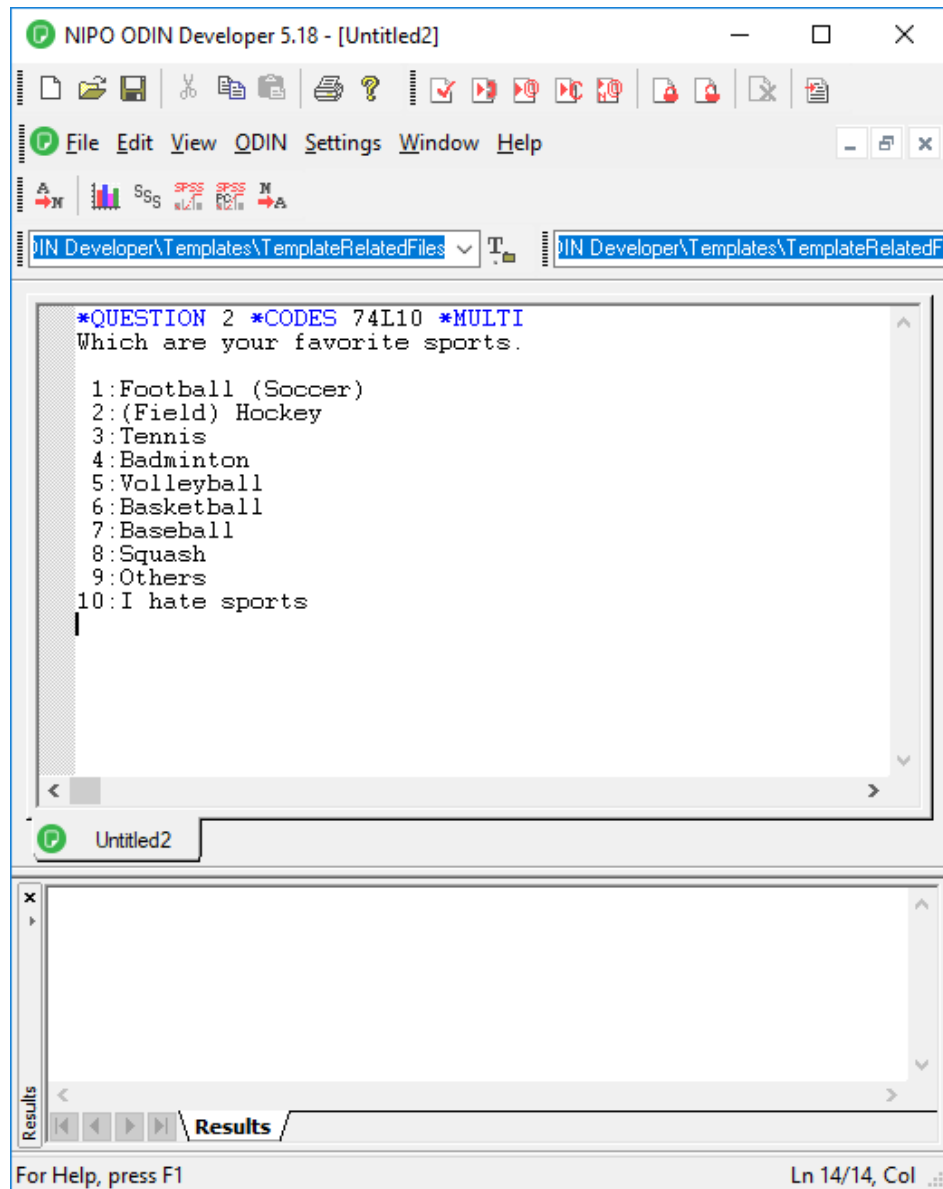
### 2.4.7 Create Code Numbers Before Code Labels and Fields

Select the lines you want to number, then select the **ODIN > Insert numbers** from the menu.

The NIPO ODIN Developer does not immediately verify code numbers against any duplicates or against the question definition. To create unique numbers, do not update partial lists. Use the syntax check to check on duplicate code numbers. Numbering is performed on all lines containing a carriage return.



### Creating code category numbers



Result

## 2.5 The NIPO ODIN Developer

The NIPO ODIN Developer allows you to export your data to a variety of statistical software packages.

The following formats are supported:

- NIPO Diana / Nvision Script
- Triple-S XML
- SPSS Portable

- SPSS-PC (script format)
- Ascribe
- Quantime
- Image / Sound

### 2.5.1 NIPO Diana/DSC Export

The export to NIPO Diana/DSC may be used to create files that may be used by both NIPO Diana and Nvision Script, as well as input for the NIPO DSC. This export is also known as .Var export.

Export to NIPO Diana/DSC variables

**DIANA**

Options

Language: Default language

☐ Add filters to variables

☐ Variable name in front of question text ☐ + [LF]

☐ Use CODE 'n' if no code text specified

☐ 'Question n.' in front of question text ☐ + [LF]

☐ Write random and order numbers

☐ Insert [LF] in question text

☐ Insert [LF] in code text

☐ Include questions without text and unused questions

☐ Generate codes for scale questions

☐ Include null-evaluating filter and dummy questions

☐ No duplicate text for "OPEN" and "NUMBER" questions

☐ No comment lines in var file

☐ Matrix text as DSC table label

Maximum codes per variable:

Line length:

☐ Open VAR file after export

☐ Unicode VAR file

OK Cancel Save

Choose the following options:

#### Language

If more than one language is defined within the questionnaire, select the language to use for the question and code labels in the export.

#### Add filters to variables

Any questionnaire filters are also defined in the variables.

**Variable name in front of question text**

The question label is preceded by the export name of the variable. Optionally add a line feed to the variable name.

**Use CODE 'n' if no code text specified**

For code category labels without text, a label is created containing the word 'CODE' followed by the code number.

**'Question n' in front of question text**

All question texts are preceded by the word "Question" followed by the question number originally used in the questionnaire. Optionally add a line feed to the variable name.

**Write random and order numbers**

This option is not relevant for Nfield.

**Insert [LF] in question text**

If line feeds are used in the question labels, these are transferred in the export. Duplicate line feeds are removed. By default, line feeds are translated into spaces.

**Insert [LF] in code text**

If line feeds are used in the code labels, these are transferred in the export. Duplicate line feeds are removed. By default, line feeds are translated into spaces.

**Include questions without text and unused questions**

Exports questions without text and questions that are never shown due to routing. By default, these are not exported.

**Include null-evaluating filter and dummy questions**

Exports questions that use a filter that always evaluates to false (for example `*IF [ 1 = 0 ]`) and `*DUMMY` questions. By default, these are not exported.

**No duplicate text for \*OPEN and \*NUMBER questions**

For `*OPEN` and `*NUMBER` questions, creates a label with the question text, removing the label for the question text.

**No comment lines in VAR file**

Removes comment lines (`COM`) from the variable file. Comment lines are used as export directives for the NIPO DSC for IBM SPSS, but NIPO Diana cannot correctly handle certain comment directives. Check this option if the export must be used by NIPO Diana. Do not check this option if the export must be used by the NIPO DSC for IBM SPSS.

**Maximum codes per variable**

Enforces a maximum number of codes per variable. Codes beyond this limit are not exported. This may be required for NIPO Diana, which has a limit of 200 variables.

**Set line length**

Sets the maximum amount of characters to be used for question labels and code labels on a single row. Rows are split using the backslash character. Sentences are split at complete words.

**Open VAR file after export**

Automatically opens the exported variable file in the NIPO ODIN Developer.

**Unicode VAR file**

Exports a Unicode variable file instead of an ASCII file (supported by Nvision Script and DSC).

**2.5.1.1 Code Label Exports in a \*FORM Question**

In a \*FORM question, all text of a code label (both before and after the data field) is used for the export to NIPO Diana variables.

**Exporting text of \*FORM question**

```
*QUESTION 1 *FORM
How much did you pay for:

1: Product A      *NUMBER 61L3.2 Euros
2: Product B      *NUMBER 66L3.2 Euros
```

**Is exported as:**

```
*V1_1 61L3.2: How much did you pay for: Product A Euros
*V1_2 66L3.2: How much did you pay for: Product B Euros
```

**2.5.1.2 Exports of \*FORM Questions with Codes**

Exports of \*FORM questions using codes for fields appropriately numbers variables according to the code numbers. The label is created from all successive text for that code.

**Example script**

```
*QUESTION 1 *FORM
What is your address?

1: Street  *ALPHA 61L35in
2: House number  *NUMBER 96L5
3: Postal code  *ALPHA 101L10
4: City  *ALPHA 111L30
```

**Result export to NIPO Diana script**

```
*V1_1 *TEK 61L35: What is your address? Street
*V1_2 *SNG 96L5: What is your address? House number
*V1_3 *TEK 101L10: What is your address? Postal code
*V1_4 *TEK 111L30: What is your address? City
```

## 2.5.2 Triple-S XML

Triple-S XML is an open standard data format using XML. It is supported by a variety of products, including but not limited to:

- Bellview
- CfMC Survent
- Merlin
- Miriad (TNS)
- Preport
- SNAP

An export to Triple-S requires that the survey data file is present in same the directory as the questionnaire that is exported. For a questionnaire `NAMEQ`, the data file should be named `NAME.DAT`.

### Export to Triple-S XML

### Export to Triple-S version

Sets the Triple-S format to use. Check your statistical analysis package for details. Supported exports are 1.1, 1.2 and 2.0.

### Data file format

Sets the data file format for the export, either **fixed** or **csv** (character-delimited). This is only supported for Triple-S version 2.0.

**Convert text using code page**

Select the code page to convert the export into. Make sure the selected code page matches the language of your questionnaire.

**Subsurvey**

Select which survey to export if *\*NEW* was used within the questionnaire.

**Language**

If more than one language is defined within the questionnaire, select the language to use for the question and code labels in the export.

**Include alphanumeric questions**

Include *\*ALPHA* and *\*NUMBER* questions in the export.

**Include questions without text and unused questions**

Exports questions without text and questions that are never shown due to routing. By default, these are not exported.

**Include null-evaluating filter and dummy questions**

Exports questions that use a filter that always evaluates to false (for example *\*IF [ 1 = 0 ]*) and *\*DUMMY* questions. By default, these are not exported.

**Recode questions containing code '0'**

Code 0 for code categories is by default not supported in the Triple-S XML format. This option recodes code labels if code 0 was used in the questionnaire.

**Include open answers**

In addition to optionally coded open-ended questions, this exports open-ended answer verbatim into the data file.

**Open answer length**

Sets the maximum number of characters to be used for open-ended answers. Answers beyond the length limit are truncated.

**Check QPS compliance**

Checks compliance with Triple-S exports for QPS. If the compliance is not met, a warning message is displayed.

**Open SSS file**

Automatically opens the Triple-S variable definition file in the NIPO ODIN Developer after export.

The following characters in the question and code text are escaped when exporting to Triple S XML:

- < is replaced by `&lt;`;
- > is replaced by `&gt;`;

- & is replaced by &amp;
- ' is replaced by &apos;
- " is replaced by &quot;

## 2.5.3 SPSS Portable

An export to SPSS Portable requires that the survey data file is present in same the directory as the questionnaire that is exported. For a questionnaire `NAMEQ`, the data file should be named `NAME.DAT`.

### Export to SPSS POR file

#### Subsurvey

Select which survey to export if `*NEW` was used within the questionnaire.

#### Language

If more than one language is defined within the questionnaire, select the language to use for the question and code labels in the export.

#### Convert text using code page

Select the code page to convert the export into. Make sure the selected code page matches the language of your questionnaire.

#### Retain ODIN data format

Not available for this format.

#### Include open answers

Not available for this format.

**Max record length**

The maximum number of positions of a single record. Check your statistical analysis software for the limits. Longer records are split at the configured threshold.

**Include questions without text and unused questions**

Exports questions without text and questions that are never shown due to routing. By default, these are not exported.

**Include null-evaluating filter and dummy questions**

Exports questions that use a filter that always evaluates to false (for example `*IF [ 1 = 0 ]`) and `*DUMMY` questions. By default, these are not exported.

**Include save script**

Not available for this format.

**Variable name in front of question text**

The question label is preceded by the export name of the variable. Optionally add a line feed to the variable name.

**'Question n' in front of question text**

All question texts are preceded by the word "Question" followed by the question number originally used in the questionnaire.

**Use question-ID's as variable names**

If specified, use names defined with `*VAR` as variable names.

**Use alternative names for random and order variables**

Uses a slightly shorter format for `*RANDOM` and `*ORDER` variables, where the default first letter is replaced by `R` or `O` respectively.

**Write random and order numbers**

Where `*RANDOM` and `*ORDER` are used with a position definition, this stores the order in which the codes appeared during the questionnaire in an additional variable.

**Use code numbers for Multiple Dichotomy fields**

By default, for `*MULTI` questions all codes are exported as multiple dichotomy questions (mentioned / not mentioned). This option places the original code number in the question label for these questions.

**Text for mentioned**

For multiple dichotomy questions, sets the label to be used for 'mentioned'.

**Text for not mentioned**

For multiple dichotomy questions, sets the label to be used for 'no mentioned'.

**Variable name length**

Sets the maximum length for variable names. Choose either **8** or **64**, dependent on your SPSS version in use.

**Variable label length**

Sets the maximum length for the variable label (the question text). Longer texts are truncated at the threshold.

**Value label length**

Sets the maximum length for the value label (the code label text). Longer texts are truncated at the threshold.

**Variable names first letter**

Sets the first letter for an exported variable name. This only happens question IDs are not used, or if the question does not have a question ID defined (\*VAR).

**Number of chars from multi question**

By default, multiple dichotomy questions receive the full original question text in addition to the code label. This option configures the maximum number of characters from the question label to include.

**Note:**

The selected settings for variable names may cause the export to generate variable names that exceed the maximum number of characters for the SPSS version you are using. A warning is issued for the minimum limit for older versions - adjust the settings if required.

**2.5.4 SPSS-PC**

An export to SPSS-PC script files requires that the survey data file is present in same the directory as the questionnaire that is exported. For a questionnaire `NAMEQ`, the data file should be named `NAME.DAT`. This export creates two files: a variable definition file (`SPS` file) and a data file (`EXT` file). Run the `SPS` file in SPSS to create the data set.

**Export to SPSS-PC (script files)**
**Subsurvey**

Select which survey to export if \*NEW was used within the questionnaire.

**Language**

If more than one language is defined within the questionnaire, select the language to use for the question and code labels in the export.

**Write Unicode data**

If you need your data in Unicode.

**Convert text using code page**

Select the code page to convert the export into. Make sure the selected code page matches the language of your questionnaire.

**Retain ODIN data format**

Keeps the exported data file (EXT file) in the same format as the original NIPO ODIN data file (DAT file). Included open-ended answers, if any, are appended at the end of each record.

**Include open answers**

Includes open-ended answer verbatim in the data file as additional variables.

**Max record length**

The maximum number of positions of a single record. Check your statistical analysis software for the limits. Longer records are split at the configured threshold.

**Include questions without text and unused questions**

Exports questions without text and questions that are never shown due to routing. By default, these are not exported.

**Include null-evaluating filter and dummy questions**

Exports questions that use a filter that always evaluates to false (for example `*IF [ 1 = 0]`) and `*DUMMY` questions. By default, these are not exported.

**Include save script**

Includes the line `"SAVE OUTFILE='survey.SAV' /COMPRESSED"` where `survey` is the name of the survey.

**Variable name in front of question text**

The question label is preceded by the export name of the variable. Optionally add a line feed to the variable name.

**‘Question n’ in front of question text**

All question texts are preceded by the word "Question" followed by the question number originally used in the questionnaire.

**Use question-ID's as variable names**

If specified, use names defined with `*VAR` as variable names.

**Use alternative names for random and order variables**

Not applicable to Nfield.

**Write random and order numbers**

Not applicable to Nfield.

**Use code numbers for Multiple Dichotomy fields**

By default, for `*MULTI` questions all codes are exported as multiple dichotomy questions (mentioned / not mentioned). This option places the original code number in the question label for these questions. If disabled, the dichotomy questions are renumbered and do not carry any relation to code numbers used in the questionnaire.

**Text for mentioned**

For multiple dichotomy questions, sets the label to be used for 'mentioned'.

**Text for not mentioned**

For multiple dichotomy questions, sets the label to be used for 'not mentioned'.

**Variable name length**

Sets the maximum length for variable names. Choose either **8** or **64**, dependent on your SPSS version in use.

**Variable label length**

Sets the maximum length for the variable label (the question text). Longer texts are truncated at the threshold.

**Value label length**

Sets the maximum length for the value label (the code label text). Longer texts are truncated at the threshold.

**Variable names first letter**

Sets the first letter for an exported variable name. This only happens question IDs are not used, or if the question does not have a question ID defined (\*VAR).

**Number of chars from multi question**

By default, multiple dichotomy questions receive the full original question text in addition to the code label. This option configures the maximum number of characters from the question label to include.

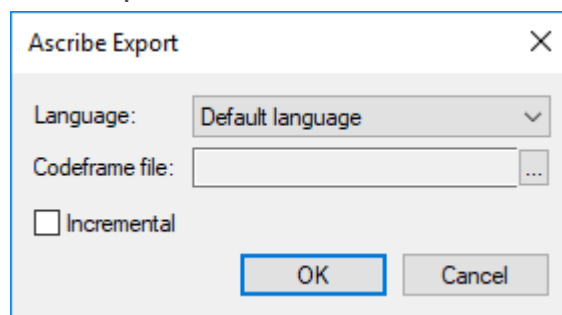
**Note:**

The selected settings for variable names may cause the export to generate variable names that exceed the maximum number of characters for the SPSS version you are using. A warning is issued for the minimum limit for older versions - adjust the settings if required.

**2.5.5 Ascribe**

Open-ended answers in NIPO Fieldwork System surveys may be coded using Language Logic's on-line coding solution *Ascribe*. Ascribe imports coding projects using a proprietary (ZIP compressed) XML format.

An export to Ascribe requires that the survey data file is present in same the directory as the questionnaire that is exported. For a questionnaire NAMEQ, the data file should be named NAME.DAT.

**Ascribe Export**

**Language**

If more than one language is defined within the questionnaire, select the language to use for the question and code labels in the export.

### Codeframe file

If the project was previously coded, the current questionnaire may not contain the required Code Frames. Select the Ascribe Code Frame file to use for coding.

### Incremental

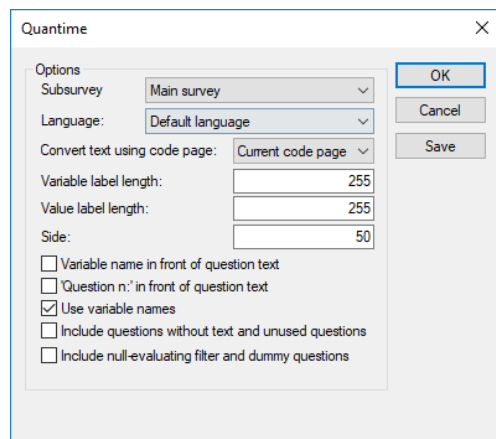
Check this option if you plan to add the current export to an existing Ascribe project. Use this feature to add new data to an existing project.

The result file is called `[surveyname]_setup.zip` where `surveyname` is the name of your survey. See the Ascribe documentation for details on how to create a project from this file or how to add incremental data to an existing project.

## 2.5.6 Quantime

An export to Quantime requires that the survey data file is present in same the directory as the questionnaire that is exported. For a questionnaire `NAME.Q`, the data file should be named `NAME.DAT`.

### Export to Quantime



The screenshot shows a dialog box titled "Quantime" with a close button (X) in the top right corner. The dialog contains several options for exporting data to Quantime:

- Options:**
  - Subsurvey:** A dropdown menu currently showing "Main survey".
  - Language:** A dropdown menu currently showing "Default language".
  - Convert text using code page:** A dropdown menu currently showing "Current code page".
  - Variable label length:** A text input field containing "255".
  - Value label length:** A text input field containing "255".
  - Side:** A text input field containing "50".
- Checkboxes:**
  - ☐ Variable name in front of question text
  - ☐ "Question n:" in front of question text
  - ☒ Use variable names
  - ☐ Include questions without text and unused questions
  - ☐ Include null-evaluating filter and dummy questions

On the right side of the dialog, there are three buttons: "OK" (highlighted with a blue border), "Cancel", and "Save".

### Include questions without text and unused questions

Exports questions without text and questions that are never shown due to routing. By default, these are not exported.

### Include null-evaluating filter and dummy questions

Exports questions that use a filter that always evaluates to false (for example `*IF [ 1 = 0 ]`) and `*DUMMY` questions. By default, these are not exported.

### Subsurvey

Select which survey to export if `*NEW` was used within the questionnaire.

### Language

If more than one language is defined within the questionnaire, select the language to use for the question and code labels in the export.

**Convert text using code page**

Select the code page to convert the export into. Make sure the selected code page matches the language of your questionnaire.

**Variable label length**

Sets the maximum length for the variable label (the question text). Longer texts are truncated at the threshold.

**Value label length**

Sets the maximum length for the value label (the code label text). Longer texts are truncated at the threshold.

**Side**

Defines the default left column width for the question text and code labels.

**Variable name in front of question text**

The question label is preceded by the export name of the variable. Optionally add a line feed to the variable name.

**'Question n' in front of question text**

All question texts are preceded by the word "Question" followed by the question number originally used in the questionnaire.

**Use variable names**

Instead of automatically assigning question variable names based on question numbers, use the labels defined by \*VAR.

**Include questions without text and unused questions**

Exports questions without text and questions that are never shown due to routing. By default, these are not exported.

**Include null-evaluating filter and dummy questions**

Exports questions that use a filter that always evaluates to false (for example \*IF [ 1 = 0 ]) and \*DUMMY questions. By default, these are not exported.

## 2.6 Imports

The imports for the NIPO ODIN Developer are used to import open-ended coding projects from Ascribe. This re-exports data previously exported to Ascribe. These imports immediately start an export for the data analysis software selected in the import dialog.

### 2.6.1 Ascribe

This imports an Ascribe export file. Once a project has been partially or fully coded in Ascribe, it may be used to merge the coded open-ended answers back into the data file.

Since this import issues and export, it requires that the survey data file is present in same the directory as the questionnaire that is exported. For a questionnaire `NAMEQ`, the data file should be named `NAME.DAT`.

#### Ascribe import

#### Language

Sets the translation to use as defined within the coding project. This only applies if multiple translations of questions and codes have been made.

#### Input file

Selects the `.ZIP` export file to use. Note that this export file must match the currently opened questionnaire file.

#### Convert single to multi if coded multi

If any previously single-coded questions have been changed to multiple-coded questions, they are converted. Otherwise, only the first selected code is accepted.

#### Output format

Sets the export type to be used. Confirming this dialog brings you to the relevant export dialog.

## 2.7 NIPO ODIN Developer Configuration

## 2.7.1 Syntax Highlighting

The NIPO ODIN Developer allows for script highlighting in the script editor. You can define the editor font and the foreground and background colors for:

- Text
- Text selection
- Number
- Comment (remarks)
- Keywords (NIPO ODIN commands)
- Routing commands
- Variable declarations

Select ODIN > Fonts... from the menu to configure syntax highlighting.

---

**Note:**

Syntax highlighting does not work in bi-directional (Hebrew, Arabic) mode.

---

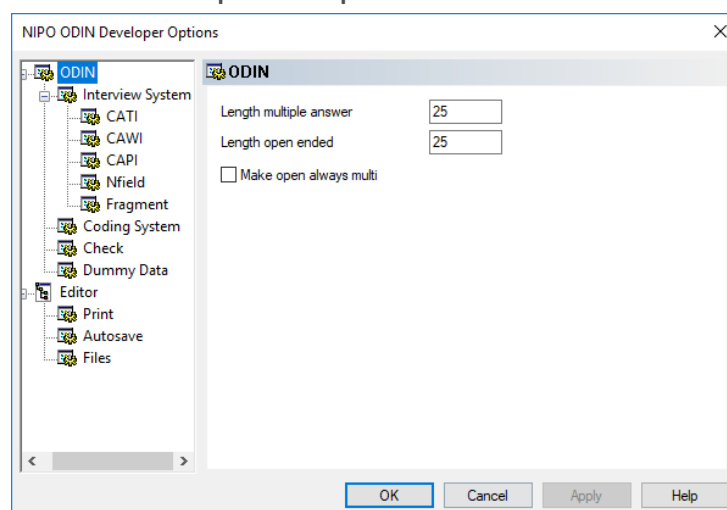
## 2.7.2 Application Configuration Options

Select **Settings > Options** from the menu to change the application configuration options. These are described in this section.

### 2.7.2.1 ODIN Options

This dialog sets various options for the pop-up menu action to insert template questions.

#### NIPO ODIN Developer ODIN options



#### Length multiple answer

Sets the default length inserted for multiple-coded questions.

**Length open ended**

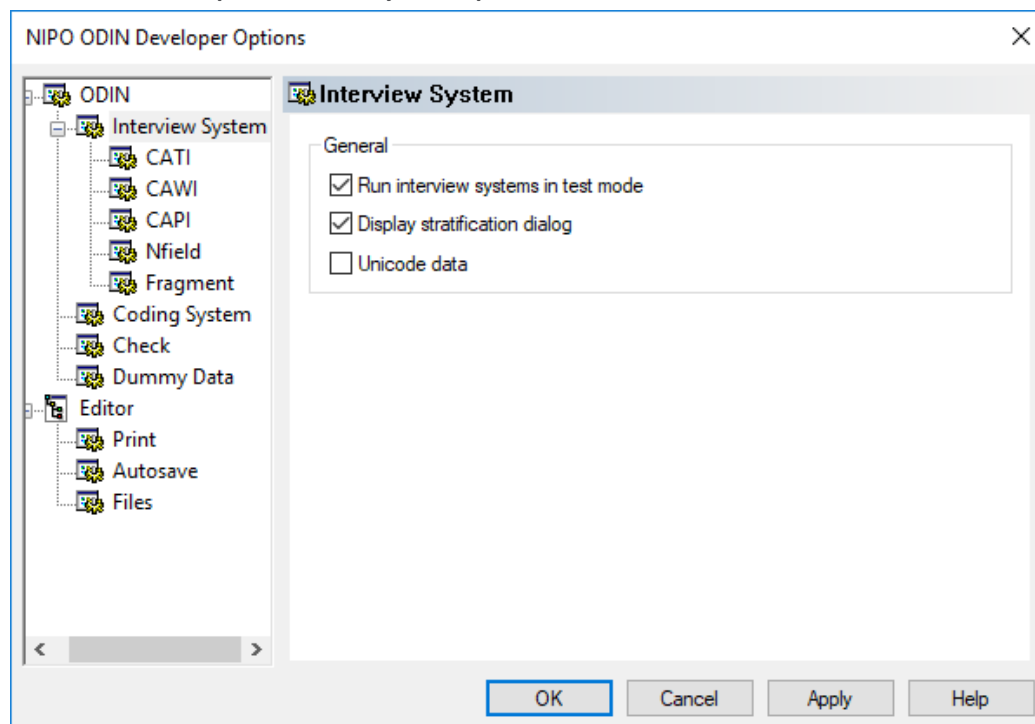
Sets the default length inserted open-ended questions.

**Make open always multi**

Adds \*MULTI to the \*OPEN question template.

**2.7.2.2 Interview System Options**

The configuration options in the **Interview System** section configure the application file locations for the NIPO CATI and Nfield previews and various preview-related matters.

**NIPO ODIN Developer Interview System Options****Interview System Options****Run interview systems in test mode**

When in test mode, no result data is stored for a test interview. Otherwise, DAT-file and O-file data is created from a preview.

**Display stratification dialog**

For every \*STRAT command in the questionnaire, the preview asks if it should follow the routing for 'stratification reached'. Note that no stratification file or sample is tested for this option.

**Unicode data**

When not in test mode, this option stores the DAT-file and O-file records in Unicode.

### 2.7.2.3 CATI

#### CATI Client

Sets the location of the NIPO CATI Client application, which is required for the NIPO CATI preview. By default, it can be found in the NIPO ODIN Developer application directory. Select `ODQES.EXE` for the non-Unicode version or `ODQESU.EXE` for the Unicode version.

#### Adjustable screen size

Allows the NIPO CATI Client window to be resized. By default, the window is maximized and appears on top of all other applications.

#### Single shot

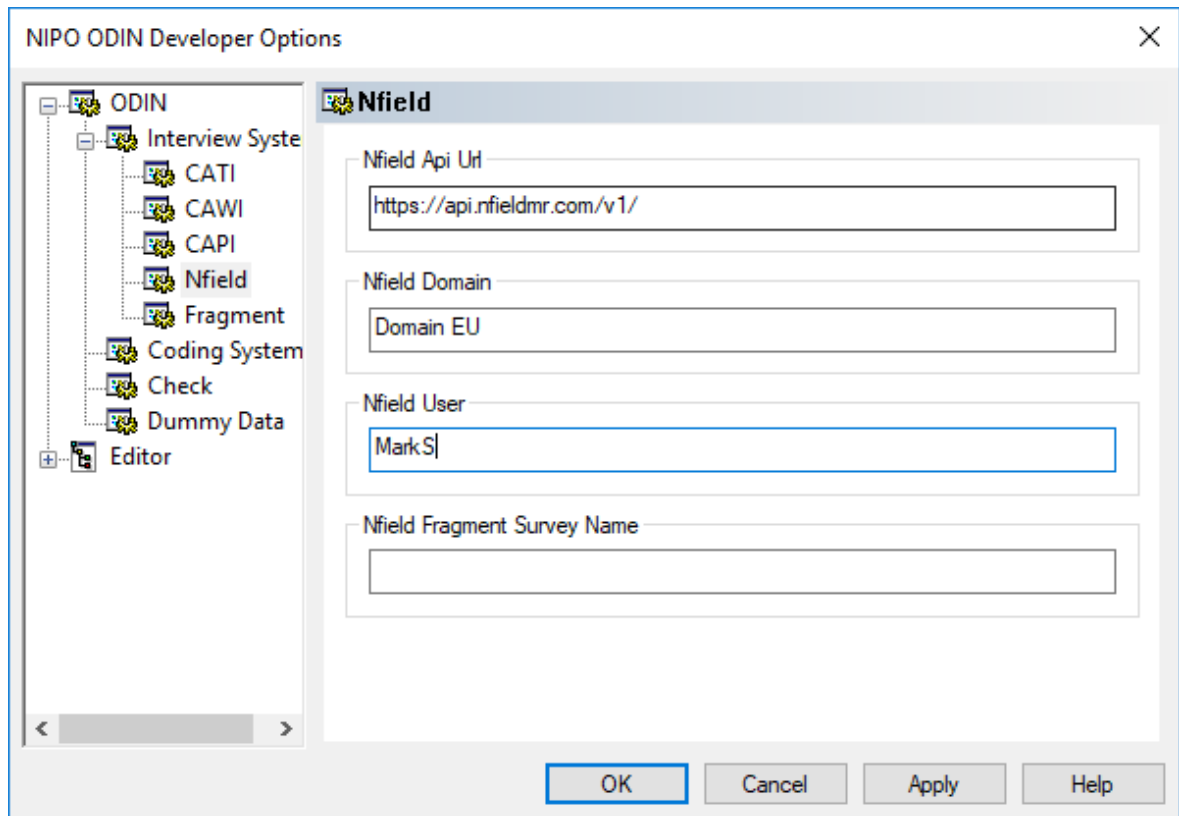
If set, automatically ends the preview at the end of the questionnaire. Otherwise, it restarts.

### 2.7.2.4 How to setup the Nfield interview system

To setup the Nfield interview system (to be able to run preview interviews in Nfield), please enter the Nfield API URL for your area:

- EU: <https://api.nfieldmr.com/v1/>
- Americas: <https://apiam.nfieldmr.com/v1/>
- Asia Pacific: <https://apiap.nfieldmr.com/v1/>
- China: <https://apicn.nfieldcn.com/v1/>

Also please enter the domain name and user name that were created for you by NIPO.



### 2.7.2.5 Coding System Options

The **Coding System** options are obsolete in this version of the NIPO ODIN Developer.

### 2.7.2.6 Check Options

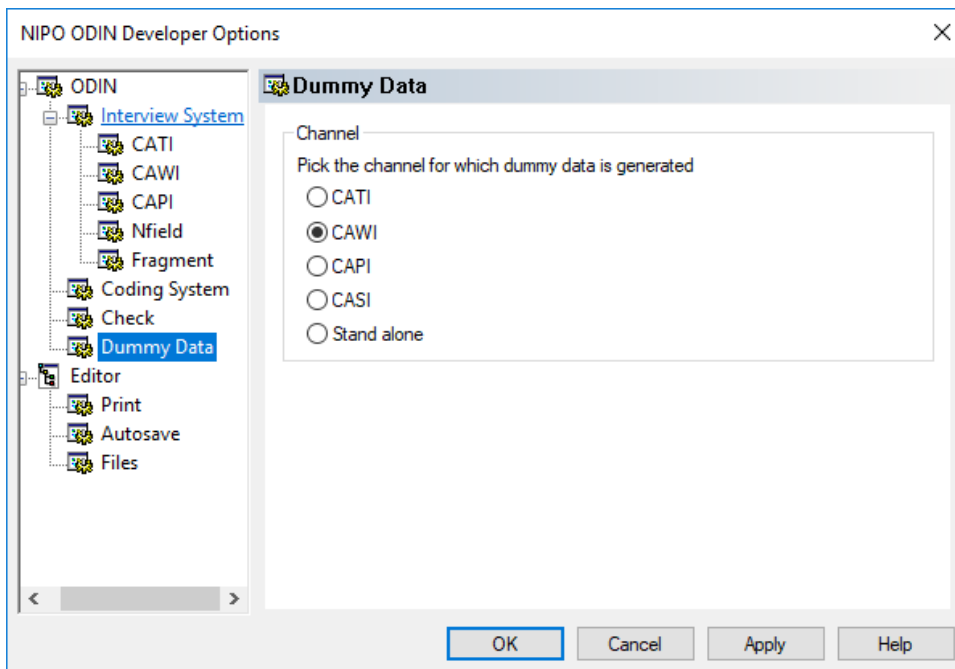
The **Check** options are described in section [2.2](#).

### 2.7.2.7 Dummy Data Options

This section configures the **ODIN > Generate Dummy Data...** feature. In particular, it specifies what channel is assumed to create dummy data. You may opt to select the required channel if your questionnaire contains any channel-aware routing.

**Note:** this generates dummy data for NFS only (not for Nfield). You can use this to check your logic provided it is not dependent on template commands.

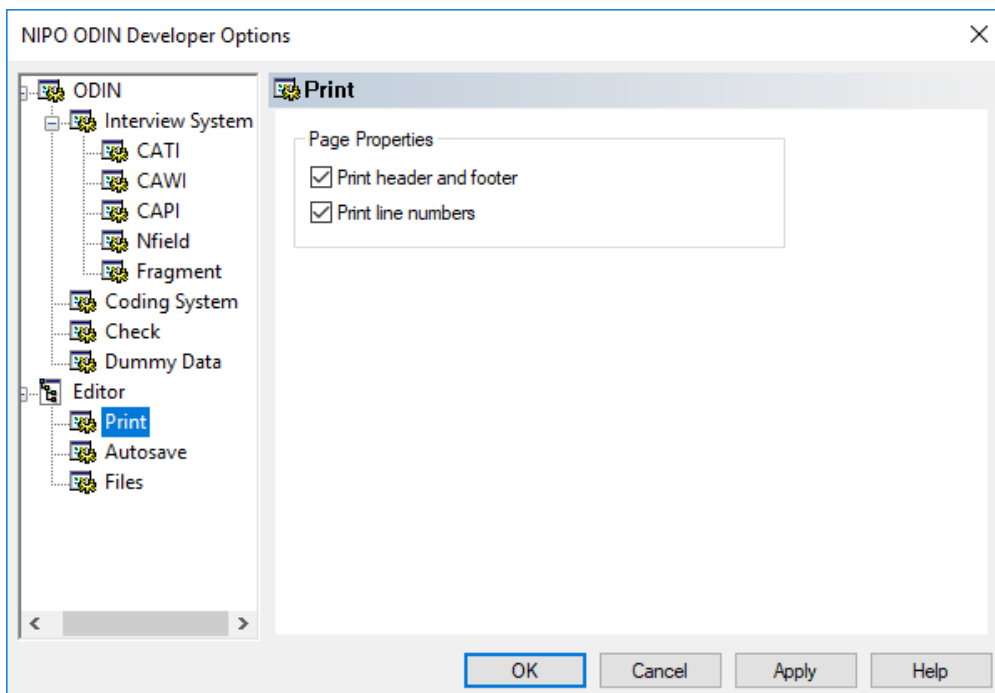
### NIPO ODIN Developer Dummy Data Options



### 2.7.2.8 Print Options

The **Print** options allow you to define some settings when printing the currently opened questionnaire. Note that long lines are automatically wrapped.

### NIPO ODIN Developer Print Options



**Print header and footer**

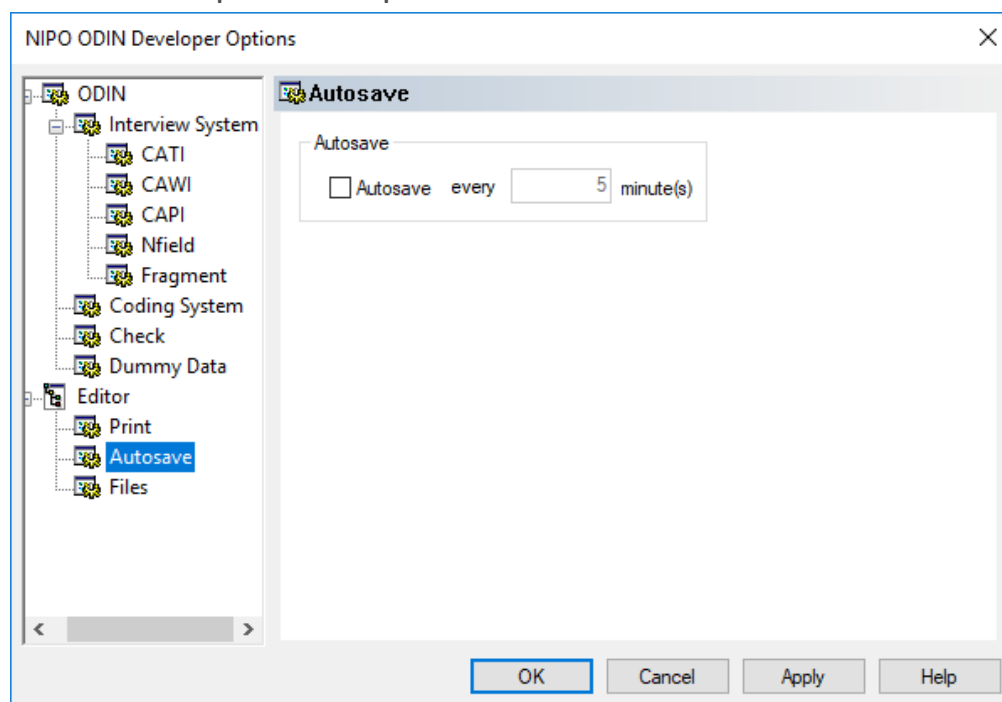
Prints header and footer information: the questionnaire file name and a page number.

**Print line numbers**

Prints line numbers in front of the code lines, except where lines are wrapped.

**2.7.2.9 Autosave Options**

The **Autosave** options make the NIPO ODIN Developer save backups of changed questionnaire files automatically at configurable intervals.

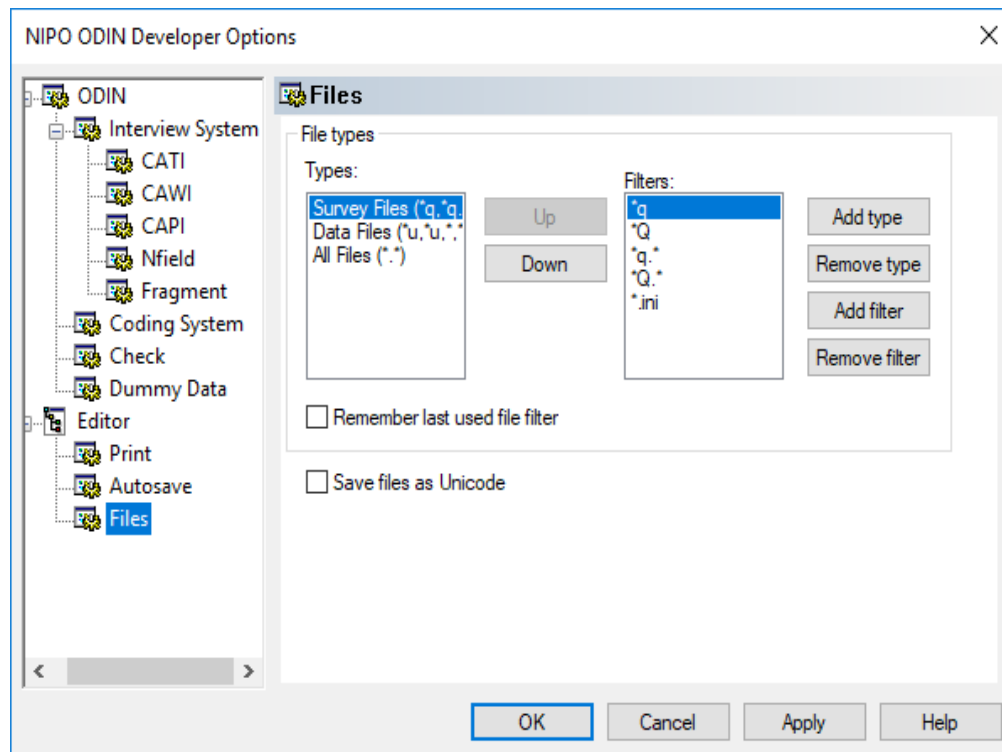
**NIPO ODIN Developer Autosave options**

Select **Autosave** to enable auto-saving. Optionally change the number of minutes between saves. Once configured, **Autosave** saves files at the configured interval. If a file is closed in a regular fashion, its related backup file is deleted. If execution of the NIPO ODIN Developer terminates unexpectedly, for example due to a system crash or a power failure, the backup files can be found in the folder where the original is saved. Auto-backups contain the extension **OBK**. Opening any file for which a backup is available also opens the backup file.

**2.7.2.10 Files Options**

The behavior of the **File > Open** and **File > Save** dialogs can be configured in the Files options.

## NIPO ODIN Developer Files Options

**File types**

This section allows you to configure the file types filtered by the NIPO ODIN saving and loading dialog boxes.

Click **Add type** to create a new type. You are asked to specify the name of the type as it appears in the drop-down box in the file dialogs. Note that it is good custom to include the file filters in round brackets. Click **Remove type** to remove unwanted types. In addition, select a type and click **Up** or **Down** to change its order in the list.

For every type, click **Add filter** to add a file filter. More than one file filter per type may be specified. Select a filter and click **Remove filter** to delete it again.

**Remember last used file filter**

Automatically selects the last used file filter when reopening a Open or Save dialog.

**Save files as Unicode**

Automatically sets the file encoding to Unicode for file saves.



# 3. Using the NIPO ODIN Script Language

This section provides an overview of NIPO ODIN script language syntax and features.

## 3.1 Naming Conventions

Names of variables, subroutines and lists have to be unique and may consist of a maximum of 12 characters. Only letters, underscores and numbers are allowed. The first character must be a letter.

### 3.1.1 Questions

Question numbers are made up of positive numbers ( $\geq 1$ ). No alphabetic (except X, see below) or other characters are allowed. Each question number has to be unique within one (sub-) questionnaire. Question numbers can be used indifferently. The NIPO ODIN Developer allows you to renumber the questionnaire, but this is not deemed necessary. Instead of numbers you can also use the character X on each question. In this case the question numbers are not defined, and routing has to be done afterwards. When you renumber the questions, question numbers will be inserted.

The highest question number is 2147483647.

The `*QUESTION` command may be abbreviated to `*Q`.

#### Example question definitions

Valid question definitions	Invalid question definitions
<code>*QUESTION 1</code>	<code>*QUESTION 0</code>
<code>*QUESTION 2</code>	<code>*QUESTION -2</code>
<code>*Q 3</code>	<code>*QUEST 3</code>
<code>*QUESTION 2501</code>	<code>*QUESTION 25a</code>
<code>*QUESTION 109</code>	<code>*QUESTION 10.9</code>
<code>*QUESTION 11001</code>	<code>*QUESTION 11.001</code>
<code>*QUESTION 102</code>	<code>*QUESTION 1-2</code>
<code>*QUESTION 2147473647</code>	<code>*QUESTION 2234567890</code>
<code>*QUESTION 1234567890</code>	<code>*QUESTION 1 234 567 890</code>

### 3.1.2 Data Fields

In the NIPO ODIN Scripting Language it is not necessary to define the starting position of a data field when you define a question. Only the length of the data field is required. NIPO ODIN keeps track of all the positions used in the questionnaire. In the NIPO ODIN Developer, you can automatically insert the starting positions using the **Fix Positions** command. The starting positions always have to be defined for Nfield, so please always fix your positions.

In a closed, but not multiple question the number of digits of the highest code number defines the minimum length of the data field. In a closed, multiple question the highest code number (rather than number of codes) defines the minimum length of the data field.

In numeric questions the minimum and maximum length depend on the answer that you expect. The maximum size is approximately 16 positions, including decimals and a floating point.

The highest position for a data field is 200.000. Make sure your analysis program can handle the data size.

## 3.2 Variables

You can store a text or a value in a variable to display in a question or to use for routing. Variables have to be defined with `*VARS`, `*TEXTVARS` before they may be used. Variable names are not case-sensitive.

A variable can be filled with the `*PUT` or the `*SAVE` command. Its contents can be displayed with the `*?` command. Note that the contents of the variable depend on the type of variable.

A numeric variable (`*VARS`) may contain any value. Use the `string manipulation routines` to change the on-screen appearance of numeric values. When text or a text variable is saved in a numeric variable, the NIPO ODIN syntax check will give a warning message.

`*TEXTVARS` defines a text variable and may contain a text of unlimited length. The text may contain CR/LF which will be used when displaying the variable. Variables may *not* contain NIPO ODIN commands, except for `*FONT`.

You can also create a variable array: `*Vars Brand[10]` This creates one variable, an array of 10 numbers. You can now reference each of the variables in the array as: `Brand[1]`, `Brand[2]`, `Brand[3]`, etc.

---

#### Note:

The contents of variables are not stored in the data file automatically. So, make sure that if you base questions or routing on a variable, you store this information in a (dummy) question.

---

## 3.3 System Variables

In addition to custom-defined variables, NIPO ODIN recognizes system variables for various purposes. Unless mentioned, these system variables do not need to be defined using `*VARS`.

### 3.3.1 Test Mode Aware Scripting

Test Mode Aware Scripting allows you to mark the interviews running in the test mode by using the `_ISTEST` variable.

When an interview is a test interview `_ISTEST` is set to 1 (*true*) or to 0 (*false*) otherwise.

#### Example routing for Test Mode Aware Scripting

```
*QUESTION 1 *CODES 61L1 *IF [ _ISTEST ]
This question will appear only for test interviews

1: OK

*QUESTION 2 *CODES 62L1
This question will appear on all platforms
1: OK
```

**Note:** When running the preview or test link with a script that includes `_ISTEST`, if the interviewer halts for 15 minutes or more at a screen, the interview gets suspended. If one then resumes the interview, all statements under `_ISTEST` condition are skipped.

### 3.3.2 Repeat Number

The operator `?R` contains the current value of a `*REPEAT` loop, a number between and including 1 and `n` where `n` is the value specified at the `*REPEAT` statement. Note that this is not the *iteration* number - for example, in the third execution in the loop the value of `?R` is not necessarily 3. This is especially true for randomized, controlled, or rotated loops.

In nested blocks, the operator `?R` returns the value of the loop where it is located. To access values from loops above or below the current one, you should use a question or a numerical variable to pass the value. The operator does not exist outside the scope of a `*REPEAT` block.

`?R` is a system variable and can *not* be changed (for example by using `*SAVE` or `*PUT`) nor can it be displayed directly in question text using the `*?` command. Copy the value to a variable another variable to access it (e.g. `*PUT Temp [?R]`).

#### Example using the repeat number

```
*TEXTVARS paper

*QUESTION 2 *CODES L7 *MULTI
Which of the following newspapers do you know?

1: La Repubblica
2: La Stanza
3: The Mirror
```

```

4: The New York Times
5: Le Figaro
6: La Libération
7: None *NMUL

*REPEAT 6 *CONTROL Q2 W
*PUT paper Q2, ?R

*QUESTION 3 *CODES L1
How often do you read *? paper?

1: Daily
2: Once a week
3: Once a month
4: Don't know

*ENDREP

```

In nested blocks you have to transfer the repeat number ?R of the outer block to a numeric variable first to be able to use it in the inner block.

#### Example storing the repeat value

```

*VARS X,Y,Z
*REPEAT 3
*PUT X [?R]
*REPEAT 2
*PUT Y [?R] ** this line is only necessary to display the value of ?R
*PUT Z [X+?R] ** or *PUT Z [X+Y]
*QUESTION 1

*?X + *?Y = *?Z

*ENDREP
*ENDREP

```

### 3.3.3 Timer

**Please note** that timer only works in CAPI.

Stopwatch[3] can be used as a timer that you have to reset yourself setting the value to 0. This timer starts as soon as the interview is started or once it is reset.

The script below uses the timer to measure the time of selecting a code, pressing enter and measuring the length of a (section of) an interview.

#### Timers Example

```

*PUT Stopwatch[3] [0]
*QUESTION 1 *CODES L1
This is a question.
1: Continue

*PAGE
The total time to conduct this interview was *? Stopwatch[3] seconds
*END

```

### 3.3.4 Elapsed Time Function

To know how much time a respondent spend answering a (set of) question(s) there is a function `?Elapsedtime (QuestionID)`.

It measures the time spent on a question: if the same question has been visited multiple times, it holds the total time spent. This functionality also simplifies checking how long certain sections of your questionnaire take so that you can detect the so called “speeders”.

This function takes a list of question IDs separated by comma(s) as an input and returns the sum of time spend on those questions. That number matches the elapsed time you would find in the audit log.

#### Example 1

```
*TEMPLATE NfieldChicago

*QUESTION 1 *CODES 61L1 *ID FirstQ
First question
1:ok
2:no

*QUESTION 2 *CODES 62L1 *ID SecondQ
Second Question
1:ok
2:no

*QUESTION 3 *CODES 63L1 *ID ThirdQ
Third Question
1:ok
2:no

*VARS EsTQ[4]

*PUT EsTQ[1] [?ELAPSEDTIME(Q{FirstQ})]
*PUT EsTQ[2] [?ELAPSEDTIME(Q2)]
*PUT EsTQ[3] [?ELAPSEDTIME(Q3)]
*PUT EsTQ[4] [?ELAPSEDTIME(Q1,Q{SecondQ},Q3)]

*PAGE
First Q: *?EsTQ[1]
Second Q: *?EsTQ[2]
Third Q: *?EsTQ[3]
Total Q: *?EsTQ[4]
```

For repeated questions, like repeat blocks or subroutines we have the following rule:

- If `?Elapsedtime()` is executed within a `*REPEAT` or a subroutine, it will return the value for that iteration (for a `*REPEAT`) or the value for that instance (for a subroutine).
- If elapsed time is called outside of `*REPEAT` or subroutine, while the question itself is in one of these, it will return the sum of elapsed times for every iteration or instance of that question so far.

#### Example 2

```
*TEMPLATE NfieldChicago
```

```

*PAGE
Now let's do a repeat.

*REPEAT 2 *FIELD 70L2

*VARS RepNum
*PUT RepNum [?R]

*QUESTION 4 *CODES 1L1 *ID RepeatQ
Question for repeat iteration *? Repnum
1:ok
2:no

*VARS ETRepQ
*PUT ETRepQ [?ELAPSED TIME(Q{RepeatQ})]

*PAGE
The question in this repeat took *? ETRepQ seconds

*ENDREP

*VARS ETTTotalRepQ
*PUT ETTTotalRepQ [?ELAPSED TIME(Q{RepeatQ})]

*PAGE
The whole repeat took *? ETTTotalRepQ seconds.

```

**Note:** if a question is skipped due to routing, the `?ElapsedTime()` will return 0 for that question.

### 3.3.5 Language

The `LANGUAGE` variable contains the name of a define language section currently in use. See the [\\*LANGUAGE](#) command for more details.

## 3.4 Expressions

NIPO ODIN supports the use of expressions in questionnaires. An expression may indicate all sorts of data but will always result in a value. Expressions are used in two different ways:

- As a *boolean*; the result of the expression is either false (the expression has as result value 0) or true (the expression has as result the value 1).
- To make calculations or manipulations with a value as result.

Some commands will require or allow for an expression as argument. Expressions have to be enclosed in square brackets.

Expressions are evaluated from left to right and operators apply to the operands immediately left and right. NIPO ODIN expressions do not follow the regular precedence rules! If a left-to-right evaluation is not appropriate, you can use parenthesis to evaluate an expression first.

### 3.4.1 Expression Operators

#### Expression operators

Operator	Description	Use	Note
Q	Question reference	Qn	Reference to question n or its contents
	Length of field	nLm	Position n, length m
F	Field test	QxFn	Contents of field (cell) number n within form-question x
S	Refers to the statement in a matrix	QxSn	Answer of question x for the nth statement
,	code-test	Qx, n	1 if present, otherwise 0
-	code-string	Qx, n-m	Code string
-	minus sign	-n	Negation of n
#	not	#Qx	1 if no answer in question x, otherwise 0
#	negation	#n	1 if value n not true, otherwise 0
+	add	n+m	Sum of n and m
-	subtract	n-m	Difference of n and m
*	multiply	n*m	Product of n and m
/	divide	n/m	Division of n and m
^	exponent	n^m	Exponent m of n
&	logical AND	n&m	1 if both not 0, otherwise 0
\	logical OR	n\m	0 if both 0, otherwise 1
RAN	random value	RAN n	Random value from 0 up to and including n-1
=	equal to	n=m	Results in 1 if true or 0 if false
<	less than	n<m	Results in 1 if true or 0 if false
>	more than	n>m	Results in 1 if true or 0 if false
<=	less than or equal to	n<=m	Results in 1 if true or 0 if false
>=	more than or equal to	n>=m	Results in 1 if true or 0 if false

Operator	Description	Use	Note
<>	not equal to	n<>m	Results in 1 if true or 0 if false
?JSON	Returns the value for a value pair from a JSON string	?JSON (JSONString, "ValueName"]	Will not return anything if the ValueName does not match or the JSONString is an invalid JSON
?BUTTON (Qn)	Returns the number of the button if the question was answered with a button.	*IF [?BUTTON (Q1) = 2] *GOTO 2	Number of the button if the question was answered with a button, or -1 if the button was not used.
?R	repetition number	?R	current (logical) repetition number
TO	Range	n1 TO n2	Range n1 through n2
;	separate ranges	n1 TO n2 ; n3 TO n4	Range n1 through n2 or n3 through n4
QxM	order of mentions in a *MULTI question	QxM	
QxR	display order of a question, when using *RAN or *ROT	QxR	
QxMy	order of mentioned number y in a *MULTI question. Returns the code number of the y mentioned from question x.	QxMy (see example below)	
QxRy	display order Used in combination with *ORDERnumber y of a question, when using *RAN or *ROT. Returns the code number of the y randomized or rotated.	QxRy	

### Question reference

The reference to a question by means of Q<sub>n</sub> where n represents the question number. For example, Q<sub>10</sub> references question 10. If it concerns a code test then more code values may occur; these will be treated as a logical or.

### Code string

Any given number of codes separated by one or more commas or a code range indicated by a dash. Valid codes consist of the digits 0 up to and including 9, or B for a blank value. For example, Q<sub>2</sub>, B returns 1 (*true*) if Q<sub>2</sub> is blank, Q<sub>3</sub>, 1-5 returns 1 if in Q<sub>3</sub> any of the codes in the range 1-5 are marked, and Q<sub>4</sub>, 2, 4, 8, 12 returns 1 if Q<sub>4</sub> contains any of the codes 2, 4, 8, or 12 are marked.

Values n or m

The values n or m may represent any of the following:

- Number
- Expression or variable
- Text enclosed in double quotes with, possibly, the embedded contents of a variable

Example of usage QxMy

Out of 5 choices in a multiple question 1, the user asked to select between three and five options. In question 3, the first 3 selected options are shown in the order of selection (the 1<sup>st</sup> selected option is shown on top, the 2<sup>nd</sup> is shown below it, etc.).

```
*QUESTION 1 *CODES 61L5 *MULTI
Please choose at least 3 options
1: one
2: two
3: three
4: four
5: five

*QUESTION 2 *CODES 66L5 *MULTI *DUMMY
1: one
2: two
3: three
4: four
5: five

*INCLUDE Q2 [Q1M1]
*INCLUDE Q2 [Q1M2]
*INCLUDE Q2 [Q1M3]

**Display to the user the 1st 3 selected choices in order of selection.
*QUESTION 3 *CODES 88L9 *MULTI *CONTROL Q2 W *ORDER Q1M
1: one
2: two
3: three
4: four
5: five
```

3.4.2 Examples of Expressions

Examples of expressions are presented here. Note that expressions are always specified between square brackets. The expressions in these examples may be either assigned to a variable or applied to an \*IF filter.

Expression Examples

Expression	Results in
[ 100 ]	The value 100.
[ RAN 100 ]	A random between 0 and 99 inclusive.
[ Total[3] ]	The value stored in the third element of the number array TOTAL.
[ TOTAL2 * 3 + 10 ]	the value stored in the variable TOTAL2, multiplied by 3 and then increased by 10.

[ Q11,1 ]	1 if code 1 was marked in Q11, otherwise 0.
[ Q12,1,2,3 ]	1 if any of the codes 1, 2 or 3 were marked in Q12.
[ Q13 >= 18 ]	1 if the value in Q13 is equal to or larger than 18.
[ Q11,1 \ Q12,1,2,3 ]	1 if code 1 is marked in Q11 or if code 1, 2 or 3 is marked in Q12.
[ Q12,1,2,3 & Q13 >= 18 ]	1 if code 1, 2 or 3 is marked in Q12 and Q13 is equal to or larger than 18.
[ Q12 ]	The value of Q12 if Q12 is a single-coded or numerical question; the highest marked code if Q12 is a multiple-coded question.
[ Q13 = Q14 ]	1 if the (highest) value of Q13 is equal to the (highest) value of Q14
[ # (Q5 \ Q6) & Q7 = 1 ]	1 if either Q5 or Q6 does not contain a value and Q7 equals 1. The brackets are required to let the expression evaluate Q5 and Q6 first.
[ Q5 , 1 TO 3 ]	1 if any of the codes 1, 2 or 3 are marked in Q5.
[ Q5 , 1 TO 3 ; 6 TO 8 ]	1 if any of the codes 1, 2, 3, 6, 7, or 8 are marked in Q5.

### 3.4.3 Common Mistakes in Expressions

This section describes common mistakes made in expressions, where a syntax check does not reject the syntax used but where a misunderstanding exists on how the expression is evaluated. These examples focus on filtering constructions.

#### Incorrect expressions and how to correct them

Incorrect expression	What it was intended to do	What it does	Correct expression
*IF [ 20 < Q2 < 40 ]	Check if Q20 is between the values 20 and 40.	This firsts checks if 20 is smaller than Q2 (result 0 or 1) then if the result is smaller than Q40.	*IF [ Q2 > 20 & Q2 < 40 ]

<code>*IF [ Q20, 3\4\5 ]</code>	Check if Q20 has code 3, 4 or 5 marked.	This checks if Q20 has code 3. Then it checks if either the result of the expression, or 4, or 5 are true. Any value above 0 is always considered true, therefore the expression always results in 1.	<code>*IF [ Q20, 3, 4, 5 ]</code>
<code>*IF [ Q21=1, 2, 3 ]</code>	Check if Q21 has code 1, 2 or 3 marked.	This checks if Q21 is equal to the expression 1,2,3. The latter expression checks if position 1 in the DAT-file contains a 2 or a 3.	<code>*IF [ Q21, 1, 2, 3 ]</code>
<code>*IF [ Q21 = 1-3 ]</code>	Check if Q21 has code 1, 2 or 3 marked.	This checks if the value of Q21 equals -2 (1-3).	<code>*IF [ Q21, 1, 2, 3 ]</code>
<code>*IF [ #Q21 = 1 ]</code>	Check if Q21 is not equal to 1	This first checks if Q21 is not empty, then compares the result to 1. In other words, the expression returns 1 if Q21 is not empty.	<code>*IF [ Q21 &lt;&gt; 1 ]</code>

## 3.5 String manipulation routines

NIPO ODIN includes a set of routines for text (string) and text variable manipulation. These routines can be used for various purposes, including (but not limited to):

- Checking if a string meets the requirements for input. For example, a routine to check the respondent input is an Email address with valid syntax.
- Reformatting text to make it more suitable for displaying the text in follow-up questions.
- Reformatting text to make it more suitable for storage in output data.

- Checking if a text has certain text (pattern) matches, for example for finding answers in \*OPEN ended questions.

NIPO ODIN considers strings as ranges of unique characters and is therefore indifferent to code page use.

All string manipulation routines expect a text variable or a static text string as input parameter. If the answer to a question must be checked, use \*SAVE or \*PUT to store the answer in a text variable first. Strings routines are considered expressions, and therefore always require square brackets to be evaluated.

### 3.5.1 STRFINDMATCH

#### Syntax

```
?STRFINDMATCH(<string>,<regex>)
```

#### Description

Returns a substring from a string that matches the first occurrence of a regular expression. If no match is found, an empty string is returned. NIPO ODIN follows the Microsoft regular expression implementation. Regular expressions are a powerful tool to find particular texts in an answer given by a respondent by using matching patterns rather than literal text. A complete reference for these regular expressions can be found on the Microsoft Developer Network (<http://msdn.microsoft.com/en-us/library/az24scfc.aspx>).

#### Arguments

<string>

String to search for the regular expression.

<regex>

Regular expression that the substring must match.

#### Example

The following example retrieves the first number found in an open-ended answer.

```
*VARS value
*TEXTVARS answer

*QUESTION 10 *OPEN L1 *SAVE answer
Let me find the value in your text:

*PUT value [?STRFINDMATCH(answer,"^\d+")]

*Q 20
Value found: *? value
```

### 3.5.2 STRHASMATCH

#### Syntax

```
?STRHASMATCH(<string>,<regex>)
```

#### Description

Checks if a string matches a regular expression. Returns 1 if true, 0 otherwise. NIPO ODIN follows the Microsoft regular expression implementation. Regular expressions are a powerful tool to find particular texts in an answer given by a respondent by using matching patterns rather than literal text. A complete reference for these regular expressions can be found on the Microsoft Developer Network (<http://msdn.microsoft.com/en-us/library/az24scfc.aspx>).

#### Arguments

<string>

String to search for a match with the regular expression.

<regex>

The regular expression to search for.

#### Example 1

The following example ensures an answer only contains letters and digits.

```
*TEXTVARS zipcode

*Q 10 *ALPHA L4 *SAVE zipcode
Please enter your zip code:

*Q 20 *CODES L1 *IF [?STRHASMATCH(zipcode,"\\W")]
A zip code may only contain characters and letters.

1: Go back and correct *BACK 10
```

#### Example 2

This more advanced example uses a regular expression that checks the syntax validity of the Email address.

```
*TEXTVARS email

*Q 10 *ALPHA L40 *SAVE email
What is your Email address?

*PUT email [?STRLOWER(?STRTRIM(email))]

*Q 20 *CODES L1 *IF [#(?STRHASMATCH(email,"^([a-z0-9_\\-]+)@([\\da-z\\-]+)\\.([a-z\\.]{2,6})$"))]
It appears that *?email is not a valid Email address.

1: Go back and correct *BACK 10
```

### 3.5.3 STRINDEX

#### Syntax

```
?STRINDEX(<string>,<findstring>)
```

**Description**

Returns the start position of a substring in a string, or 0 if the substring was not found. The match must be case-sensitive. Note that 1 indicates the first position of a string.

**Arguments**

<string>

String in which to search.

<findstring>

String to search for.

**Example**

```
*TEXTVARS answer
*Q 10 *OPEN L1 *SAVE answer
What is your opinion of the European Union?

*Q 20 *IF [?STRINDEX(answer,"!")>0]
You seem to have formed a strong opinion on the subject.
```

**3.5.4 STRLENGTH****Syntax**

?STRLENGTH(<string>)

**Description**

Returns a number indicating the length of the specified string or text variable.

**Arguments**

<string>

String to calculate the length of.

**Example**

```
*TEXTVARS zip
*Q 10 *ALPHA L4 *SAVE zip
Enter the zip code:

*Q 20 *CODES L1 *IF [?(?STRLENGTH(zip)=4)]
The zip code must be 4 characters or digits!

1: Go back *BACK 10
```

**3.5.5 STRLOWER****Syntax**

?STRLOWER(<string>)

**Description**

Convert a string to lowercase.

**Arguments**

<string>

String to convert to lowercase.

**Example**

```
*TEXTVARS email

*Q 10 *ALPHA L40 *SAVE email
What is your Email address?

*Q 20 *ALPHA L40 *DUMMY
Email in lowercase

*COPY Q20 [?STRLOWER(email)]
```

**3.5.6 STRREPLACE**

**Syntax**

?STRREPLACE (<string>)

**Description**

Replaces text in a string.

**Arguments**

<string>

String in which you want to replace some characters.

<findstring>

The exact sequence of characters you want to replace.

<replacestring>

The exact sequence of characters you want as a replacement.

**Example**

```
*TEXTVARS OriginalString, FindString, ReplacementString, ResultString

*PUT OriginalString "Hello World"
*PUT ResultString [?STRREPLACE(OriginalString, "World", "Universe")]

*PAGE
*?OriginalString
*?ResultString
```

Please note:

- The `<string>` and `<findstring>` parameters cannot be empty.
- The function is case-sensitive.
- If the answer to a question must be checked, use `*SAVE` or `*PUT` to store the answer in a text variable first.
- String manipulation routines are considered expressions, and therefore always require **[square brackets]** to be evaluated.

### 3.5.7 STRSUBSTR

#### Syntax

```
?STRSUBSTR(<string>,<start>,<length>)
```

#### Description

Returns a substring of a string.

#### Arguments

`<string>`

String to return a substring of.

`<start>`

Start position of substring in the string.

`<length>`

Number of characters to return, starting from the start position. Use -1 to return the substring from the start position until the end of the string.

#### Example

```
*VARS len
*TEXTVARS answer

*REPEAT 10
  *Q 10 *OPEN LI *SAVE answer
  This is the echoing question. Type something.

  *PUT len [?STRLENGTH(answer)]
  *PUT answer [?STRSUBSTR(answer,len/2+1,-1)]

  *Q 20
  ... *? answer...

*ENDREP
```

### 3.5.8 STRTRIM

#### Syntax

```
?STRTRIM(<string>)
```

**Description**

Removes leading and trailing spaces from a string.

**Arguments**

<string>

The string to trim.

**Example**

```
*TEXTVARS name

*Q 10 *ALPHA L40 *SAVE name
What is your name?

*PUT NAME [?STRTRIM (name)]

*Q 20
Welcome to the questionnaire, *? NAME.
```

**3.5.9 STRUPPER****Syntax**

?STRUPPER(<string>)

**Description**

Convert the string to uppercase.

**Arguments**

<string>

String to return in uppercase.

**Example**

```
*TEXTVARS answer

*Q 10 *OPEN L1 *SAVE answer
What brand is your smartphone?

*PUT answer [?STRUPPER(answer)]

*Q 20
The following questions evaluate your experience with your *?answer.
```

**3.6 Form Field References**

Expressions may refer to fields in a \*FORM question using the F operator.

**Example of \*FORM field references**

```
*Q 10 *FORM *CONTROL Q5 W
At the garden center, how many items did you buy of each?

1: Flowers: *NUMBER L2  5: Flower price *NUMBER L2.2
2: Trees:  *NUMBER L2  6: Tree price  *NUMBER L2.2
3: Rakes:  *NUMBER L2  7: Rake price  *NUMBER L2.2
4: Shovels: *NUMBER L2  8: Shovel price *NUMBER L2.2

*VARS items
*PUT items [Q10F1+Q10F3+Q10F5+Q10F7]
```

The references with operator `F` do *not* refer to the codes used in the `*FORM` question, but to the index of the field when searching for fields from the top-left to the bottom-right of the screen in rows. In other words, the field behind "Flowers" is field 1, the field behind "Flower price" is field 2, the field behind "Trees" is field 3 and so forth.

Field numbers do not change even if fields are hidden using the `*CONTROL` command. For example, field 6 remains field linked with "Rake price" regardless of using `*CONTROL`.

A reference to a field that does not exist in the `*FORM` question will always receive to the value of the first field in the question, regardless of whether that field was shown. For example, `Q10F10` or `Q10F15` would both refer to `Q10F1`.

Fields can also be referenced to copy values into:

**Copying values into fields**

```
*COPY Q10F1 [50]
*COPY Q10F2 [1.25]
```

Field indexes may be referenced using a variable or an expression. The following is valid syntax to add all price values:

**Referencing fields using the repeat number**

```
*VARS totalprice

*put totalprice [0]
*REPEAT 5
*put totalprice [totalprice+Q10F(2*?R)]
*ENDREP
```

However, variables or expressions may not be used to store values *into* the fields. The following syntax is invalid:

**Invalid field reference syntax**

```
*REPEAT 8
*COPY Q10F(?R) [0]
*ENDREP
```

## 3.7 Custom Properties

### 3.7.1 Overview

Custom properties are useful way of defining and storing user specific information at question/categories/list level. This feature has high potential and use in automation where information/resources are picked up based on request/user input from global setup/repository. They are also useful in driving variable content in the script.

### 3.7.2 Custom Properties in Nfield

NIPO ODIN uses the [\\*PROPERTIES](#) command to attach properties to questions and categories, and a property function to query properties. These properties are intended for segmentation and querying and filtering on questionnaire properties in the reporting phase. The categories' properties can also be used for filtering in the questionnaire script with the property function in expressions and as an additional argument with [\\*USELIST](#).

Custom properties, in Nfield, are little different than the way they are used in Dimensions. Nfield allows little freedom in terms of where & how you can use them however that doesn't limit the alternate ways of using it.

### 3.7.3 `property()` Function

A built-in function has to be called for fetching values of custom properties. This function needs few compulsory arguments. Below describes how it should be used in ODIN script –

```
?property(Q[question number],[category number],"key")
```

`property()` is a function to fetch the custom prop values. It cannot be used without passing arguments.

`Q[question number]` is a question from which custom prop value it to be fetched.

`[category number]` is a category name (response index) which will be fetched from the above-mentioned question. This is numeric index passed with categories. The category number can also be an expression.

`Key` is a name of custom property value of which is to be fetched in.

**Example 1**

```
*TEXTVARS moodtype
*PUT moodtype [?property(Q1, Answer, "Feeling")]
```

Since it is a function, it returns property value as text type which is to be stored into text variable before use. You can directly use the function as per your needs. The function can also be used as expression under conditional statements.

**Example 2**

```
*IF [?property(Q1, Answer, "mood") = "Bored"]
```

**Example 3**

Filtering a question based on custom property defined in other question

```
*VARS Answer
*QUESTION 1 *CODES 61L1 *PROPERTIES "segment=hotel" *SAVE Answer
How will you rate the service during your stay at hotel Marriot?
1: Very Good *PROPERTIES "emotion=Positive;mood=Happy"
2: Good *PROPERTIES "emotion=Positive;mood=Happy"
3: Average *PROPERTIES "emotion= Negative;mood=Bored"
4: Bad *PROPERTIES "emotion=Negative;mood=Bored"

*QUESTION 2 *OPEN 62L1 *IF [?property(Q1, Answer, "mood") = "Bored"]
What was that which made you feel bored?
```

Here `Question 1` is about rating the overall service at hotel Marriot. The answer (index) is saved in a variable (Answer) which was later used in `property()` to fetch the value of custom property **mood**.

Next question (`Q2`), is filtered on custom property **mood** and asked only if respondents have felt **bored** (i.e. value of custom property mood).

**Example 4**

Summarizing (netting) the responses of one question into other dummy (summary) variable at back of script.

```
*VARS Answer, propval
*QUESTION 1 *CODES 61L2 *SAVE Answer
Please enter region you live in.
1: Schleswig-Holstein *PROPERTIES "DE_Reg=1"
2: Hamburg *PROPERTIES "DE_Reg=1"
3: Niedersachsen *PROPERTIES "DE_Reg=1" 4: Bremen *PROPERTIES
"DE_Reg=1"
5: Nordrhein-Westfalen *PROPERTIES "DE_Reg=2"
6: Hessen *PROPERTIES "DE_Reg=3"
7: Rheinland-Pfalz *PROPERTIES "DE_Reg=3"
8: Saarland *PROPERTIES "DE_Reg=3" 9: Baden-Wuerttemberg
*PROPERTIES "DE_Reg=3"
10: Bayern *PROPERTIES "DE_Reg=3"
11: Berlin *PROPERTIES "DE_Reg=4"
12: Mecklenburg-Vorpommern *PROPERTIES "DE_Reg=4"
13: Sachsen-Anhalt *PROPERTIES "DE_Reg=4" 14: Brandenburg
*PROPERTIES "DE_Reg=4"
15: Thuringen *PROPERTIES "DE_Reg=4"
16: Sachsen *PROPERTIES "DE_Reg=4"

*QUESTION 2 *CODES 63L1 *DUMMY
1: North
```

```

2: West
3: South
4: East

*PUT propval [?PROPERTY(Q1, Answer, "DE_Reg")]
*COPY Q2 [propval]

*PAGE
propval = *? propval
Answer = *? Answer

```

Here `Question 1 (region)` is asked from respondent, and zone at `Question 2` is back-coded using custom properties defined at `Question 1`.

There could be many ways of implementing it using custom properties; this is just one way to fulfill the requirement of summarizing regions into zones.

#### Example 5

Use in multi-language projects the property for the data in the sample rather than the category text for quota control purposes.

```

*SAMPLEDATA sGender
*VARS ansQ102
*QUESTION 102 *CODES L1 *SAVE ansQ102
Select your gender.

1: male *PROPERTIES "gender=Male"
2: female *PROPERTIES "gender=Female"

*PUT sGender [?PROPERTY(Q102, ansQ102, "gender")]

*LANGUAGE Dutch
*QUESTION 102
Kies uw geslacht.

1: man
2: vrouw

```

### 3.7.4 In \*USELIST command

Properties can be used to filter the categories shown in questions with the `*USELIST` command. When more than one property is specified they are combined with a logical and.

#### Syntax

```
*USELIST "ListName{, [property]=[value]}{; [property]=[value]}{;...}"
```

#### Example

Display in a question only the categories available in their region.

```

*LIST "Outlets"
1:Outlet 1 *PROPERTIES "nl=Y;be=N;de=N"
2:Outlet 2 *PROPERTIES "nl=N;be=Y;de=Y"
3:Outlet 3 *PROPERTIES "nl=Y;be=Y;de=N"
4:Outlet 4 *PROPERTIES "nl=N;be=N;de=Y"
5:Outlet 5 *PROPERTIES "nl=N;be=N;de=Y"
6: Outlet 6 *PROPERTIES "nl=Y;be=Y;de=Y"

*QUESTION 101 *CODES L1
Choose your country

```

```
1:Belgium *PROPERTIES "country=be"  
2:Germany *PROPERTIES "country=de"  
3:Netherlands *PROPERTIES "country=nl"  
  
*TEXTVARS sCountry  
*PUT sCountry [?PROPERTY(Q101, Q101, "country")]  
  
*QUESTION 201 *CODES L1  
Which outlet did you visit?  
  
*USELIST "Outlets, *? sCountry=Y"
```

3.7.4.1 See Also

\*PROPERTIES..... 214  
\*USELIST .....282

## 3.8 Date Functions

### 3.8.1 Overview

Date functions are designed to make calculation of dates easier.

We accept 2 date/time formats for the date functions:

1. YYYY-MM-DD HH:MM:SS(+offset), where the offset is optional. So  
[2023-09-12 12:30:45+0400] or [2023-09-12 12:30:45]

Or

2. The native Nfield format produced by the \*DATE command when stored in a text variable:  
[2024/07/15 20:30:18 +060 2]. Date functions will ignore the day of week from this format.

### 3.8.2 Adding Seconds to a Date

`?DATETIMEADD(<datetime>, <offset>)`

The offset is offset from the <datetime> in seconds. If you want to add 10 minutes to the date, the offset needs to have a value of 600. Negative numbers are allowed to subtract data.

The result will be a new datetime value, with the same format as the input <datetime> value.

**Example:**

```
*PUT NewDate[?DATETIMEADD(OldDate, Seconds)]
```

Where NewDate will hold the OldDate plus the Seconds.

### 3.8.3 Calculating the Time Difference Between Two Dates

`?DATETIMESPAN (<datetime1>, <datetime2>)`

**Example:**

```
*PUT TimeDifference [?DATETIMESPAN(datetime1, datetime2)]
```

Where `TimeDifference` will hold the seconds between `datetime1` and `datetime2`. Both `datetime` formats need to be exactly the same, including the `offset` if included. The result will be the difference in seconds obtained by doing `<datetime1> - <datetime2>`.

### 3.8.4 Obtaining the Day Number of the Week

```
?DATETIMEDAYOFWEEK (<datetime>)
```

**Example:**

```
*PUT WeekDay [?DATETIMEDAYOFWEEK(datetime)]
```

Where `WeekDay` is a number, which maps to a day-of-the-week. 1 through 7, Monday to Sunday.

**Note:** the native `*DATE` command also gives the number of the week day, when stored as an index variable, but it counts from Sunday to Saturday.

### 3.8.5 Obtaining the Week Number

```
?DATETIMEWEEKNUMBER (<datetime>)
```

**Example:**

```
*PUT WeekNumber [?DATETIMEWEEKNUMBER(datetime)]
```

Where `Weeknumber` holds the number of the week.

`Weeknumber` is according to the ISO 8601 specification, which describes the following:

The week number can be described by counting the Thursdays: week 12 contains the 12th Thursday of the year.

The ISO week-numbering year starts at the first day (Monday) of week 01 and ends at the Sunday before the new ISO year (hence without overlap or gap). It consists of 52 or 53 full weeks. The first ISO week of a year may have up to three days that are actually in the Gregorian calendar year that is ending; if three, they are Monday, Tuesday and Wednesday. Similarly, the last ISO week of a year may have up to three days that are actually in the Gregorian calendar year that is starting; if three, they are Friday, Saturday, and Sunday. The Thursday of each ISO week is always in the Gregorian calendar year denoted by the ISO week-numbering year.

## 4. Quota

### 4.1 NIPO Academy Sessions on Quota

We strongly recommend that you watch the four NIPO Academy sessions on quota that we've created [here](#) (sessions 51 through 54) to get a thorough introduction to quotas from basics to advanced.

Session 54 is specifically devoted to quota problems we often see at NIPO helpdesk. So, if you have a problem with quotas, please first check this video to see if your problem is mentioned. If yes, follow the steps in the video to solve it, and only in case it still does not work for you contact the NIPO helpdesk with full details of what you have done already and what the outcome was.

### 4.2 What is Quota?

A quota is a subset of respondents with specific characteristics, such as a common zip code, city, gender, or age group. In Nfield Manager, we can create quota frames and specify the specific subsets of respondents we are looking for and how many of each subset we need for our project. Here is an example of a simple quota frame:

Total Target:	100
Age: Below 35	40 (min)
Age: 35 and above	40 (min)

In this case 100 respondents will be interviewed: at least 40 aged below 35, 40 older ones, and 20 that could be any age.

So, a **quota frame** is the entire quota setup for a project.

**A total Target** is the required number of interviews for the entire quota frame. In our example, 100.

**A quota variable** is the criteria we use to subdivide the quota frame for the project, in this example, Age. Quota variables are used in the ODIN script as sample variables.

**Quota levels** are the cells that make up to the quota variable. In our example, for the quota variable Age, there 2 quota levels: 'Below 35' and '35 and above'.

**Level Target** – required number of interviews for a specific quota level. Sometimes we just call the level target “quota” for that level. In our example, both quota targets are equal to 40.

The quota count for a level increases by 1 when an interview for that quota is **successfully completed**.

This is what the ODIN script for this quota would look like:

```
*SAMPLEDATA age
*QUESTION 1 *CODES 101L1
Are you above or below 35 years old?
1:Below 35
2:35 and above
```

```
*IF [Q1,1] *PUT age "below 35"
*IF [Q1,2] *PUT age "35 and above"
```

## 4.2.1 Steps to Create Quota Frame

Let us take another example, where the requirement is to interview a number of respondents in each of several different age groups, and certain number of them have to be males, and others -- females.

To create the quota for this survey, we would first need to define a quota frame for it in Nfield. This is done in 3 steps:

1. Define quota variables & levels;
2. Order variables;
3. Define targets.

The quota variables and levels that you define in Nfield Manager need to be called exactly the same as in the ODIN script (they are case sensitive). When you enter a quota variable name, the Nfield Manager automatically suggests a questionnaire variable name for you.

## 4.2.2 How to Define Quota Variables and Levels

To define the quota variables and levels:

1. In Nfield, go to the *Quota* tab of the *Setup Survey* left-side menu.
2. Click on *Add a Quota Variable* link.
3. Add the quota variable “Gender”, and then the level “Male”.
4. Click on the plus button to add another level, “Female”.
5. Repeat for the “Age”.

If you need to set up a **large quota frame**, you can create a list of quotas in Word, Excel, or the Notepad containing the variable names and level names and copy and paste directly into the quota frame.

Place each variable and level on a separate line in your source file, as shown in the example. The first item is always the *Variable name*:

### Syntax:

```
Variable name
Level value Label
Level value Label
Level value Label
```

### Example:

```
City
Hyderabad
Amsterdam
Madrid
```

Place your cursor on the *Variable name* input as shown in screenshot below, then copy the quota cells from your source file and paste them in using **CTRL-V** or **right click-Copy**:

General Questionnaire **Quota** Fieldwork Retention

VersionQ Online Under construction

Quota + Add Variable

1 Define Variables & Items 2 Order & Nest Variables 3 Define Targets

Variable name  
Place your cursor here ?

Questionnaire variable ?

Multi ☐

Level value label

+ Add Variable

Order & Nest Variables

**To copy only the levels:** Place your cursor on the *Level value label* input as shown in the below screenshot, then copy only the levels needed from your source file (excluding the variable name) and paste them:

General Questionnaire **Quota** Fieldwork Retention

VersionQ Online Under construction

Quota + Add Variable

1 Define Variables & Items 2 Order & Nest Variables 3 Define Targets

Variable name ?

Questionnaire variable ?

Multi ☐

Place your cursor here

+ Add Variable

Order & Nest Variables

If one of your quota variables allows for multiple answers (the users are allowed to select/satisfy multiple sub-quotas for this variable), please move the *Multi* toggle to “On” position (as in example below):

**Please note:** you can also perform all the steps above through the script. See section [\\*QUOTA](#).

### 4.2.3 How to Organize and Order the Quota Variables

The next step is to organize your quota variables. You can change the order of quota variables, and define if and how they interlink.

In our example we have created two quota variables “Gender” and “Age”. If these are separate quota variables, your frame structure would look like this (two variables, each with two levels):

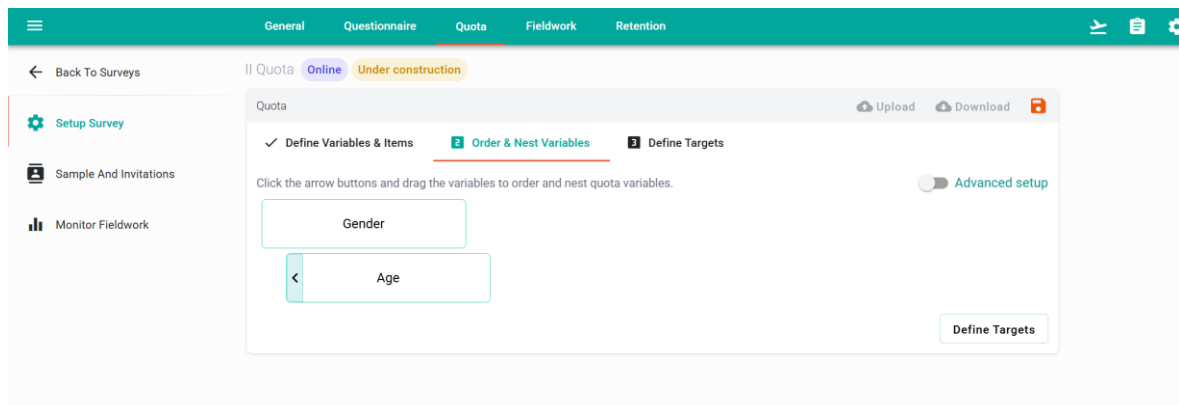
<b>Gender</b>	Male	Female
<b>Age</b>	below 35	35 and above

Any targets you would put in this quota frame would be evaluated per variable and level.

You can also make these two variables **interlinked**. Interlinked quota variables mean that “Gender” and “Age” are evaluated in combination of each other, like this:

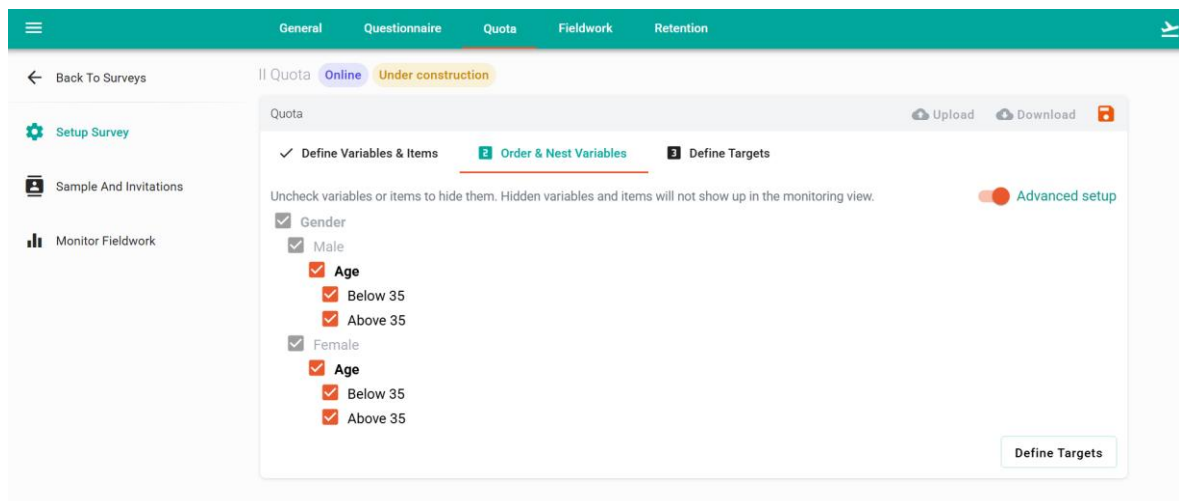
<b>Gender &amp; Age</b>		
	Male & below 35	Male & 35 and above
	Female & below 35	Female & 35 and above

In ‘*Order and nest variables*’ each quota variable (Gender/Age) is represented by a green box. You can move the boxes up or down to reorder them. Or, using the arrows on the left and right side of each box  $\diamond$  you can make the quota variables interlinked (move one under the other).



**Please note:** If you are using interlinked quotas, if your resulting quota exceeds three levels, you should really consider what you are doing. Interlinked quota variables, four levels deep, means you are using a tremendous number of sample, just to find a very small number of people.

The result is then this, when you tick “*Advanced setup*”:



If you have a quota variable that you would like to use more than once, you can simply click the **Copy** button next to that variable at the *Order & Nest Variables* step with *Advanced setup* turned off. This will create a new instance of the quota variable, which reduces manual effort.

Quota Upload Download ✓

✓ Define Variables & Items 2 Order & Nest Variables 3 Define Targets

Click the arrow buttons and drag the variables to order and nest quota variables. Advanced setup

Define Targets

#### 4.2.4 How to Add Quota Targets

The third and final step is to add targets for each quota variable and item. In this example we will assume the quotas are interlinked. Enter either only the minimum, only the maximum, or both min and max targets for each variable and item until you have entered the targets for all your quota variables.

GENERAL QUESTIONNAIRE **QUOTA** FIELDWORK ≡ 📄 ⚙️ 🔔 🎓 IRIN NIPO

✓ Define variables & items ✓ Order & Nest variables ✓ Define targets

Total target:  Limit overshoot of active interviews

**Gender**

Male	40	50
------	----	----

**Age**

below 35	set minimum	25
35 and above	25	set maximum

**Female**

	40	50
--	----	----

**Age**

below 35	set minimum	25
35 and above	25	set maximum

You will normally want to achieve your minimum targets with some oversampling to be safe.

#### 4.2.5 Max Overshoot Option

*Online surveys only.*

Nfield has a feature called Max Overshoot for Online surveys. With this feature, the system takes into account not only (successfully) completed interviews, but also the currently active interviews, when it evaluates whether from a quotas perspective a new respondent is still needed for the survey.

This feature helps against getting large overshoots in quota cells, for instance in the following scenarios:

- When a survey is started, it is immediately flooded with a large number of respondents, because the project manager wants quick results. If by chance many of these concurrent respondents fit in one specific quota cell, that cell easily gets overfilled, especially when the quota evaluation is done at the start of interview, while the filling of a cell is in Nfield only incremented at the end of an interview.
- When a survey is nearing the deadline, with some quota cells requiring just one or two more respondents, and some extra sample gets thrown at the survey. Maybe that sample has been specifically selected to fill these quota cells. If by chance the respondents all connect at once, the cells again easily get overfilled.

With max overshoot enabled, the manager elects to take into account active interviews as well as completed interviews. For example, if a quota cell requires two more respondents and max overshoot is configured at 2, then at most 4 concurrent interviews can be active on that cell (2 respondents needed + 2 max overshoot respondents), meaning overshoot for that quota cell will be limited to two at most. All the other respondents starting an interview for that quota (even before the above 4 respondents are done) will be stratted out.

To enable Max Overshoot, please set the toggle *Limit overshoot of active interviews* to On, and enter the number for the maximum overshoot allowed:

The screenshot shows the 'Quota' configuration page for a survey named 'TestMaxOvershoot'. The interface has a teal header with tabs for GENERAL, QUESTIONNAIRE, QUOTA (selected), and FIELDWORK. On the left is a sidebar with options: Setup survey, Sample and Invitations, Monitor fieldwork, and Reports. The main content area shows the quota configuration steps: Define variables & items, Order & Nest variables, and Define targets. The 'Total target' is set to 20, and 'Max overshoot' is set to 2. A toggle for 'Limit overshoot of active interviews' is turned on. Below, the quota frame is defined with 'AgeGroup' and 'Gender' as variables. The 'AgeGroup' variable has a dropdown set to '18-' with a 'set minimum' of 4. The 'Gender' variable has two options: 'Male' and 'Female', both with a 'set minimum' of 2.

**Notes:**

- Max over shoot works only on an explicit maximum target. So if you only fill in minimum targets the overshoot will have no effect.
- Max over shoots only works on the lowest target of the quota tree. So only on the lowest child cells. Not on parent cells.
- The overshoot is max overshoot per cell, not a max overshoot on the entire quota frame.

**4.2.6 Changing Quota Frames**

Once your survey is published you are can always change your targets (both total and level targets). You can also change your quota frame (add/remove/update levels and quota variables).

You can to discard any quota frame changes that have not yet been published, if, for example, you edit a quota frame but later decide not to keep the changes, or accidentally upload a quota frame meant for a different survey into a running survey by clicking on the *Discard Unpublished Quota Frame Changes* button that appears after you saved your quota frame changes:

Quota

✓ Define Variables & Items ✓ Order & Nest Variables ✓ Define Targets

**Requires publish**  
Changes to targets will not take effect until you publish the new quota frame.

Discard Unpublished Quota Frame Changes Edit Live Targets

Limit overshoot of active interviews Max overshoot: 1

Quota overview	Minimum target	Maximum target
Total target:		10
Gender		
Male2		1
Age		
Below 35		
Above 35		
Female2		

When changing the quota frame, there will be a new quota frame created, but the old one will still be there. It will just no longer be active. Each of the quota frames (the old one(s) and the active one) will have their own achievements stored, and its own targets.

There can be only 1 quota frame active: the latest published one.

You can look at all the quota frames with their corresponding achievements in *Nfield Monitor Fieldwork/Quota* tab (using the pull-down to select the version of the quota frame). In paradata you can see to which quota frame a specific interview belongs, since quota frame's ETAG is shown for each interview.

**Please note:**

- When changing quota frames you will need to adjust the targets accordingly to what was already achieved in the previous frame, and what still needs to be achieved. So, when new quota is published, the targets should be recalculated taking into account the achievements for the old quota frame(s). For example, if in the old frame the target for a cell was 10, with 6 successful interviews so far. Then the quota frame was updated, we need to make sure that the target for this cell is 4 (if we still want 10 total for that cell).
- Total Target belongs to each quota frame version (not to all of them), so each time you update the quota frame, please adjust the Total Target accordingly.
- New ETAG (an unique, random GUID number, found in paradata) for a survey is generated each time you do a Publish. Let's look at an example:
  - When you first create a quota frame, and publish it, an ETAG will be generated (let's call it ETAG1).
  - If you make a change the quota frame, and re-publish, an ETAG2 will be generated.
  - If you then make a change to other aspect of the survey (such as a script, etc.), but not the quota frame, and publish, an ETAG3 will be generated. But in the Monitor Fieldwork/Quota, Select Version you will only see quotas with ETAG1 and ETAG2. So if you

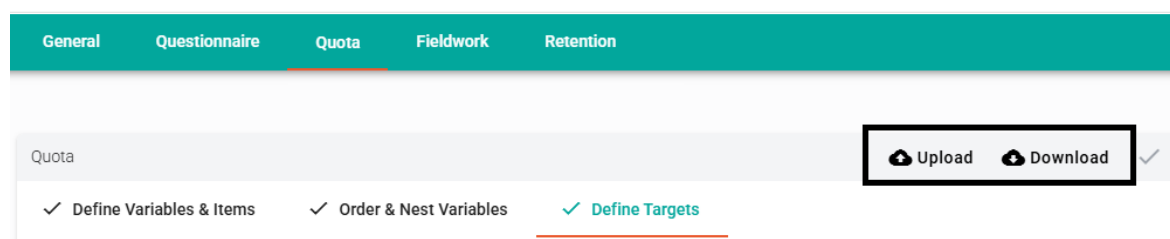
wanted to see all the interviews with quota with ETAG2, need to see all the interviews with ETAG2 and ETAG3.

- Changing quota frame during fieldwork is not desirable. Mistakes happen, so we do allow it, but it's a bad practice.

### 4.2.7 Uploading and Downloading Quota Frames

If you are setting up a survey and have a similar quota frame set up already in another survey, you can simply download it from there and then upload it into your current survey. This is especially helpful when you are working with large or complex quota frames, because it means you do not have to create them from scratch manually, saving you a lot of time and effort.

You can upload and download Quota frames directly in Nfield Manager in *Setup Survey / Quota* tab:



### 4.2.8 Quota in ODIN Script

The actual quota checks take place inside the ODIN script, using the [\\*STRAT](#) command. The respondents are usually checked early in the script to see if they fit the particular quota levels.

### 4.2.9 Quota Out in Paradata

When an interview is rejected because the quota has been fulfilled, it is reflected in the paradata. There could be multiple reasons why an interview can be rejected. The paradata will only list the first reason found. For more information, please see section [Paradata for the Quota Out](#).

## 4.3 Minimum and Maximum Targets

Minimum and maximum level targets are needed to limit the number of interviews for a quota level(s). For a quota level you can set:

- only minimum targets,
- only maximum targets,
- both minimum and maximum targets.

Here are several examples of how to add target limitations:

### 4.3.1 Only a Total Target Entered

If a quota frame has only the total target entered (for example, 10), it means that after 10 interviews, all the following interviews will not be allowed (will \*STRAT-out).

### 4.3.2 Total Target and Max Targets for Levels are Entered

Let us say we have a quota frame with the total target entered (16), and for all the quota levels only the maximum values are entered (4):

Quota	
Total target: 16	
Brand	
Brand 1	set minimum 4
Brand 2	set minimum 4
Brand 3	set minimum 4
Brand 4	set minimum 4

In this case, the interviewing will stop (\*STRAT-out) for each level when 4 interviews are reached for that level, which will get us the total of 16 interviews.

### 4.3.3 If the Total Target is Not the Same as the Sum of Maxes

Let's say we have a quota frame as below:

The screenshot shows the 'Quota' tab in a survey setup tool. The left sidebar has options: Setup survey, Sample and Invitations, Monitor fieldwork, and Reports. The main area shows progress for 'Define variables & items', 'Order & Nest variables', and 'Define targets'. The 'Total target' is set to 12. Under the 'Brand' section, there is a table with four rows, each representing a brand and its 'set minimum' target of 4.

Brand	set minimum
Brand 1	4
Brand 2	4
Brand 3	4
Brand 4	4

Here, the total target is not equal to the sum of maximum targets of levels. In this situation, the interviewing for each quota level will stop when the 4 interviews for that level are reached. But the interviewing for other levels can continue till the total of 12 interviews is reached or that level's total of 4 is reached. One or more levels in this frame will not reach their maximum target, since the total target will be reached before that. One of the levels might be left empty (if the other 3 all reach 4 first).

#### 4.3.4 Total Target and Minimum Level Targets are Entered

The screenshot shows the 'Quota' tab in a survey setup tool. The left sidebar has options: Setup survey, Sample and Invitations, Monitor fieldwork, and Reports. The main area shows progress for 'Define variables & items', 'Order & Nest variables', and 'Define targets'. The 'Total target' is set to 16. Under the 'Brand' section, there is a table with four rows, each representing a brand and its 'set maximum' target of 4.

Brand	set maximum
Brand 1	4
Brand 2	4
Brand 3	4
Brand 4	4

If only the minimum targets would be entered, and no total target, the interviewing can continue indefinitely. For every level at least 4 interviews are needed, but the maximum number per level is not set. So that is why a total target needs to be set here to limit the number of interviews. If the total is 16, then only 4 interviews for each of the 4 levels will be done.

### 4.3.5 Total Target Smaller than the Sum of Level Minimums

Quota: II: test

Quota

Define variables & items Order & Nest variables Define targets

**! Your minimum targets (totaling 16) exceed your overall target (15).** UPDATE

Total target: 15

Brand

Brand 1	4	set maximum
Brand 2	4	set maximum
Brand 3	4	set maximum
Brand 4	4	set maximum

If you try to enter the total (15) that is smaller than the sum of the levels' minimums (16), you will not be able to save the frame, until corrected.

### 4.3.6 Total Target Greater than the Sum of Level Minimums

Quota: II: test

Quota

Define variables & items Order & Nest variables Define targets

Total target: 20

Brand

Brand 1	4	set maximum
Brand 2	4	set maximum
Brand 3	4	set maximum
Brand 4	4	set maximum

If the total target is greater than the sum of the level minimums, then (in the example of the frame above), we will get at least 4 interviews in every level, and in one or more levels we will have more interviews than 4, till the total will become 20. For example, might have the following results:

*Brand 1: 6 interviews*  
*Brand 2: 4 Interviews*  
*Brand 3: 6 interviews*  
*Brand 4: 4 interviews*  
*Total: 20 interviews*

### 4.3.7 Combining Maximum and Minimum Level Targets

If we want to limit the number of interviews in a single level, but still make sure that a certain minimum is reached for that level, we need both minimum and maximum values filled:

Brand	Minimum	Maximum
Brand 1	5	10
Brand 2	5	10
Brand 3	5	10
Brand 4	5	10

In the case above every level will have at least 5, but no more than 10 interviews. The total will be 30, so the results might look something like this:

*Brand 1: 5 interviews*  
*Brand 2: 10 Interviews*  
*Brand 3: 9 interviews*  
*Brand 4: 6 interviews*  
*Total: 30 interviews*

If we did not have minimum values here, one of the brands might have not gotten any interviews at all:

*Brand 1: 10 interviews*  
*Brand 2: 0 Interviews*  
*Brand 3: 10 interviews*  
*Brand 4: 10 interviews*  
*Total: 30 interviews*

### 4.3.8 Overshoot

All the examples above are correct in a “perfect” world. Unfortunately, in the real world an overshoot of interviews can happen: more interviews done on a level than the maximum set for this level. So, for example, the maximum for the level might be 10, but the actual number of interviews that got done on this level is 12. The overshoot can happen if multiple respondents are doing an interview on that level at the same time, while the level was close to its maximum value. This can happen in large online surveys.

Let us look at what consequences the overshoot will have for the 6 cases above:

1. **Only a total target is entered:** you will end up with a few extra interviews than the total target. So, if the total target was 10, in case 3 respondents started an interview on the 10<sup>th</sup> target at close to the same time (before the first one to start is finished), you will end up with the total of 12 interviews.

2. **Total target and max targets for levels are entered:** can end up with one or more of the levels overshoot by a few interviews, and the total will be correspondingly overshoot:

	<i>Max</i>	<i>Actual</i>
<i>Brand 1:</i>	4	5
<i>Brand 2:</i>	4	4
<i>Brand 3:</i>	4	4
<i>Brand 4:</i>	4	4
<i>Total:</i>	16	17

3. **If the Total Target is Not the Same as the Sum of Maxes:** similar to the previous case, can end up with one or more levels overshoot by a few interviews, and the total will correspondingly be overshoot:

	<i>Max</i>	<i>Actual</i>
<i>Brand 1:</i>	4	5
<i>Brand 2:</i>	4	3
<i>Brand 3:</i>	4	3
<i>Brand 4:</i>	4	3
<i>Total:</i>	12	14

4. **Total target and min targets for levels are entered:** since every level has a minimum of 4 (in our example), and the total maximum is 16, we normally would just have 4 interviews for each level. If one or more of the levels overshoot the 4, the others will still be allowed to fill to the minimum of 4 interviews. **Please note that the minimums are always stronger than the maximum:** if we have a minimum number of interviews to fill for a level, it will always be allowed, even if the total (which is a maximum number) has to overshoot.

	<i>Min</i>	<i>Actual</i>
<i>Brand 1:</i>	4	5

<i>Brand 2:</i>	<i>4</i>	<i>4</i>
<i>Brand 3:</i>	<i>4</i>	<i>5</i>
<i>Brand 4:</i>	<i>4</i>	<i>4</i>
<i>Total (Max):</i>	<i>16</i>	<i>18</i>

6. **Total target greater than the sum of level minimums:** similarly to the case 4 above, even when the total is reached, if one of the levels has not yet reached the minimum target, it will be allowed to do that, even if the total has to overshoot:

	<i>Min</i>	<i>Actual</i>
<i>Brand 1:</i>	<i>4</i>	<i>6</i>
<i>Brand 2:</i>	<i>4</i>	<i>4</i>
<i>Brand 3:</i>	<i>4</i>	<i>7</i>
<i>Brand 4:</i>	<i>4</i>	<i>4</i>
<i>Total (Max):</i>	<i>20</i>	<i>21</i>

7. **Combining maximum and minimum level targets:** the levels will end up at least at the minimum and normally, less or equal than the maximum number of interviews, but it can happen that one of more levels overshoots the maximum. The total in that case might also end up overshoot if it is necessary to get to the minimum for some remaining level:

	<i>Min</i>	<i>Max</i>	<i>Actual</i>
<i>Brand 1:</i>	<i>5</i>	<i>10</i>	<i>5</i>
<i>Brand 2:</i>	<i>5</i>	<i>10</i>	<i>11</i>
<i>Brand 3:</i>	<i>5</i>	<i>10</i>	<i>10</i>
<i>Brand 4:</i>	<i>5</i>	<i>10</i>	<i>5</i>
<i>Total (Max):</i>	<i>30</i>		<i>31</i>

## 4.4 Least-filled Quota

A script should be able to select a path throughout a questionnaire based on the least-filled quota levels, or offer selections based on a list ordered by least-filled quota levels. For example, if a respondent has experience with more than one brand, we can decide to ask questions about one or more of the brands whose quota levels are least filled so that these quota levels are prioritized when trying to fulfill quota.

### Syntax

```
*GETLFLQLIST <Nrlevels><LevelName><QuotaVar>
```

### Where

- *Nrlevels* is the count of quota levels that are still open (numeric variable).
- *LevelName* is an array with the quota levels sorted from least filled to most filled.

- `QuotaVar` is the quota variable name -- the name of the `*SAMPLEDATA` variable as defined in the questionnaire. This may either be a single value sample data variable or an array. The sample table column must be associated with a quota variable.

**Note:** Command `*GETLFLQLIST` is only supported in ODIN Developer 5.18 and above.

#### Usage Notes

- `*CONTROL <ArrayVariable> <W|N>` on a `*QUESTION` supports using an array of texts. The text(s) in the array are matched against the category texts in the question, and the related codes are displayed or hidden.
- `*SAVE <ArrayVariable>` on a `*QUESTION` will take the selected categories of a question and save their labels in the indicated variable (clearing the existing contents of the variable) in the order of answers, up to a maximum as defined by the size of the array.
- `*ORDER <ArrayVariable>` on a `*QUESTION` displays the categories in the order in which they are in the array, by label.
- The `ArrayVariable` in these commands can either be a `*TEXTVARS` array or a `*SAMPLEDATA` (text) array. Matching is done case-insensitive.
- `Nrlevels` – If you have 6 levels, and 4 of them are not yet filled, `Nrlevels` will return number 4 (referring to the first 4 levels returned in the array variable).

**Important:** `*CONTROL` and `*ORDER` match array texts against the category labels of the *Default language*. Likewise, `*SAVE` stores category labels coming from the category list in the Default language. It is important that scripter and researcher ensure that category texts and sample table contents match.

#### Example 1: Least Filled Sorting

The script below returns models sorted to relative filling order (least filled first).

```
**Least filled command script
*TEMPLATE NfieldChicago

**Setup vars for least filled
*SAMPLEDATA model_nr
*TEXTVARS ModelLevel[4]
*VARS NrOfNotFilledModelLevels

*GETLFLQLIST NrOfNotFilledModelLevels ModelLevel model_nr
**Show the result of the least filled command
**Least filled are sorted on relative fillings based on the minimum quota

*PAGE
LevelsReturned: *? NrOfNotFilledModelLevels
Least filled 1: *?ModelLevel[1]
```

```

Least filled 2: *?ModelLevel[2]
Least filled 3: *?ModelLevel[3]
Least filled 4: *?ModelLevel[4]
*END

```

The following quota frame needs to be setup up for an Nfield survey to correspond with this script:

1. Create a variable `model_nr`.
2. Add 4 levels for it (*Model1-Model4*).
3. Define targets for each level.

Next, we need to:

4. Upload a script that will allow us to choose one of the quota levels. For example, this:

```

*TEMPLATE NfieldChicago

*SAMPLEDATA model_nr

*QUESTION 1 *CODES 101L1 *SAVE model_nr
Please choose a model
1:Model1
2:Model2
3:Model3
4:Model4

*END

```

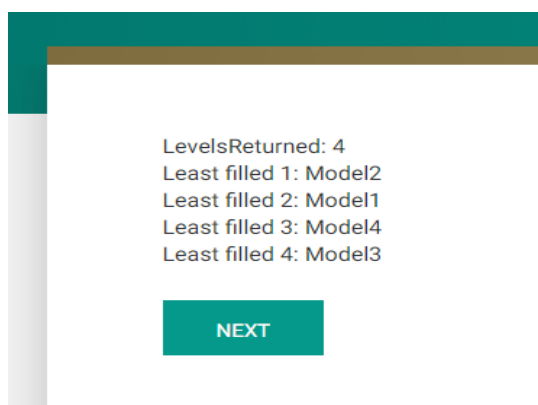
5. Publish.
6. Start Fieldwork.
7. Do several interviews selecting different model number.

We can then check in *Monitor Fieldwork/Quota* how many interviews were already done for each quota level:

Level	Minimum target	Maximum target	Successful interviews
Model1	10	20	2
Model2	20	20	2
Model3	10	20	4
Model4	20	20	7
<b>Total target:</b>		<b>60</b>	<b>15</b>

In the case above, we can see that Model2 is the least filled quota at this time (only 2 successful interviews out of the minimum target of 20, so only 10% filled); Model3 is the most filled (4 out of 10 – 40% filled).

If we now upload the Least Filled Script (above, under the Example 1), re-publish, and run the live interview, we will see the levels sorted least to most filled:



LevelsReturned: 4  
 Least filled 1: Model2  
 Least filled 2: Model1  
 Least filled 3: Model4  
 Least filled 4: Model3

**NEXT**

### Example 2: \*ORDER Command with Least Filled Quota

Upload the script below to the same survey:

```

**Least filled command with ordering
*TEMPLATE NfieldChicago

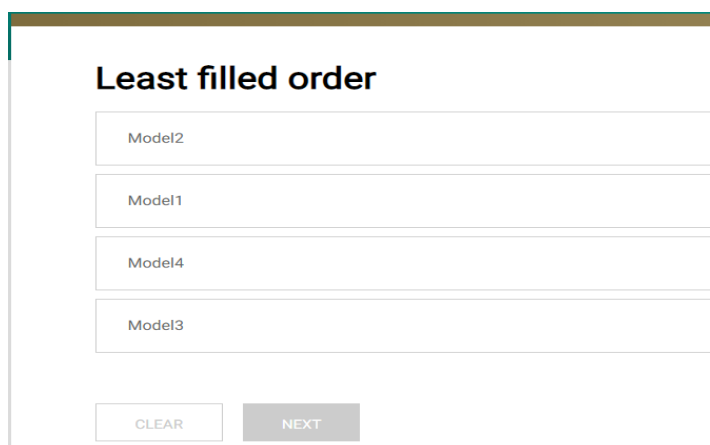
*SAMPLEDATA model_nr
*TEXTVARS ModelLevel[4]
*VARS NrofNotFilledModelLevels

*GETLFLQLIST NrofNotFilledModelLevels ModelLevel model_nr

**Show least filled order using the array returned by the *GETLSTQLIST as the argument for an *ORDER command
*QUESTION 10 *CODES 61L1 *ORDER ModelLevel
1:Model1
2:Model2
3:Model3
4:Model4

*END
  
```

Publish and run a live interview. We get the same least filled model order as in the previous example, but now in a question:



### Least filled order

Model2

Model1

Model4

Model3

**CLEAR** **NEXT**

**Example 3: \*CONTROL Command with Least Filled Quota**

\*CONTROL command can now also be based on the \*CODES label instead of only the \*CODES number. To make this work, use a text array (like the one \*GETLFQLIST returns) as an argument. Also, you can adjust the array size to only receive the X number of least filled levels. Here is an example script:

```
*TEMPLATE NfieldChicago

*SAMPLEDATA model_nr

**Use array size 1 to only receive the first of the least filled quotas
*TEXTVARS ModelLevel[1]
*VARS NOfNotFilledModelLevels

*GETLFQLIST NOfNotFilledModelLevels ModelLevel model_nr

**Use control to show only the labels returned by the *GETLFQLIST, in this case only one
*QUESTION 10 *CODES 61L1 *CONTROL ModelLevel W
1:Model1
2:Model2
3:Model3
4:Model4

*END
```

If you upload and run this script in the same survey we've been using earlier, it will only return Model2, which is the highest one in the least filled quota list:

The screenshot shows a survey interface with a teal header. Below the header, the text "Least filled order" is displayed in a large, bold, black font. Underneath this text is a rectangular box containing the text "Model2". At the bottom of the interface, there are two buttons: a light gray button labeled "CLEAR" and a darker gray button labeled "NEXT".

**Example 4: Fill the Sample Data with Least Filled Quota, Store in DAT-File**

```
*TEMPLATE NfieldChicago

**set up the vars for the least filled quotas
*SAMPLEDATA model_nr
*TEXTVARS ModelLevel[4]
```

```

*VARS NrofNotFilledModelLevels

**Get the least filled quotas for the quota model_nr
*GETLFQLIST NrofNotFilledModelLevels ModelLevel model_nr

**Show the order of the least filled, just to check
*QUESTION 10 *CODES 61L1 *ORDER ModelLevel
Least filled order
1:Model1
2:Model2
3:Model3
4:Model4

**Use the most lacking quota for this interview
*PUT model_nr [ModelLevel[1]]

** For easy data processing store it also in the Ufile

*QUESTION 11 *CODES 62L1 *DUMMY *VAR ModelUsed
ModelUsed for this interview
1:Model1
2:Model2
3:Model3
4:Model4

** Loop over the models until it matches the nr 1 returned and then store that code in the dummy Q11
*TEXTVARS BrandToCheck
*REPEAT 4
*PUT BrandToCheck Q11, ?R
*IF [BrandToCheck = ModelLevel[1]] *INCLUDE Q11 [?R] *END
*ENDREP

*END

```

### Example 5: Least Filled Quota with Mins and Maxes filled, Single Quota Variable

Let's create the following quota frame (very similar to example 1, but this time with maximum values also filled):

Quota: II:LQ Test Single Quota

Quota

Define variables & items Order & Nest variables Define targets

Total target: 18

Color

Blue	3	5
Red	3	5
Green	3	5
Yellow	3	5

Upload a script below, publish, and start fieldwork.

```

*TEMPLATE NfieldChicago

*SAMPLEDATA color

*QUESTION 1 *CODES 101L1 *SAVE color
Please choose a color
1:Blue
2:Red
3:Green
4:Yellow

```

\*END

Do several interviews selecting different colors. You can then check in the Monitor fieldwork/Quota how the fieldwork is going. Here we can see that for 2 of the levels the minimum target is met.

**Please note:**

- A checkmark next to the minimum target and the green color mean that the min target is met for the level.
- A cross next to max target and an orange color mean that the maximum target for that level is overshot.

Level	Minimum target	Maximum target	Successful interviews
Blue	✓ 3	5	5
Red	✓ 3	5	4
Green	3	5	1
Yellow	3	5	2
<b>Total target:</b>		<b>18</b>	<b>12</b>

```

**Least filled command script
*TEMPLATE NfieldChicago

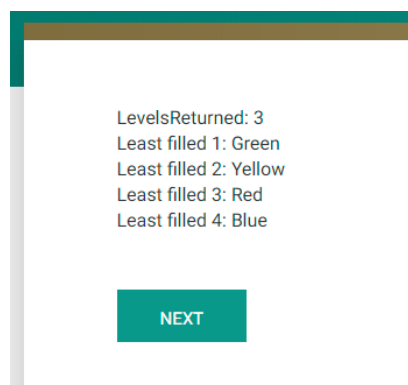
**Setup vars for least filled
*SAMPLEDATA color
*TEXTVARS Colors[4]
*VARS NrofNotFilledModelLevels

*GETLFQLIST NrofNotFilledModelLevels Colors color
**Show the result of the least filled command
**Least filled are sorted on relative fillings based on the minimum quota

*PAGE
LevelsReturned: *? NrofNotFilledModelLevels
Least filled 1: *?ModelLevel[1]
Least filled 2: *?ModelLevel[2]
Least filled 3: *?ModelLevel[3]
Least filled 4: *?ModelLevel[4]
*END

```

The result is this:



It might seem surprising that the number of not filled levels is 3, since two of the four levels have their minimum targets already met. The reason is that while least filled quota only looks at the minimum targets, the *NrOfNotFilledModelLevels* is based on what the *\*STRAT* command considers filled or not filled level target. In our example, level Red has not yet reached its maximum target (it is currently at 4 interviews, while the max target for it is 5), and the Total Target of 18 still allows it to reach the maximum target (we only have 12 interviews done, and if Red gets another interview, it will not interfere with Green or Yellow getting to their minimum targets). So, level Red is added to Yellow and Green in the *NrOfNotFilledModelLevels* to make it 3.

If the total target in this case was 15, instead of 18, it would not be possible to fill in Green and Yellow's minimum targets while still having another interview for the Red, so Red would have been excluded from the *NrOfNotFilledModelLevels*, and it would have been 2.

### Notes

Please remember that:

- Least filled quota levels are taken over minimum targets.
- Least filled is based on relative fillings.
- List returned is sorted from least filled level to most filled.
- If levels are filled equal, they are returned in the random order.
- Overshot levels are also returned, ordered from least overshot to most overshot.
- *Number of Not Filled Levels* is based on what the *\*STRAT* command considers filled or not filled level target.

## 4.5 Multi Quotas

To turn quota variable to multi-select, please turn the slide *Multi* on when creating a quota.

### Example 5: Multi Quota with Least Filled

Ask which brands are known by the respondent. Continue the interview for the 3 least known brands.

\*TEMPLATE NfieldChicago

\*\*Brands is a multi-quota so we use an array as sample data, array must be big enough to hold all the quota levels

\*SAMPLEDATA Brands[26]

\*TEXTVARS BrandName[26], LeastFilledBrand, SelectedBrand

\*VARS NOfBrands, NOfSelectedBrands, NOfKnownBrands

\*LIST "Brands"

1:BrandA

2:BrandB

3:BrandC

4:BrandD

5:BrandE

6:BrandF

7:BrandG

8:BrandH

9:BrandI

10:BrandJ

11:BrandK

12:BrandL

13:BrandM

14:BrandN

15:BrandO

16:BrandP

17:BrandQ

18:ModelR

19:ModelS

20:ModelT

21:ModelU

22:ModelV

23:ModelW

24:ModelX

25:ModelY

26:ModelZ

\*Get the brands known by the respondent

\*QUESTION 10 \*CODES 61L26 \*MULTI \*VAR "KnownBrands" \*UIOPTIONS "columns=3"

Which of these brands do you know?

\*USELIST "Brands"

\*\*if respondent does not know at least 3 brands screen him/her out

```

*COUNT NrOfKnownBrands Q10
*IF [NrOfKnownBrands < 3] *GOTO 9998

*QUESTION 20 *CODES 87L26 *MULTI *VAR "SelectedBrands" *DUMMY
Brands used for interview
*USELIST "Brands"

*GETLFQLIST NrOfBrands BrandName Brands

*PUT NrOfSelectedBrands [0]
**start repeat to loop over all returned brands in order of least to most filled
*REPEAT 26
*PUT LeastFilledBrand [BrandName[?R]]
**start repeat to check known brands against the least filled
*REPEAT 26 *CONTROL Q10 W
*PUT SelectedBrand Q10,?R
**if known brand match least filled then include this a selected brand raise the nr of brand already selected and end the loop for this
known brand
*IF [LeastFilledBrand = SelectedBrand] *INCLUDE Q20 [?R] *PUT NrOfSelectedBrands [NrOfSelectedBrands + 1] *END
*ENDREP
*IF [NrOfSelectedBrands = 3] *END **If 3 brands are selected stop looping
*ENDREP

**Fill the quota variable with the selected brands
*REPEAT 26 *Control Q20 W
*PUT Brands[?R] Q10,?R
*ENDREP

**Show the selected brands on screen
*QUESTION 999 *CODES 139L2 *CONTROL Q20 W
Selected brands
*USELIST "Brands"

*END

*Q9998
This survey is not for you

*ENDNGB

```

For this script we need to create a new survey with quota frame containing a variable `Brands`, with 26 levels in it called `BrandA-BrandZ`, and `Multi` slide turned on. Let's set only minimum targets for each level to 2.

Quota: MutiBrand

Define variables & items ✓ Order & Nest variables ✓ Define targets ✓

⚠ Quota editing has been locked because fieldwork has started. The targets are still editable.

Total target: 52

Brands	Target	Action
BrandA	2	set maximum
BrandB	2	set maximum
BrandC	2	set maximum
BrandD	2	set maximum
BrandE	2	set maximum
BrandF	2	set maximum
BrandG	2	set maximum
BrandH	2	set maximum
BrandI	2	set maximum
BrandJ	2	set maximum

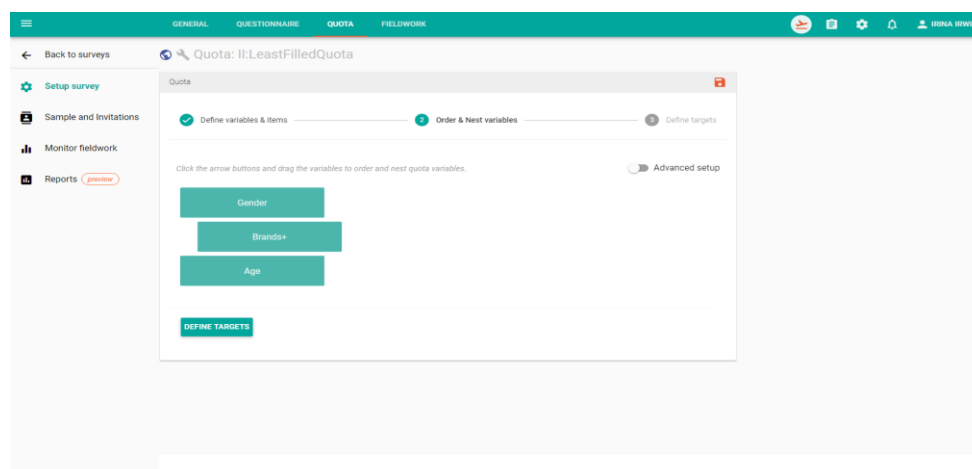
When we run the interview for this survey, we are first presented all the brands, out of which we select, let's say, 6 brands: D, I, N, P, T, X. We are then presented the 3 least selected brands (out of these 6), and since at this point all 6 are least selected, it will randomly select 3 brands from this list: let's say P, N, D.

If we run this script again, and choose the same 6 brands: D, I, N, P, T, X, we will be presented with I, T, X (not necessarily in that order), since the other 3 brands were chosen in the previous run.

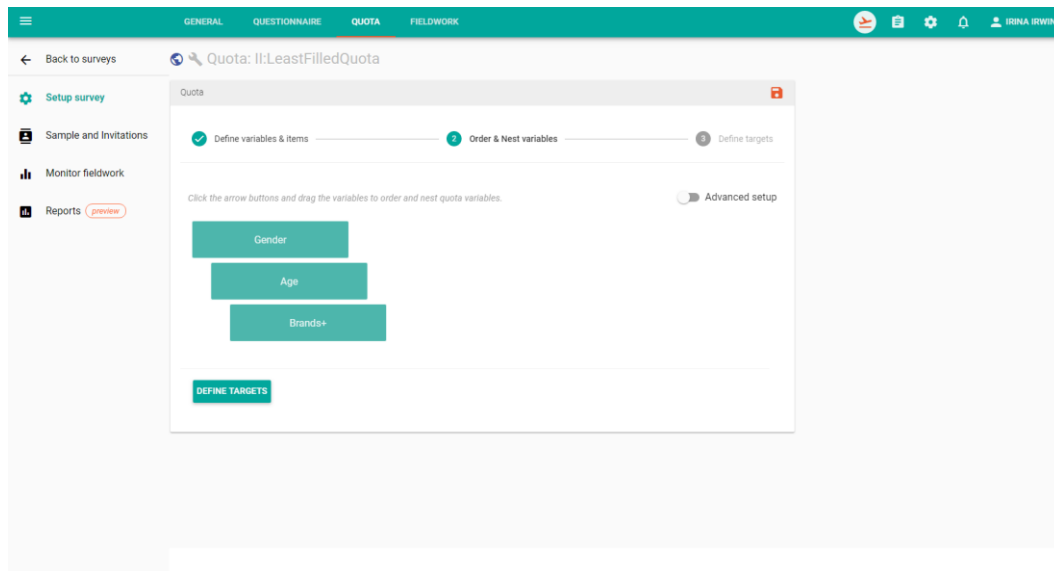
#### 4.5.1 Linking Multi Quota with Other Quotas

Multi quota variable should be at the end of the branch. If it is placed anywhere else, it is not possible to interlock it with other quota variables.

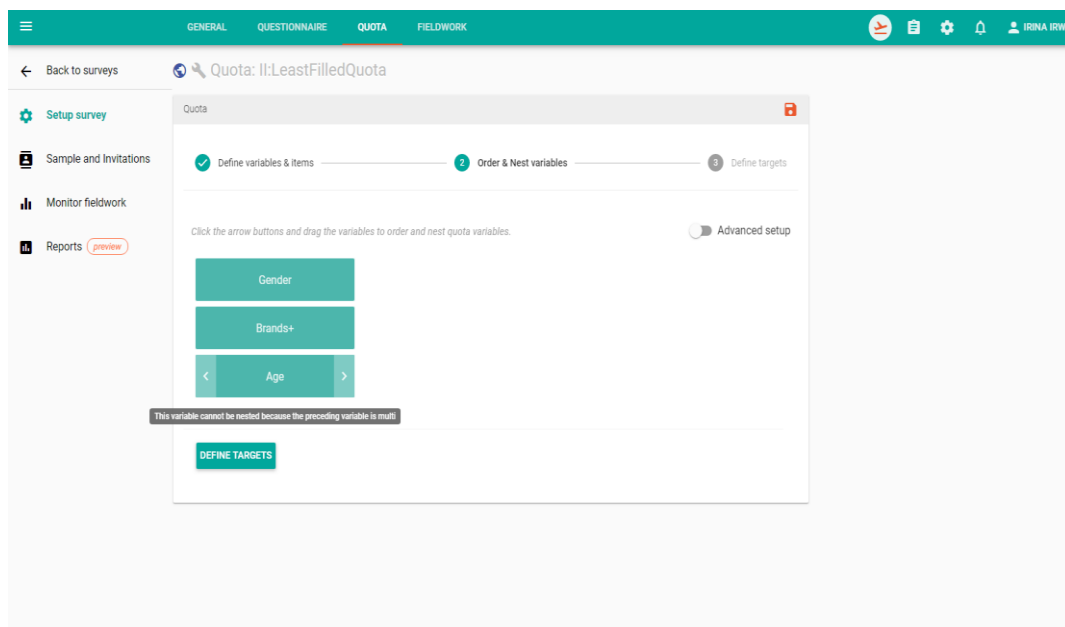
In the picture below, Brands, which is a multi-quota variable (identifiable by the plus next to its name) is linked to a quota variable Gender:



It is also possible to link all the brands, but only if the multi quota variable, Brands, is the last one in the chain, as in the picture below:



If you try to link `Brands` to, for example, `Age` while `Age` is below it, you will get an error, and will not be able to save:



## 4.6 Quota frame validation

When a quota frame is defined and target values are filled in, we want to ensure that the configured quota frame is valid, meaning the given minimum and maximum values in the frame are achievable by a set of counts that for all levels is higher than the minimum (if configured) and lower than the maximum (if configured).

In brief, this means that:

1. for each level, the minimum value (if configured) must be lower than or equal to the maximum value (if configured)
2. for each level that has nested single-code variables, the sum of the minimum values of the nested variable categories has to be lower than or equal to the level's maximum value
3. for each level that has nested single-code variables, the sum of the maximum values of the nested variable categories has to be higher than or equal to the level's minimum value
4. for each level that has nested multi-code variables, the minimum values of the nested variable categories have to be lower than or equal to the level's maximum value

If for a quota frame one of these conditions is not met for at least one level, that quota frame is not consistent, and during fieldwork no set of level fillings can be achieved that meets with the quota criteria. The user should be shown an error in this case and suggestions for improvements should be shown. For case 1, the respective level should be marked, for cases 2-5, the respective 'parent' level should be marked.

## 4.7 Counting Quota Level More than Once per Interview

For some scenarios, you might want to count quota level more than once per interview. Nfield allows to write the same value to a different index in an array of the \*SAMPLEDATA variable. This value will count as many times as it occurs for quota.

Example script:

```
*VARS numberoftrips
*SAMPLEDATA MyCountries[10]

*QUESTION 1 *NUMBER 61L2 *MAX 10
How many trips abroad did you make past year?

*PUT numberoftrips Q1

*REPEAT 10 *FIELD 63L70

*VARS iteration
*PUT iteration [?R]

*IF [iteration > numberoftrips] *END
```

```
*QUESTION 2 *CODES 1L7 *MULTI *SAVE MyCountries[?R]
```

```
Where did you go on trip #*? iteration ?
```

```
1:Netherlands
```

```
2:Spain
```

```
3:Greece
```

```
4:France
```

```
5:Portugal
```

```
6:Poland
```

```
7:India
```

```
*ENDREP
```

#### 4.7.1.1 See Also

\*CONTROL..... 116

\*GETLFQLIST..... 145

\*ORDER ..... 206

\*STRAT.....252

[Nfield Academy videos](#) 51 through 54

## 5. Suspend and Resume Interview

### 5.1 Ways to Suspend an Interview

There are 4 ways in which an interview can fall into the Suspend status:

1. For the Online interviews, if the respondent does not have any activity on an interview for 15 minutes, the interview will be suspended.
2. For the Online interviews: if the interviewers chooses the **Pause** button.
3. For the CAPI interview, if the interviewer presses the device's **Back** button, and chooses the "Save and Suspend" option.
4. For both channels, if the script ends the interview with a response code that allows a suspend (standard system code 29).
5. For CAPI interviews, you can also schedule appointments via script. To do that you need to first upload the list of custom response codes you want to use in the Nfield Manager making sure to check the option 'Appointment' for each of these codes. You are now able to use the *\*ENDST* command to get to the appointment dialog during the interview, so that an interviewer can create an appointment with the respondent.

Response Code	Description	Definite	Selectable	Appointment
18	Successful			
19	No successful, answers written			
20	System error			
21	Stratification reached			
22	*ABORT			
26	Duplicate interview			
29	Interrupted			
104	Stopped by interviewer			
105	Reset by interviewer			
201	My response code	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
202	Color is not red	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
302	code 302	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

#### Notes:

- All suspended interviews are counted as drop-outs in the **Nfield Manager/Monitor** screen.
- As soon as an interview is suspended, it is no longer an active interview. So it will no longer be counted for quota if you have turned on "Limit Overshoot of Active Interviews" (see chapter 4, Quota).

## 5.2 Resume Interview

For CAPI, the only types of surveys for which the interviews can be resumed are CAPI with Addresses. For Online, you can only resume interviews if you use response keys.

When the interview is resumed, it is restored in the exact same state as it was when it was paused. The system does it by re-running the interview, and filling it with the answers given in the previous session. There is one exception: the `*STRAT` command is always run and checked against the current quota. So, if the quota has been filled in the meantime, the resumed interview will be sent to quota full state.

So quota will be handled as is. Everything else will run as it was. That means that an interview date that was stored (using the `*DATE` command) in the previous run of the interview (the date of the 1<sup>st</sup> run), will still be the date that is stored in the resumed interview. You can modify this behavior using the `*INIT` command.

Also, any variables that you pass in the URL are always processed as was resuming, and not as is. For example, if you are starting an interview with a parameter *gender=Male*, and you are resuming the interview with *gender=Female*, the value of gender will still be *Male* for the resumed interview. This might be unwanted, especially if you are stepping out of an interview to run a 3<sup>rd</sup> party application. In that case, you would probably want the information gathered to be passed back to the Nfield interview. This is possible by using a special result code 107. So normally you would pause an interview with code `*ENDST 29` hand it over to the 3<sup>rd</sup> party application, but if you to read the (updated) parameters, you would pause with code `*ENDST 107`.

Order in which script items are executed when resuming:

- If you have an `*INIT` block, normally, it will be executed first when you run an interview. If you pause, and then resume an interview, the `*INIT` block will be executed after the script has resumed until the point where it needs input from the respondent, but before the page is rendered.
- If you use a result code 107, the URL variables will be re-read before the `*INIT` block is run.

### Notes:

- Be aware that if you resume on a different version of the script, it could impact where the interview is resumed. So, if for example, if a question was added before the question on which the interview was stopped, it will on resume go to the new question. The same will happen if the routing was changed – on resume, the new route will be used.
- If you use code 107, the interview will still be counted in the quota. If you resume this interview, it will not be checked against quota again.

## 6. Command Index

This section contains an alphabetically sorted list of NIPO ODIN script commands.

### 6.1 **\*?**

#### Purpose

Retrieves the contents of a variable or array as text.

#### Syntax

```
*? <var | var[n|expression] | QquestionNumber | QquestionId>
```

#### Description

The contents of a variable or array are retrieved and put in place of the command and will be processed as text. This command may occur at any place where text may occur.

In addition to referencing variables, **\*?** supports references to question number and question ID.

#### Arguments

**var**

This is the name of the variable from which the contents are retrieved.

**var[n|expression]**

When an array-variable is used, this the name of a variable followed by (between square brackets) a positive integer or expression that indicates the element-number of this array.

**QquestionNumber/ QquestionId**

Pipes the answer to the named question. In case of a multi-coded question, it pipes in the label of the lowest mentioned code.

#### Remarks

- You can use uppercase and lowercase names indifferently; variable names are case insensitive.
- System variables may be displayed, but do not need to be defined using **\*VARS** or **\*TEXTVARS**.

#### Example 1

```
*TEMPLATE NfieldChicago

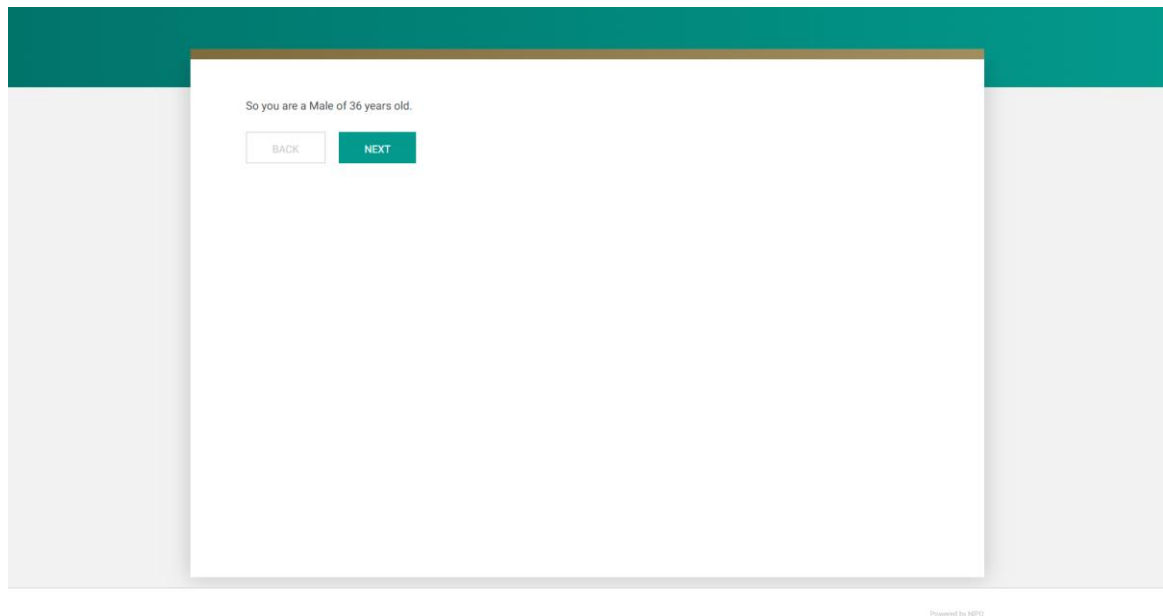
*TEXTVARS Gender, Age
*QUESTION 1 *CODES L1 *SAVE Gender
Int. type gender of respondent

1: Male
2: Female

*QUESTION 2 *NUMBER L2 *SAVE Age
How old are you?

*PAGE
So you are a *?Gender of *?Age years old.
```

## Result of Example 1



## Example 2

```
*TEMPLATE NfieldChicago

*TEXTVARS txt[4]

*PUT txt[1] "This is a text."
*PUT txt[2] "Another text."
*PUT txt[3] "Third text."
*PUT txt[4] "The fourth line contains: '*?txt[1] *?txt[2] *?txt[3]'"

*PAGE
*? txt[1]
*? txt[2]
*? txt[1] *? txt[2] *? txt[3]
*? txt[4]
```

## Result of Example 2

This is a text.

Another text.

This is a text. Another text. Third text.

The fourth line contains: 'This is a text. Another text. Third text.'

NEXT

## Example 3

```
*TEMPLATE NfieldChicago

*TEXTVARS Number, Price, PricePerFax, PricePerFaxFormatted
*FONT 0 "10 Arial"

*QUESTION 1 *FORM
Specify count and price:

1: Number of fax machines bought:      *NUMBER 61L3 *SAVE Number
2: Total price paid:    € *NUMBER 64L4.2 *SAVE Price

*PUT PricePerFax [Q1F2 / Q1F1]
**format PricePerFax to 2 spaces after the dot (cents).
*PUT PricePerFaxFormatted [?STRSUBSTR(PricePerFax,1,?STRINDEX(PricePerFax,".")+2)]

*QUESTION 2 *CODES 70L1
You bought *? Number fax machines for € *? Price?

So the average price per fax machine was € *? PricePerFax ?

1: Yes
2: No
```

## Result of Example 3

You bought 3 fax machines for € 425.50?

So the average price per fax machine was € 141.83 ?

Yes

No

BACK CLEAR NEXT

## Using a variable (\*?) to reference ids

In expressions it's possible to use a variable instead of a direct id reference. For example:

```
*PUT result [Q1, { *?id_variable}]
```

where `id_variable` is a variable containing the value of the id. In this case, if a category with the id of the value of `id_variable` is not found then the expression will evaluate to 0, the same as if the category with the id existed but was not answered.

## Example 1

```
*QUESTION 1 *NUMBER 61L1 *ID "NumberId"
How many fruits do you eat daily?
*QUESTION 2 *OPEN 62L10 *ID "FruitId"
What fruit you eat *?Q1 times daily?, in roughly 67% of the cases the interview jumps to question 100.
```

Referencing question number using \*?.

## Example 2

```
*QUESTION 1 *NUMBER 61L1 *ID "NumberId"
How many fruits do you eat daily?
*QUESTION 2 *OPEN 62L10 *ID "FruitId"
What fruit you eat *?Q{NumberId} times daily?
```

Referencing question id using \*?.

6.1.1.1 See Also

*ID.....	160
*PUT.....	217
*SAMPLEDATA .....	246
*SAVE (codes option).....	249
*SAVE (question option) .....	247
*TEXTVARS.....	268
*VARS .....	286

## 6.2 \*ALPHA

### Purpose

Defines a text question.

### Syntax

```
*ALPHA [pos]L<length>|<pos>
```

### Description

This command is always used in combination with `*QUESTION` and must be specified after `*QUESTION` on the same line. Defines a question that expects an alphanumerical answer. Contrary to an `*OPEN` question the length of an answer to an `*ALPHA` question is limited and the answer is stored in the DAT-file.

`*ALPHA` can be also used to define a text filed in `*FORM` questions.

### Arguments

`pos`

This is the start of the data field where the data is written in the DAT-file.

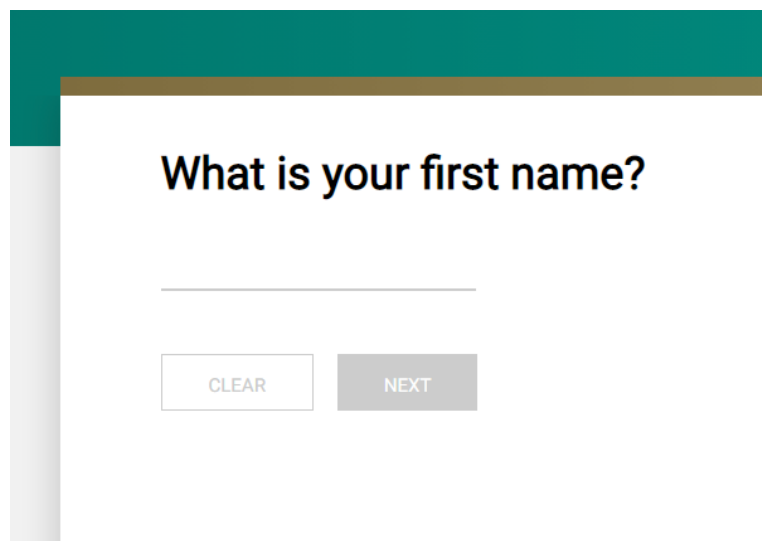
`length`

This is the length of the data field.

### Example 1

```
*TEMPLATE NfieldChicago
*QUESTION 1 *ALPHA 61L20
What is your first name?
```

### Result:



### Example 2 (in \*FORM)

```
*TEMPLATE NfieldChicago

*QUESTION 1 *FORM *BUT 99 "Refusal"
Please, fill in your name and address:

1:Name : *ALPHA 73L30

2:Street: *ALPHA 103L40

3:Place : *ALPHA 143L20
```

### Result:

Please, fill in your name and address:

Name :

Street:

Place :

### 6.2.1.1 See Also

*CODES.....	114
*LIST (definition) .....	176
*NUMBER .....	201
*OPEN (question type) .....	203
*QUESTION .....	220
*FORM .....	141

## 6.3 \*BACK

### Purpose

Jumps back to a previous question.

### Syntax

```
*BACK <n| [expression]>
```

### Description

The interview will return to the question specified in the argument. It does in fact the same as pressing the **Back** button as many times in succession as needed to reach that question.

When the \*BACK command is executed, all commands and answers between the current question and the question you are returning to are undone in the data (they are still visible on the screen as they are still in memory cache, but unless re-submitted, they will not be saved into the data file), as if you actually stepped back through the questionnaire. This is major difference with the command \*GOTO.

### Arguments

n|expression

This is a positive integer or expression that indicates an existing question.

### Remarks

With \*BACK you can only jump back to a question that was previously displayed according to the current routing. If the indicated question was never displayed, the \*BACK command is ignored.

### Example

```
*VARS tot
*VARS answer[3]
*QUESTION 1 *FORM
What proportion of travels are done by ....

(Int. don't know = '999')

1:Bus      *NUMBER L3 *RANGE [ 0 TO 100;999 ] *SAVE ANSWER[1]
2:Train    *NUMBER L3 *RANGE [ 0 TO 100;999 ] *SAVE ANSWER[2]
3:Car      *NUMBER L3 *RANGE [ 0 TO 100;999 ] *SAVE ANSWER[3]

** If one of the values is a Don't Know, skip the check
*IF [ Q1F1 = 999 \ Q1F2 = 999 \ Q1F3 = 999 ] *GOTO 3

*PUT tot [ Q1F1 + Q1F2 + Q1F3 ]
*IF [ tot > 95 & tot < 105 ] *GOTO 3

*QUESTION 2 *CODES L1
Your percentages must add up to 100%!

*? answer[1]
*? answer[2]
*? answer[3]
-----
*? Tot

1: Go back and try again *BACK 1

*QUESTION 3
```

In this example, we jump back to the first question to correct the percentages if they are not in the range 95 to 105.

6.3.1.1 See Also

\*GOTO ..... 153

\*STRAT..... 252



## 6.4 \*BLOCK

### Purpose

You can show multiple questions on one page, with optional conditions with the \*BLOCK command. This is a useful feature for displaying related questions, or to speed up interviewing, displaying more information on fewer screens. In the Example 1 below, the 2nd question will be shown on the same page, next to the 1st question, only if the answer to the 1st question is "1", otherwise it does not get shown. If we remove the condition, both questions will be displayed on the same page.

\*BLOCK opens what you wish to display on the page. The page ends with \*ENDBLOCK.

### Syntax

*\*BLOCK*

*Questions, arguments, scriptcode*

*\*ENDBLOCK*

### Description

Creates multiple questions on the same page.

### Remarks

All forms of routing are supported in \*BLOCK. Unlike in \*FORM command, you can use \*CODES on questions in \*BLOCK. You can also create \*IF and use \*CONTROL statements in \*BLOCK referring to other questions inside that \*BLOCK.

**Note:** It is not allowed to use \*BUT, \*MATRIX, \*REPEAT, \*SAVE, \*PUT or \*COPY commands in \*BLOCK.

### Example 1

```
*TEMPLATE NfieldChicago

*BLOCK
*QUESTION 1 *CODES 61L1
Q1 What car do you drive?

1: BMW
2: Audi
3: Volkswagen

*QUESTION 2 *CODES 62L1 *IF [Q1, 1]
Q2 Do you drive it often?

1: Yes
2: No

*ENDBLOCK
```

**Q1 What car do you drive?**

✓ BMW

Audi

Volkswagen

**Q2 Do you drive it often?**

Yes

No

CLEAR NEXT

Powered by NPO

## Example 2

\*TEMPLATE NfieldChicago

\*BLOCK

\*QUESTION 1 \*CODES 61L1

What is your gender?

1: Male

2: Female

9: Do not wish to answer

\*QUESTION 2 \*CODES 70L9 \*MULTI \*IF [Q1, 1,2]

Have you ever used the following shaving products?

1: Gillette Razors

2: Palmolive Shaving Cream

3: Gillette Aftershave

9: None of these \*NMUL

\*ENDBLOCK

We begin by asking about the respondent's gender. An additional question appears on the page if the respondent has provided his/her gender, asking if they have ever used the specific shaving products.

What is your gender?

✓ Male

Female

Do not wish to answer

Have you ever used the following shaving products?

Gillette Razors

Palmolive Shaving Cream

Gillette Aftershave

None of these

CLEAR

NEXT

Powered by MPD

6.4.1.1 See Also

\*MATRIX ..... 182

\*FORM ..... 141

## 6.5 \*BUT

### Purpose

Defines a button.

### Syntax

```
*BUT <code> <"text">
```

### Description

With this command you create user-defined buttons which are displayed during the interview. These buttons can be used instead of or next to answer categories. If you click on the button the relating code is written.

### Arguments

code

This is the code that is stored when pressing the button.

text

This is the button text.

### Remarks

- A maximum of five buttons per question is allowed.
- When using buttons on a \*FORM question the value of the buttons is stored in the first field of the form question.
- Allowed in \*LANGUAGE section.
- Buttons are not supported on matrixes and blocks.

### Example

```
*TEMPLATE NfieldChicago

*QUESTION 1 *CODES 61 *BUT 5 "Don't know"
How often do you read a newspaper?

1: Daily
2: Once a week
3: Once a month
4: Never
```

In this example, the category for 'Don't know' is left out of the category list. Instead a button is defined to enter this answer:

**Result:**

**How often do you read a newspaper?**

☐ Daily

☐ Once a week

☐ Once a month

☐ Never

**Example 2**

\*TEMPLATE NfieldChicago

\*QUESTION 1 \*FORM \*BUT 99 "Refusal"

Please, fill in your name and address:

1:Name : \*ALPHA 73L30

2:Street: \*ALPHA 103L40

3:Place : \*ALPHA 143L20

**Result:**

**Please, fill in your name and address:**

Name :

Street:

Place :

Powered by NPS

If the **Refusal** button is used, the code 99 is stored in the first field (*Name*).

6.5.1.1 See Also

\*USEBUTTONS.....277

## 6.6 \*CODES

### Purpose

Defines a question type.

### Syntax

```
*CODES [pos]L<length>|<pos>
```

### Description

This command is always used in combination with `*QUESTION` and must be specified after this command on the same line. It defines a closed question with a set of precoded answer categories, where each category has a unique value.

### Arguments

`pos`

This is the start of the data field where the data is written in the DAT-file.

`length`

This is the length of the data field.

In a closed, but not multiple question the number of digits of the highest category number defines the minimum length of the data field. In a closed, multiple question the highest category number (i.e. not the number of categories) defines the minimum length of the data field.

### Remarks

- It is also possible to define a dummy question by omitting the categories in order to store data from other questions or variables or to use it as a label.
- The NIPO ODIN Developer reports a warning message when the length is more than the highest code number requires. The NIPO ODIN Developer reports an error message when a `*CODES` question without `*MULTI` exceeds the maximum size of L10 (max. code number is  $2^{32}-1$ ).

### Example 1

```
*QUESTION 1 *CODES 61
```

Do you own a car?

1: Yes  
2: No

In this example, a closed question is defined with two pre-coded answer categories.

### Example 2

```
*QUESTION 2 *CODES 62L25 *MULTI
```

What brands of cars do you know?

3: Citroën  
4: Fiat  
5: Ford  
6: Hyundai  
7: Mazda  
8: Mitsubishi  
9: Nissan  
10: Opel  
11: Peugeot  
12: Renault  
13: Suzuki

14: Toyota  
 15: Volkswagen  
 16: Volvo  
  
 24: Other \*NOCON \*OPEN  
 25: Don't know

In this example a closed question with multiple answers is defined. Because the highest code-number is 25, the field has to be at least 25 positions long. When the interviewer or respondent is using the keyboard to type the code-numbers, the various codes should be separated by a space, that is generated automatically when a code number is 'complete'. So, it is best to skip code 1 when using code 10 and higher, skip code 2 when using code 20 and higher.

#### 6.6.1.1 See Also

*ALPHA.....	104
*FORM .....	141
*MULTI.....	195
*NUMBER .....	201
*OPEN (question type) .....	203

## 6.7 \*CONTROL

### Purpose

Makes the display of answer codes and labels dependent on answers to a previous question.

### Syntax

```
*CONTROL <Qn|ArrayVariable> <W|N>
```

### Description

This command is always used in combination with `*QUESTION`, `*FORM` or `*REPEAT` and has to be specified after these commands on the same line. This command controls the display of answer codes and texts depending on answers to a previous question. This so-called ‘controlling’ question is referred to by means of `Qn`.

- When used in combination with `*REPEAT`, execution of repetition numbers only takes place (or not) if the numbers correspond with the answer codes mentioned in the controlling question.
- On a `*QUESTION`, it supports using an array of texts. The text(s) and numbers in the array are matched against the category texts in the question (in default language), and the related codes are displayed or hidden.

### Arguments

`Qn`

The reference to the control question where `n` is the number of the question.

`ArrayVariable`

An array of texts.

`W`

Only answer categories that were mentioned at the control question are being displayed.

`N`

Only answer categories that were not mentioned at the control question are being displayed.

### Remarks

- Make sure that the control question and the current question have the same set of answer categories. Categories that are suppressed as a result of this command, can't be part of the answer. When all categories are suppressed the question will be skipped.
- `W` and `N` are optional because you can also use the `*CONTROL` command to export the correct variable names of a question in a `*REPEAT`.

### Example 1

```
*QUESTION 1 *CODES 61L5 *MULTI
What PC makes do you know?

1: Acer
2: IBM
3: Tulip
4: Other *OPEN
5: Don't know *NMUL
```

```
*QUESTION 2 *CODES 66L5 *MULTI *CONTROL Q1 N
```

Which of the following PC makes do you know?

```
1: Acer
2: IBM
3: Tulip
```

```
5: None of these *NMUL *NOCON
```

In this example, the answers mentioned in question 1 are not displayed in question 2. However, if *Don't know* is selected in question 1, this category is still displayed in question 2, because it is excluded from the control option by `*NOCON`.

## Example 2

`*CONTROL` command can also be based on the `*CODES` label instead of only the `*CODES` number. To make this work, use a text array (like the one `*GETLQFLIST` returns) as an argument. Also, you can adjust the array size to only receive the X number of least filled levels. Here is an example script:

```
*TEMPLATE NfieldChicago
```

```
*SAMPLEDATA ModelNr
```

```
**Use array size 1 to only receive the first of the least filled quotas
```

```
*TEXTVARS ModelLevel[1]
```

```
*VARS NrofReturnedModelLevels
```

```
*GETLQFLIST NrofReturnedModelLevels ModelLevel ModelNr
```

```
**Use control to show only the labels returned by the GETLQFLIST, in this case only one
```

```
*QUESTION 10 *CODES 61L1 *CONTROL ModelLevel W
```

```
1:Model1
2:Model2
3:Model3
4:Model4
```

```
*END
```

This script will return the highest level in the least filled quota list.

### 6.7.1.1 See Also

Least Filled Quota..... 83

\*FORM ..... 141

\*GETLQFLIST ..... 145

\*NOCON ..... 198

\*QUESTION ..... 220

\*REPEAT ... \*ENDREP ..... 230

## 6.8 \*COPY

### Purpose

Transfers data within a questionnaire.

### Syntax

```
*COPY <Qn> <Qm| [expression] | "text">
```

### Description

This command is also allowed under condition. Data specified in the second argument will be transferred to the question specified in the first argument. \*COPY is interpreting the contents of questions to be moved. The original contents of the receiving question, if any, will be replaced.

### Arguments

*Qn*

The question reference of the question where the data will be transferred to (i.e. the receiving field).

*Qm*

The question reference of the question that will be transferred (i.e. the sending field).

*expression*

If an expression is given as second argument the result of the expression will be a value. This value will then be put as data in the receiving question.

*text*

This text will be transferred literally to the receiving question. It is allowed to refer to a variable in the text. In that case, the contents of this variable will be embedded in the text and then will be transferred to the receiving question.

### Remarks

- It is recommended to use the same question type for both sending and receiving fields.
- If the data that has to be transferred consists only of digits, NIPO ODIN regards this as a number. Therefore, to transfer data from a multiple question you have to use \*INCLUDE.
- When using the \*INCLUDE command to include codes in a single-coded question, the contents will be set to the highest code. Use the \*COPY command to overwrite original contents.
- Contrary to the \*INCLUDE command the \*COPY command will give warning messages when data positions are used more than once. Use the \*INCLUDE command to suppress these warnings.
- When a question already contains the category that was copied, nothing changes.

### Example 1

```
*QUESTION 1 *CODES 71L2
*COPy Q1 [3]
```

The contents of Q1 will be 03.

### Example 2

```
*QUESTION 1 *CODES 71L2
*QUESTION 2 *CODES 81
*COPy Q1 Q2
```

The contents of Q2 are stored in Q1. For example, if Q2 contains 8, the contents of Q1 will be 08.

6.8.1.1 See Also

*EXCLUDE.....	132
*INCLUDE.....	165

## 6.9 \*COUNT

### Purpose

Counts the number of answers.

### Syntax

```
*COUNT <Qn|numvar> <Qm>
```

### Description

This command is also allowed under condition. The number of answers to the question specified in the second argument is counted and will be put as a number in the data field or variable specified in the first argument.

### Arguments

*Qn*

The question reference of the data field where the number is stored.

*numvar*

The name of the numeric variable where the number is stored. This variable has to be defined earlier in the questionnaire.

*Qm*

The question reference from which the number of answers have to be counted.

### Remarks

This command is only useful for multiple codes (*\*MULTI*) questions.

### Example

```
*QUESTION 1 *CODES 178L7 *MULTI
```

What brands of beer do you know?

```
1: Heineken
2: Amstel
3: Grolsch
4: Carlsberg
5: Tuborg
6: Other *OPEN
7: Don't know *NMUL
```

```
*VARS numb
```

```
*COUNT numb Q1
```

```
*QUESTION 2 *CODES 185 *IF [Q1,1-6]
```

So you know \*? numb brands of beer?

```
1: Yes
2: No
```

In this example, the number of answers on question 1 is counted in the variable numb and used in the question text of question 2.

**Note:** With *\*COUNT* none of the arguments can have a variable.

6.9.1.1 See Also

*ID.....	160
*MULTI.....	195
*VARS .....	286

## 6.10 \*DATE

### System

Nfield CAPI and Nfield Manager.

### Purpose

Retrieve the current date and time to be stored in a variable or (dummy) question.

### Syntax

\*DATE <Qn|var|array>

### Description

This command is also allowed under condition. This command stores a date / time stamp in a (dummy) question or a variable. In Nfield CAPI for Android, the local time of the interviewer mobile device is stored. In a web preview, the UTC date and time of the interview are stored.

### Arguments

<Qn>

A (usually dummy) question that stores the date / time stamp. This question must be an \*ALPHA type question. To store the full return value the length of the \*ALPHA field must be 25 characters. If the length is shorter, the information stored in the question is truncated.

The date is stored in the format "YYYY/MM/DD HHmm:ss +UTC W" (without quotes) where YYYY is the current year, MM is the current month, DD is the current date, HH is the current time hour in a 24 hour format, mm is the current time minutes, and ss is the current time seconds. The value +UTC marks the offset from the UTC (Coordinated Universal Time) in minutes and is always preceded by a plus or minus sign. Note that the total number of characters used for the UTC difference is therefore always 4. Last, W is the day of the week number, where Sunday is 1 and Saturday is 7.

For example, the value "2022/07/15 20:30:18 +060 2" equals to July 15th, 2022 at 20:30 and 18 seconds, 60 minutes (1 hour) later than UTC, and it is a Monday.

<var>

A variable that stores the date / time stamp.

<array>

A numerical array that stores the date / time stamp.

### Remarks

- If the variable is a numeric variable (\*VARS) only the current year is stored.
- If the variable is a text variable (\*TEXTVARS), the date / time stamp is stored in the same format as when stored in a question (see above).
- If the variable an array of numeric or text values, the components of the date / time stamp are split across the array elements, as values. To store all values an array of size 8 is required – smaller arrays would only store components up until the maximum size:
  - Array index 1 returns the year,

- index 2 -- the month,
  - index 3 – day,
  - index 4 – the hour,
  - index 5 – the minutes,
  - index 6 – the seconds,
  - index 7 – offset to UTC time zone,
  - index 8 – day of the week, where 1 is Sunday, 2 is Monday, 3 is Tuesday, 4 is Wednesday, 5 is Thursday, 6 is Friday, and 7 is Saturday.
- Please note that while Nfield CAPI uses the device's date and time, the Online surveys always use the UTC time. The offset will thus always be +000.

**Example 1**

The following example stores the date / time stamp in an \*ALPHA question:

```
*QUESTION 1 *ALPHA 61L25 *DUMMY
Current date

*DATE Q1
```

**Example 2**

This example stores the date/time stamp components in an array variable. The month and day of week names are retrieved from dummy questions to get friendly names. In the time stamp, values less than 10 get a leading zero to make sure the time is properly displayed. The UTC time stamp is formatted in hours rather than minutes and gets a leading plus sign if it is positive.

```
*VARS Date[8], UTC, DayOfWeek, Month
*TEXTVARS DayName, MonthName, sign, LeadZeroH,LeadZeroM

*Q1 *CODES L1 *DUMMY
Weekdays
1:Sunday
2:Monday
3:Tuesday
4:Wednesday
5:Thursday
6:Friday
7:Saturday

*Q2 *CODES L2 *DUMMY
Months
1:January
2:February
3:March
4:April
5:May
6:June
7:July
8:August
9:September
10:October
11:November
12:December

*DATE Date
*PUT Month [Date[2]]
*PUT DayOfWeek [Date[8]]
*PUT MonthName Q2,Month
*PUT DayName Q1,DayOfWeek
*PUT UTC [Date[7]/60]
*IF [UTC>=0] *PUT sign "+"
```

```
*IF [Date[4]<10] *PUT LeadZeroH "0"
*IF [Date[5]<10] *PUT LeadZeroM "0"

*PAGE
Start time of this interview:

*?DayName, *?MonthName *?Date[3] *?Date[1] at *?LeadZeroH*?Date[4];*?LeadZeroM*?Date[5] (UTC*?sign*?UTC)
```

6.10.1.1 Please see section below for some date calculation functions:

Date Functions ..... 65

## 6.11 \*DUMMY

### Purpose

Specifies a dummy question.

### Syntax

\*DUMMY

### Description

This command specifies a dummy question. Dummy questions are, for example, closed questions that will be used to extract category texts from. If you do not mark these questions as dummy questions they would appear on the screen during the interview. \*DUMMY prevents the question from being displayed.

### Example

```
*VARS num
*TEXTVARS txt
*QUESTION 1 *CODES L1

Gender

1: Male
2: Female

*QUESTION 2 *CODES L1 *DUMMY
1: wife
2: husband

*PUT num [Q1]
*PUT txt Q2,num
*QUESTION 3 *CODES L1

Do you have a *?txt?

1: Yes
2: No
```

In this example, question 2 will not be displayed during the interview. However, the text from this dummy question is used to display in the third question text.

## 6.12 \*END

### Purpose

Ends the questionnaire or \*REPEAT.

### Syntax

\*END

### Description

In general, this command defines the end of the questionnaire or subparts of it. This command is also allowed under condition. With subparts a physical unconditional end is always required. The action of this command depends on the context. We distinguish the following:

- Termination of the questionnaire. The interview is marked as a successful interview. All data is stored to the appropriate files. In the sample table record, if present, the response code for 'successful' is stored.
- Premature termination of a repetition block (\*REPEAT). If \*END is used within a repetition block, the repetition will be ended. The current repetition and all not yet executed repetitions, if any, will not be executed.

### Example 1

```
*QUESTION 1 *CODES 96
Do you own a DVD recorder?

1: Yes
2: No *GOTO 99

*QUESTION 2 *CODES 97
Did you use your DVD recorder yesterday?

1: Yes
2: No

*QUESTION 3 *CODES 98 *IF [ Q2,1 ]
Did you use it for playback or recording?

1: Playback
2: Recording
3: Both

*QUESTION 98
This was my final question. Thank you for your co-operation.

*END
*QUESTION 99
Then I don't have any questions for you. Thank you.

*ENDNGB
```

#### 6.12.1.1 See Also

\*ENDNGB..... 128  
 \*ENDST ..... 131  
 \*REPEAT ... \*ENDREP ..... 230



## 6.13 \*ENDNGB

### Purpose

Ends a questionnaire.

### Syntax

\*ENDNGB

### Description

Terminates the questionnaire and marks the interview as a screen-out. The data is stored in the appropriate files. In the sample table record, if present, the response code '19: No success, answers written' is stored. Interviews with this code are (by default) not counted in the stratification.

### Example

```
*QUESTION 1 *CODES 96
Do you own a VCR?

1: Yes
2: No *GOTO 99

*QUESTION 2 *CODES 97
Did you use your VCR yesterday?

1: Yes
2: No

*QUESTION 3 *CODES 98 *IF [ Q2,1 ]
Did you use it for playback or recording?

1: Playback
2: Recording
3: Both

*QUESTION 98
This was my final question. Thank you for your co-operation.

*END
*QUESTION 99

Then I don't have any questions for you. Thank you.

*ENDNGB
```

#### 6.13.1.1 See Also

\*END ..... 126

\*ENDST ..... 131

## 6.14 \*ENDPAGE

### Purpose

A custom redirect page in the questionnaire.

### Syntax

```
*ENDPAGE <responsecode>
```

```
text
```

### Description

This is a page where you can route respondents to after the survey completion status is set. There is no navigation on this page (you cannot move back, etc.). You can configure a different end page for each response code using \*ENDPAGE.

This is especially useful if you want to thank your respondents for taking part in the survey as part of the script. Before this command was introduced, this had to be done if you wanted to do it as part of the script before you executed the \*END, \*ENDST or \*ENDNGB commands. This had the risk that your respondent closed the browser before any of these commands were executed, marking the interview as a drop-out, instead of a completed interview. By moving this thank you page after these commands you can make sure they always get executed.

### Notes:

1. \*ENDPAGE without code is the end page for successful interviews. All not successful interviews require a specific \*ENDPAGE <responsecode> (per corresponding response code).
2. Respondents will still be redirected if you do not have an \*ENDPAGE or if the specific response code does not have an \*ENDPAGE. In other words, ODIN will first check if there is an \*ENDPAGE for a response code, and then, if yes, it will go to that end page. Otherwise, it will execute the redirect as configured in the Nfield Manager.
3. It does not matter where you declare the \*ENDPAGE in the script.
4. This new command has not been implemented in the ODIN Developer yet. It will only work in Nfield for now.

### Arguments

```
responsecode
```

Optional response code, the same as in the corresponding \*ENDST.

```
text
```

Text of the end page.

### Example

```
*ENDPAGE
Thank you for participation!
*ENDPAGE 21
Sorry, for this survey we need people who own bikes.*

*QUESTION 2 *CODES 62L1
Do you own a bike?

1: Yes
2: No *ENDST 21
```

For more information on the \*ENDPAGE command, please see our [NIPO Academy](#) #45.

### 6.14.1.1 See Also

*END .....	126
*ENDST .....	131
*PAGE .....	209

## 6.15 \*ENDST

### Purpose

Ends a questionnaire and writes a response code.

### Syntax

\*ENDST <code>

### Description

Terminates the questionnaire. In the sample table record, if present, the response code as indicated is stored. The writing of the data depends on the meaning of the indicated response code.

### Arguments

code

The non-response code that has to be stored in the sample table.

### Example

```
*QUESTION 1 *CODES L1
Do you own a car?

1: Yes
2: No *ENDST 16
```

In this example, where only people owning a car should be interviewed, the code for outside target group is placed in the sample table record.

#### 6.15.1.1 See Also

\*END ..... 126

\*ENDNGB..... 128

## 6.16 \*EXCLUDE

### Purpose

Data manipulation.

### Syntax

```
*EXCLUDE <Qn|array> <Qm|[expression]|[range >
```

### Description

This command is also allowed under condition. With this command it is possible to remove one or more answer codes from the receiving data field.

### Arguments

Qn

The question reference.

array

The array of questions/labels to remove.

Qm

All the answers of this question.

expression

The result of the expression will represent a code value. Next this code will be removed.

range

You may now also specify more than one code, separated by semicolons, that should be excluded from the question. Use TO to specify a range.

### Remarks

Contrary to \*COPY and \*INCLUDE, the \*EXCLUDE command is used for both single-coded and multiple questions. When a question did not contain the code that was excluded, nothing changes.

### Example 1

```
*QUESTION 1 *CODES 71L10 *MULTI
*EXCLUDE Q1 [3]
*EXCLUDE Q1 [4]
*EXCLUDE Q1 [5]
*EXCLUDE Q1 [7]
```

If Q1 contained codes 2, 3, 4, 5, 6 and 7, it would now contain only code 2 and 6.

This can also be specified as:

```
*EXCLUDE Q1 [3;4;5;7]
```

or as:

```
*EXCLUDE Q1 [3 TO 5;7]
```

### Note:

Do not separate codes with commas. Use a range with TO or use semicolons ';' to specify more than one code.

### Example 2

```
*QUESTION 1 *CODES 71L10 *MULTI
*QUESTION 2 *CODES 81L10 *MULTI
*EXCLUDE Q1 Q2
```

If Q1 contains the codes 2, 3, 4 and 7, and Q2 contains codes 2, 7, and 9, Q1 will end up with codes 3 and 4. Question 2 remains unchanged.

### Example 3

```
*QUESTION 1 *CODES 136L7 *MULTI
*EXCLUDE Q1 Q1
```

This effectively clears all answers in Q1.

#### 6.16.1.1 See Also

\*COPY ..... 118  
 \*INCLUDE ..... 165

## 6.17 \*FIELD

### Purpose

Defines positions of a subroutine, repeat block or matrix.

### Syntax

```
*FIELD [pos]L<length>|<pos>
```

### Description

This command is used with \*GOSUB (jump to subroutine), \*REPEAT (repetition block) and \*MATRIX, and must be specified after that command on the same line. This command defines data field in the DAT-file, where the entered answers of the subroutine or repetition block have to be stored.

Positions within the routine or \*REPEAT block are relative positions within this block.

### Arguments

pos

The start of the data field where the data is written in the DAT-file.

length

The length of the data field.

### Remarks

If you're working with unfixed data fields, you don't have to specify this command. Fixing the questionnaire will place the required \*FIELD commands where required.

### Example

```
*TEXTVARS BRAND

*SUBROUTINE BRANDSUB

*QUESTION 2 *CODES 1
What do you think of the quality of *?BRAND ?

1: Very good
2: Good
3: Poor
4: Very poor
5: Don't know \ no answer

*QUESTION 3 *CODES 2
What do you think of the price of *?BRAND ?

1: Very expensive
2: Expensive
3: Cheap
4: Very cheap

5: Don't know \ no answer

*ENDSUB

*QUESTION 1 *CODES 61L3 *MULTI *SAVE BRAND
Have you ever heard of the following brands of computers?

1: Acer   *GOSUB BRANDSUB *FIELD 64L2
2: IBM   *GOSUB BRANDSUB *FIELD 66L2
3: Tulip *GOSUB BRANDSUB *FIELD 68L2
```

6.17.1.1 See Also

*GOSUB.....	151
*REPEAT ... *ENDREP .....	230

## 6.18 \*FONT (definition)

### Purpose

Defines a font.

### Syntax

```
*FONT <n> <"size facename|INHERIT [style] [(R G B, R G B)]">
```

### Description

Specifies the size, font typeface name, style and foreground and background colors of a font.

Argument n is used to identify the font when the font is selected. The font description arguments must be enclosed by double quotes (") and can be specified in any order. Font 0 does not have to be specified since it is the default font. The default font is the font that is used for every question where no font is selected. If font 0 is re-defined then the new font definition is used as the default font.

### Arguments

n

Positive integer used to identify the font.

size

Point size of the font.

facename

Name of the font. This font must be installed under Windows.

INHERIT

Keyword for inheriting the font of this style sheet.

style

The following styles are available: bold, italic, underline, strikethrough. Styles can be combined, they must be separated by at least one space.

(R G B, R G B)

This argument specifies the foreground and background colors in RGB (Red-Green-Blue) values. R, G, B are in the range of 0-255 inclusive, where r is the amount of red, g is the amount of green and b is the amount of blue. If only a foreground color has to be specified this can be done by using (R G B); If only a background color has to be specified this can be done by: (,R G B).

Black is (0 0 0); white is (255 255 255); red is (255 0 0) et cetera.

### Remarks

- If a requested font is not installed, Microsoft Windows will choose a font in the same family of fonts.
- \*FONT is not recommended to use in Nfield because it always needs a weight (font's required size) defined, and that will most likely disturb the display of Nfield on many devices with different types of screen. By using \*FONT you are overruling the carefully set up fonts of the template.

### Example

```
*FONT 0 "10 Courier"
*FONT 1 "8 ARIAL"
*FONT 2 "8 ARIAL bold"
*FONT 3 "12 Times New Roman"
*FONT 4 "16 Courier (255 0 0)"
*FONT 5 "18 Roman (.255 0 0)"
*FONT 6 "25 Script (255 0 0, 0 128 255)"
*FONT 7 "10 WingDings "
*FONT 8 "10 ARIAL ITALIC"
*FONT 9 "10 ARIAL BOLD UNDERLINE"
*FONT 10 "10 ARIAL strikeout"
```

### 6.18.1.1 See Also

\*FONT (switching)..... 138

## 6.19 \*FONT (switching)

### Purpose

Switches to an earlier defined font.

### Syntax

```
*FONT <n| [expression]>
```

### Description

Selects an earlier defined font. Can be used as a question option or in question or code texts. The selected font will remain active until a new font is selected. Every time the screen is cleared, the current font is set to the default font (`*FONT 0`) or to the font specified in the question option.

### Arguments

`n|expression`

This is a positive integer or expression that refers to an earlier defined font.

### Example 1

```
*TEMPLATE NfieldChicago

*FONT 0 "18 ARIAL"
*FONT 1 "18 ARIAL BOLD"
*FONT 2 "18 ARIAL ITALIC"
*QUESTION 1 *CODES 61 *FONT 1
Do you use a personal computer?

1: Yes
2: No

*QUESTION 2 *CODES 62 *IF [Q1,1]
Do you use it for *FONT 2business*FONT 0 or *FONT 2personal*FONT 0 purposes?

1: Business
2: Personal
3: Both
```

**Result:**

Do you use it for *business* or *personal* purposes?

Business

Personal

Both

BACK CLEAR NEXT

Powered by NPD

**Example 2**

TEMPLATE NfieldChicago

\*FONT 0 "12 ARIAL (255 0 0) BOLD"

\*FONT 1 "12 ARIAL BOLD"

\*FONT 2 "1 ARIAL (0 0 255) BOLD"

\*VARS x

\*REPEAT 201 \*RANDOM

\*PUT x [?R-101]

\*IF [x<>0] \*END

\*ENDREP

\*PAGE \*FONT 1

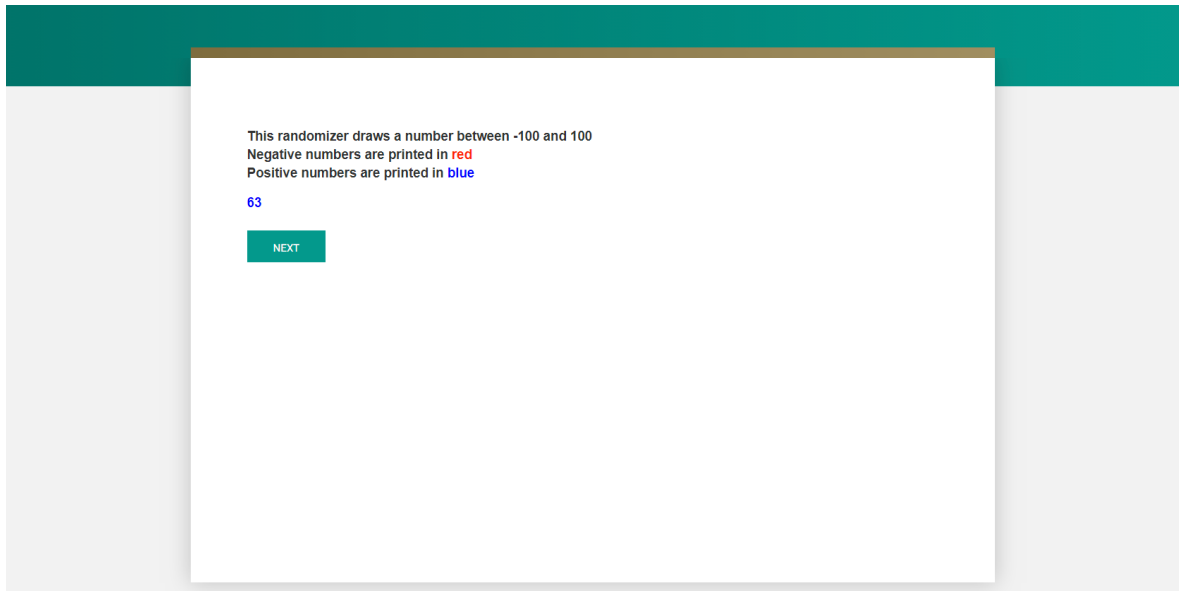
This randomizer draws a number between -100 and 100

Negative numbers are printed in \*FONT 0red\*FONT 1

Positive numbers are printed in \*FONT 2blue\*FONT 1

\*FONT [2\*(x>0)]\*? x\*FONT 1

\*END



Powered by NPGO

#### 6.19.1.1 See Also

\*FONT (definition)..... 136

## 6.20 \*FORM

### Purpose

Defines a form question.

### Syntax

\*FORM

### Description

This command is always used in combination with \*QUESTION and must be specified after this command on the same line. Defines a form question that contains one or more separate input fields that will be dealt with separately on one and the same screen. These fields will either expect numbers or alphanumerical texts of limited length as input.

The commands \*NUMBER and \*ALPHA determine of which type and how long an input level is. In the form question there may be descriptive texts that generally precede a level definition.

\*FORM questions can be controlled by another question by using the \*CONTROL command.

\*FORM questions can contain \*SCALE.

### Remarks

- If there is a code in front of the descriptive texts, this code controls in which order the levels are dealt with during the interview. The level which turn it is will be highlighted. This code is also used to determine which lines are displayed or suppressed by \*CONTROL.
- You can refer to a field in a form question with the expression Q<sub>n</sub>F<sub>m</sub>, where n is the question number and m is the field number (i.e. the order in which they were specified: from the left to the right and then from top to bottom; do not confuse this with the Form level number in front of the level, that defines the order in which fields are filled when using the <Tab>). For details, see Form Field References.

---

### Note:

If you refer to a field number that is higher than the number of fields in the form question, the value of field 1 is returned.

---

### \*NUMBER in \*FORM question

\*NUMBER can be used to define a numerical field in a \*FORM question.

### Syntax

\*NUMBER [pos]L<length>[.fraction]|<pos>

### Description

Defines a field that expects a number as an answer. Several fields may be specified in the same \*FORM question.

### Arguments

pos

The data field specification where the given answers is stored in the data file with the closed answers.

length

The length of the data field. The length of the data field defines the number of digits of the maximum value of the question. In case of a floating point value, the length of the data field is length+fraction.

fraction

The number of decimals that is allowed to be entered in a floating-point value. The fraction is stored in the data field without the decimal separator. Which separator (point or comma) is to be used by the interviewer or respondent depends on the regional configuration settings.

#### Remarks

- The answer is right-aligned and stored with leading zeros in the answer field.
- The answer can consist of an integer or a floating-point value, as defined in the data field specification.
- The decimal point in the syntax is never stored in the answer record.
- The maximum value can be set with the `*MAX` or `*RANGE` command, but is also limited by the number of positions in the data field.
- The minimum value can be set with the `*MIN` or `*RANGE` command.
- Negative values can only be entered when the `*MIN` or `*RANGE` command is defined with a negative value.
- Negative values are stored with a preceding minus sign in the data file. Note that you require an extra position in the field definition.
- Positive values are stored without sign.

#### Example 1

In this example, you can enter four values. The **Next** button is enabled after all fields are entered and match the criterion `*MAX 100`.

```
*TEMPLATE "NfieldChicago"

*UIRENDER "*NUMBER=Sumslider"

*QUESTION 120 *FORM *UIOPTIONS "instruction=What percentage of your income do you spend on ..."
Let's talk about your monthly budget

1:Rent: *NUMBER 61L4 *NON *MIN [0] *MAX [100]
2:Food: *NUMBER 65L4 *NON *MAX [100]
3:Clothing: *NUMBER 69L4 *NON *MIN [0] *MAX [100]
4:Entertainment: *NUMBER 73L4 *MIN [0] *MAX [100]
5:Total: *NUMBER 77L4 *MIN [100] *MAX [100] *UIOPTIONS "field=total"

*END
```

## Result:

**Let's talk about your monthly budget**

What percentage of your income do you spend on ...

**Rent:** 47

**Food:** 21

**Clothing:** 13

**Entertainment:** 19

**Total:** 100

Powered by NPS

## Example 2

\*TEMPLATE "NfieldChicago"

\*QUESTION 1 \*FORM \*BUT 99 "Refusal"

Please, fill in your name and address:

1:Name : \*ALPHA 73L30

2:Street: \*ALPHA 103L40

3:Place : \*ALPHA 143L20

## Result:

**Please, fill in your name and address:**

Name : \_\_\_\_\_

Street: \_\_\_\_\_

Place : \_\_\_\_\_

Powered by NPS

## Example 3

\*TEMPLATE "NfieldChicago"

\*QUESTION 10 \*FORM \*UIOPTIONS "instruction=Form with alpha and number fields. (\* marks a required field)"

Please share your personal information with us:

```
1:First name*: *ALPHA 61L25 *UIOPTIONS "placeholder=First name;characterCount=true"
2:Last name*: *ALPHA 86L25 *UIOPTIONS "placeholder=Last name;characterCount=true"
3:Age*: *NUMBER 111L3 *MIN [0] *MAX [110] *UIOPTIONS "placeholder=Age"
4:Nationality*: *ALPHA 114L50 *UIOPTIONS "placeholder=Nationality"
5:Annual income*: *NUMBER 164L6 *NON *UIOPTIONS "placeholder=Annual income"

*END
```

## Result:

**Please share your personal information with us:**

Form with alpha and number fields. (\* marks a required field)

First name\*:  First name (25)

Last name\*:  Last name (25)

Age\*:  Age

Nationality\*:  Nationality

Annual income\*:  Annual income

Powered by NPD

## 6.20.1.1 See Also

\*UIOPTIONS..... 270

## 6.21 \*GETLFQLIST

### Purpose

*Online surveys only.*

Allows script to select a path throughout a questionnaire based on the least-filled quota; offers selections based on a list ordered by least-filled quota.

For example, if a respondent has experience with more than one brand, we might want to ask questions about one or more of the brands whose quota are least filled so that these quota levels are prioritized based on that when trying to fulfill quota.

### Syntax

```
*GETLFQLIST <Nrlevels> <LevelName> <QuotaVar>
```

### Arguments

**Nrlevels**

Returns the count of quota levels that are not quota full (numeric variable).

**LevelName**

An array with the quota levels sorted from least filled to most filled.

**QuotaVar**

The quota variable name -- the name of the \*SAMPLEDATA variable as defined in the questionnaire.

This may either be a single value sample data variable or an array. The sample table column must be associated with a quota variable.

### Remark

Command \*GETLFQLIST is only supported in ODIN Developer 5.18 and above.

### Usage Notes

\*CONTROL <ArrayVariable> <W|N> on a \*QUESTION supports using an array of texts. The text(s) in the array are matched against the category texts in the question, and the related codes are displayed or hidden.

\*SAVE <ArrayVariable> on a \*QUESTION will take the selected categories of a question and save their labels in the indicated variable (clearing the existing contents of the variable) in the order of answers, up to a maximum as defined by the size of the array.

\*ORDER <ArrayVariable> on a \*QUESTION displays the categories in the order in which they are in the array, by label.

The ArrayVariable in these commands can either be a \*TEXTVARS array or a \*SAMPLEDATA (text) array. Matching is done case-insensitive.

### Notes:

- Getting the list of least filled quota through the `*GETLFQLIST` command takes active interviews into account when max overshoot is enabled. Please note that an interview only becomes "active" for a quota cell after a `*STRAT` command has been executed.
- `*CONTROL` and `*ORDER` match array texts against the category labels of the *Default language*. Likewise, `*SAVE` stores category labels coming from the category list in the Default language.  
it is important that scripter and researcher ensure that category texts and sample table contents match.

### Example 1 Least Filled Sorting

```

**Least filled command script
*TEMPLATE NfieldChicago

**Setup vars for least filled
*SAMPLEDATA ModelNr
*TEXTVARS ModelLevel[4]
*VARS NOfReturnedModelLevels

*GETLFQLIST NOfReturnedModelLevels ModelLevel ModelNr
**Show the result of the least filled command
**Least filled are sorted on relative fillings based on the minimum quota

*PAGE
LevelsReturned: *?NOfReturnedModelLevels
Least filled 1: *?ModelLevel[1]
Least filled 2: *?ModelLevel[2]
Least filled 3: *?ModelLevel[3]
Least filled 4: *?ModelLevel[4]

**pick a model and save it to the quota
*QUESTION 10 *CODES 61L1 *SAVE ModelNr ModelChosen
1:Model1
2:Model2
3:Model3
4:Model4

*END

```

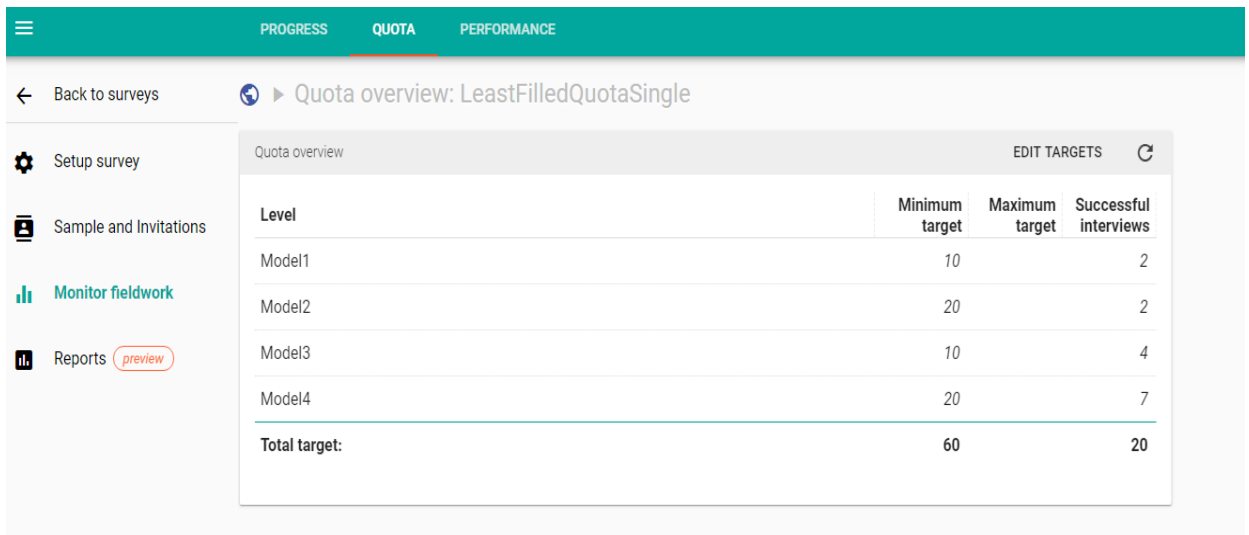
The following quota frame needs to be setup up for an Nfield survey to correspond with this script:

1. Create a variable *ModelNr*.
2. Add 4 levels for it (Model1-Model4).
3. Define targets for each level.

Next, we need to:

4. Upload a script that will allow us to choose one of the quotas.
5. Publish.
6. Start Fieldwork.
7. Do some interviews for different quotas.

We can then check in Monitor Fieldwork/Quota how many interviews were already done for each quota:

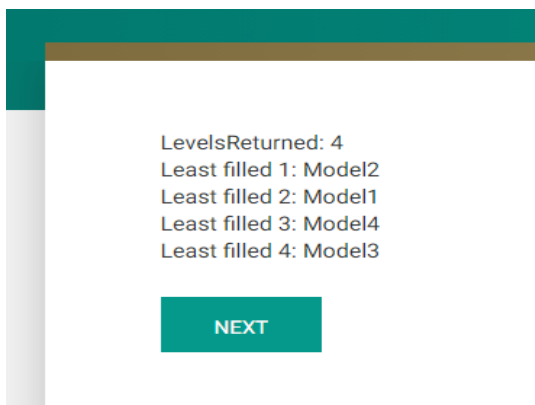


The screenshot shows the 'Quota overview' section of the Monitor Fieldwork/Quota interface. It features a table with columns for 'Level', 'Minimum target', 'Maximum target', and 'Successful interviews'. The table lists four models: Model1, Model2, Model3, and Model4, along with a 'Total target' row. The 'Successful interviews' column shows the current progress for each model.

Level	Minimum target	Maximum target	Successful interviews
Model1	10		2
Model2	20		2
Model3	10		4
Model4	20		7
<b>Total target:</b>	<b>60</b>		<b>20</b>

In the case above, we can see that Model2 is the least filled quota at this time (only 2 successful interviews out of the minimum target of 20, so only 10% filled); Model3 is the most filled (4 out of 10 – 40% filled).

If we now upload the Least Filled Script (above), re-publish, and run the live interview, we will see the levels sorted least to most filled:



#### Example 2: \*ORDER Command with Least Filled Quota

Upload the script below to the same survey:

```
**Least filled command with ordering
*TEMPLATE NfieldChicago

*SAMPLEDATA ModelNr
*TEXTVARS ModelLevel[4]
```

```

*VARS NrofReturnedModelLevels

*GETLFQLIST NrofReturnedModelLevels ModelLevel ModelNr

**Show least filled order using the array returned by the *GETLSTQLIST as the argument for an ORDER command
*QUESTION 10 *CODES 61L1 *ORDER ModelLevel
1:Model1
2:Model2
3:Model3
4:Model4

*END

```

Publish and run a live interview. We get the same least filled model order as in the previous example, but now in a question:

**Least filled order**

Model2

Model1

Model4

Model3

CLEAR NEXT

### Example 3: \*CONTROL Command with Least Filled Quota

\*CONTROL command can now also be based on the \*CODES label instead of only the \*CODES number. To make this work, use a text array (like the one \*GETLQFLIST returns) as an argument. Also, you can adjust the array size to only receive the X number of least filled levels. Here is an example script:

```

*TEMPLATE NfieldChicago

*SAMPLEDATA ModelNr

**Use array size 1 to only receive the first of the least filled quotas
*TEXTVARS ModelLevel[1]
*VARS NrofReturnedModelLevels

*GETLFQLIST NrofReturnedModelLevels ModelLevel ModelNr

**Use control to show only the labels returned by the GETLFQLIST, in this case only one
*QUESTION 10 *CODES 61L1 *CONTROL ModelLevel W
1:Model1
2:Model2
3:Model3
4:Model4

*END

```

If you upload and run this script in the same survey, we've been using earlier, it will only returns Model2, which is the highest one in the least filled quota list.

The screenshot shows a web interface with a title "Least filled order". Below the title is a text input field containing the text "Model2". Underneath the input field are two buttons: a light gray button labeled "CLEAR" and a darker gray button labeled "NEXT".

#### Example 4: Fill the Sample Data with Least Filled Quota, Store in DAT-File

```
*TEMPLATE NfieldChicago

**set up the vars for the least filled quotas
*SAMPLEDATA ModelNr
*TEXTVARS ModelLevel[4]
*VARS NrofReturnedModelLevels

**Get the least filled quotas for the quota ModelNr
*GETLQLIST NrofReturnedModelLevels ModelLevel ModelNr

**Show the order of the least filled, just to check
*QUESTION 10 *CODES 61L1 *ORDER ModelLevel
Least filled order
1:Model1
2:Model2
3:Model3
4:Model4

**Use the most lacking quota for this interview
*PUT ModelNr [ModelLevel[1]]

** For easy data processing store it also in the Ufile

*QUESTION 11 *CODES 62L1 *DUMMY *VAR ModelUsed
ModelUsed for this interview

1:Model1
2:Model2
3:Model3
4:Model4

** Loop over the models until it matches the nr 1 returned and then store that code in the dummy Q11
*TEXTVARS BrandToCheck
*REPEAT 4
*PUT BrandToCheck Q11, ?R
*IF [BrandToCheck = ModelLevel[1]] *INCLUDE Q11 [?R] *END
*ENDREP
```

\*END

## Notes

Please remember that:

- Least filled quotas are taken over minimum targets.
- Least filled is based on relative fillings.
- List returned is sorted from least filled to most filled.
- If levels are filled equal, they are returned in random order.
- Overshot levels are also returned, ordered from least overshoot to most overshoot.

### 6.21.1.1 See Also

Least Filled Quota.....	83
*CONTROL.....	116
*ORDER .....	206

## 6.22 \*GOSUB

### Purpose

Jumps to a subroutine.

### Syntax

```
*GOSUB <name>
```

### Description

Takes care of a jump to a subroutine specified by name. After return from the subroutine the interview will continue with the next command following \*GOSUB. This command is also allowed under condition.

### Arguments

name

The name of the subroutine that is called. This subroutine has to be defined earlier in the questionnaire.

### Remark

If you're working with so-called 'floating' data fields you don't have to specify the \*FIELD command. When fixing the answer fields of the questionnaire it will be put there automatically.

### Example 1

```
*TEXTVARS BRAND

*SUBROUTINE OPINION
*QUESTION 901 *CODES 1
What do you think of the service of *? BRAND?

1: Very good
2: Good
3: Poor
4: Very poor

5: Don't know \ no answer

*ENDSUB

*QUESTION 1 *CODES 61L5 *MULTI
Which of the following gas stations have you ever visited?

1: Esso *PUT BRAND "Esso" *GOSUB OPINION *FIELD 66L1
2: Shell *PUT BRAND "Shell" *GOSUB OPINION *FIELD 67L1
3: Texaco *PUT BRAND "Texaco" *GOSUB OPINION *FIELD 68L1
4: BP *PUT BRAND "BP" *GOSUB OPINION *FIELD 69L1
5: Mobil *PUT BRAND "Mobil" *GOSUB OPINION *FIELD 70L1
```

In this example, for each answer mentioned, the subroutine is called - in reverse order that they were entered in Q1. Data positions are fixed, each subroutine call requires only 1 position, specified in the \*FIELD command. Question 901 is set to position 1 in the field, a relative position within specified field.

### Example 2

```
*TEXTVARS BRAND

*SUBROUTINE OPINION
*QUESTION 1 *CODES L1
What do you think of the service of *? BRAND?

1: Very good
2: Good
```

```

3: Poor
4: Very poor

5: Don't know \ no answer

*ENDSUB

*QUESTION 2 *CODES L5 *MULTI
Which of the following gas stations have you ever visited?

1: Esso
2: Shell
3: Texaco
4: BP
5: Mobil

*IF [ Q2, 2 ] *PUT BRAND "Shell" *GOSUB OPINION
*IF [ Q2, 3 ] *PUT BRAND "Texaco" *GOSUB OPINION

```

In this example, if answer 2 or 3 would be selected, the text of the answer is stored in a variable and the subroutine is called. Data positions are not fixed, so the `*FIELD` command is not specified here.

#### 6.22.1.1 See Also

\*FIELD..... 134  
 \*SUBROUTINE ... \*ENDSUB .....260

## 6.23 \*GOTO

### Purpose

Jumps to a question.

### Syntax

```
*GOTO <n| [expression]>
```

### Description

This command can be used on separate line, after a filter expression, on a code of a \*CODES question.

If used on a category it has to be on the same line as the code number definition. If the code is selected, the interview continues with the question specified. This command can also be used unconditionally. However, jumping into or out of a subroutine or repetition block is not allowed.

### Arguments

`n|expression`

This is a positive integer or expression that indicates an existing question. Make sure that the expression always refers to an existing question number, to avoid a run-time error during the interview, that will terminate the interview in a survey error during fieldwork.

### Remark

If you jump back in your NIPO ODIN questionnaire to a question that is specified before the current position, a the message "Warning: Jump to previous question" appears during a syntax check in the NIPO ODIN Developer.

- When you jump back to correct or change answers previously entered, consider using the \*BACK command instead of \*GOTO. This shows the answers on previous questions that were entered, but will also automatically clear any answers given on a previous questionnaire routing.
- Rather than jumping to a previous question to create a loop, it is recommended to use a \*REPEAT loop to avoid creating an endless loop and / or problems when exporting questions.

This also avoids issues when an interview is suspended to be followed up later.

When you jump (unconditionally) across some questions, the syntax check in the NIPO ODIN Developer gives an overview of questions that are never used:

```
Info: Overview of unused questions:
Info:   Question <n> in line <x>
Info:   Question <m> in line <y>
```

Note that NIPO ODIN Developer might consider questions as unused, due to complex filters that always evaluate to *false*, but in fact might be *true* during an interview (for example due to expressions based on random numbers or the contents of a sample field). Consider rewriting the filter in a less complex format.

### Example 1

```
*QUESTION 10 *CODES 268
Do you have a driver's license?

1: Yes
2: No *GOTO 26
```

### Example 2

```
*QUESTION 10 *CODES 268
Do you have a driver's license?
```

```
1: Yes
2: No
```

```
*IF [Q10,2] *GOTO 26
```

In both examples, those with a driver's license continue with the first question after Q10, the others go to Q26.

**Note:** It is not possible to use a variable to refer to question ids or category ids in `*GOTO`. You can use the buffer command (`*?`) if you want to dynamically reference a question. For example:

```
*GOTO { *?<VariableContainingValidId> }
```

#### 6.23.1.1 See Also

<code>*BACK</code> .....	106
<code>*ID</code> .....	160
<code>*IF</code> .....	160

## 6.24 \*GROUP

### Purpose

Keeps categories together.

### Syntax

\*GROUP

### Description

- When displaying codes in a question randomly, rotated, you can keep codes together with the command \*GROUP after the first category of a group. All codes up to the next \*GROUP are sorted only mutually.
- Use \*GROUP on question line to randomize groups inside the question.

### Example

```
*TEMPLATE NfieldChicago
```

```
*QUESTION 202 *CODES 61L2 *RANDOM *GROUP *BUT 99 "no answer"
Question text
```

```
1: GROUP A code 1
2: GROUP A code 2
3: GROUP A code 3
4: GROUP B code 1 *GROUP
5: GROUP B code 2
6: GROUP B code 3
```

**Result (categories are randomized within each group):**

The screenshot shows a question interface with a white box titled "Question text" on a teal background. Inside the box, there are six input fields arranged in two groups. The first group, labeled "GROUP A", contains three fields with the values "GROUP A code 3", "GROUP A code 1", and "GROUP A code 2". The second group, labeled "GROUP B", contains three fields with the values "GROUP B code 2", "GROUP B code 1", and "GROUP B code 3". At the bottom of the box, there are three buttons: "CLEAR", "no answer", and "NEXT".

Powered by NFD

**Note:** \*HEADING/\*GROUP command cannot work in combination with \*ORDER command.

### 6.24.1.1 See Also

\*RANDOM..... 221

\*ROT ..... 243



## 6.25 \*HEADING

### Purpose

Defines a header text above categories.

### Syntax

```
*HEADING <text>
```

### Description

When displaying codes in a question you can define a heading above codes to be used when they are displayed randomly, rotated. The command `*HEADING` will be interpreted as an implicit `*GROUP` on the next code line.

### Arguments

`text`

A text label identifying the group of codes it precedes.

### Example

```
*TEMPLATE NfieldChicago
*FONT 0 "Arial 10"
*FONT 1 "Arial 10 Italic"

*QUESTION 201 *CODES 61L2 *RANDOM
Question text

*HEADING*FONT 1Header for code 1 to 5*FONT 0
1:code 1
2:code 2
3:code 3
4:code 4
5:code 5

*HEADING*FONT 1Header for code 6 to 10*FONT 0
6:code 6
7:code 7
8:code 8
9:code 9
10:code 10

*HEADING*FONT 1Header for code 11 to 15*FONT 0
11:code 11
12:code 12
13:code 13
14:code 14
15:code 15

98:don't know *NOCON *GROUP
99:no answer
```

## Result:

**Question text**

*Header for code 1 to 5*

*Header for code 6 to 10*

*Header for code 11 to 15*

Powered by HPS

**Notes:**

1. Please note that if there is a question with `*HEADING` command which is controlled by previous question and you have not selected any categories from the last `*HEADING` group then you will still see the `*HEADING` text appearing on the screen.
2. `*HEADING/*GROUP` command cannot work in combination with `*ORDER` command.

6.25.1.1 See Also

<code>*GROUP</code> .....	155
<code>*RANDOM</code> .....	221
<code>*ROT</code> .....	243

## 6.26 \*ID

### Purpose

Refer to a question or a code instead of a question number or a code number (respectively). Can also be used to refer to a page or matrix directly.

### Syntax

`*ID <text_id>`

### Argument

`text_id`

- If placed on a question, page or matrix, it needs to be unique within a script.
- If placed on a codes, it needs to be unique on question level.
- must be a fixed string
  - not a variable.
  - maybe optionally be surrounded with single- or double-quotes.
- case-insensitive.
- first character must be a letter, remaining characters can be alphanumeric or underscores.

### Description

Possibility to add an extra (descriptive) id to ODIN questions and codes; and then use this identifier in the script and/or reporting instead of question/code numbers.

Can be also used on `*PAGE` and `*MATRIX` to directly refer to this `*PAGE` or `*MATRIX` for ease of translation.

### Referencing IDs

The ids can be used in all the same places that question number and category code can be used.

The id is surrounded by braces (`{ }`), so `Qnumb` becomes `Q{Idname}`.

For example:

- `*GOTO {question_id}`
- `*PUT result [Q{question_id}]`
- `*INCLUDE Q2 Q{question_id}`
- `*COUNT Q{question_id} Q{another_question_id}`
- `*DATE Q{question_id}`
- `*QUESTION 1 *CODES 80L3 *MULTI *ORDER Q{question_id}`

The Q prefix is required in the same places as when using a question number, for example in `*INCLUDE`, but not for `*GOTO`.

Examples of referencing category ids:

- `*PUT result [Q1, {code_one_id}]`

- `*PUT result [Q{question_id}, {code_one_id}]`

#### Notes:

- It is not possible to use a variable to refer to question ids or category ids in other commands, such as `*GOTO`. You can use the buffer command (`*?`) if you want to dynamically reference a question. For example:  
`*GOTO { *?<VariableContainingValidId> }`
- For `*INCLUDE` the right (read) argument can have a variable reference, but not the left (destination) argument. For example, the following is valid:  
`*INCLUDE Q{M3} [Q{ *? variablewithId }]`
- With `*COUNT` none of the arguments can have a variable.
- The parser will give a warning if **some** questions in a script have an id, but **not all** questions have an id.
- Similarly, if **some** categories in a question or list have an id, but **not all** categories in the same question or list have an id.
- **No duplicate ids:** the parser will give an error if the same id is used for more than one question in a script.
- Similarly, if the same id is used for more than one category in a question or list.
- A FORM or the S argument for repeats or matrixes cannot be referenced from a `*?`.
- The ids will be present in the var file as com alongside QuestionId and CategoryId.
- ID is only fully supported from ODIN Developer version 5.18.032.
- ID is supported from DSC version 2.00.024.

#### More information

For more information, please watch the [NIPO Academy 58](#).

#### 6.26.1.1 See Also

<code>*?</code> .....	99
<code>*LANGUAGE</code> .....	169
<code>*MATRIX</code> .....	182
<code>*PAGE</code> .....	209

## 6.27 \*IF (condition), \*ELSEIF, \*ELSE, \*ENDIF

### Purpose

Creates a condition.

### Syntax

```
*IF <[expression]> <action>
<*ELSEIF <[expression]> <action>>
<*ELSE <[expression]> <action>>
<*ENDIF>
```

### Description

The expression will be interpreted and tested for true or false. If the expression is true the commands immediately following the expression will be executed. For more information on the expressions that may be used, see [Expressions](#).

### Arguments

*expression*

A boolean expression with as result true or false.

*action*

One or more commands that cause an action. Definition commands are not allowed. All these commands will be executed one after the other till the end of the command line has been reached.

You can use `*ENDIF` if you want to write your statement across multiple lines.

`*ELSEIF` terminates the `IF` block if the condition is met.

### Rules:

- The condition expression must be on the same line as the `*IF` or `*ELSEIF` commands.
- If the condition is met, the command to execute can be on multiple lines.
- `IF` block can be nested.
- Each `IF` block has to be terminated with an `*ENDIF` command.

### Example 1

```
*IF [ Q6 , 1-5 ] *GOSUB "BRAND"
```

If Q6 is answered with an answer value between 1 and 5 the subroutine *BRAND* is called.

### Example 2

```
*IF [ RAN 3 ] *GOTO 100
```

The generated values for `RAN 3` are 0, 1 and 2. Value 0 will make the expression false and thus prevent the jump. Consequently, in roughly 67% of the cases the interview jumps to question 100.

### Example 3

```
*IF [ Q11 = "PC" ] *GOTO 100
```

The condition is true if the answer to question 11 is equal to the text "PC".

**Example 4**

```

*TEXTVARS VarA, VarB, VarC, VarD

*QUESTION 1 *CODES 61L4 *MULTI
Pick a letters
1:A
2:B
3:C
4:D

*IF [Q1,1]
*PUT VarA "Check"

*ELSEIF [Q1,2]
*PUT VarB "Check"

*ELSEIF [Q1,3]
*PUT VarC "Check"

*ELSE
*PUT VarD "Check"

*ENDIF

*PAGE
VarA: ??VarA
VarB: ??VarB
VarC: ??VarC
VarD: ??VarD

```

You will see that the `IF` block terminates as soon as a condition is met, only one "letter" will ever be checked, and it will be the first one from those answered in the question.

## 6.27.1.1 See Also

\*GOTO ..... 153  
 \*IF (question option)..... 163

## 6.28 \*IF (question option)

**Purpose**

Display a question under condition.

**Syntax**

```
*IF <[n|expression]> ["text"]
```

**Description**

This command is always used in combination with `*QUESTION` and a question type definition and must be specified on the same line. This command puts a condition on a question. The question will only be displayed if the expression is true. For a detailed explanation of expressions, see Expressions (on page 48).

**Arguments**

`expression`

A boolean expression with as result *true* or *false*.

`text`

Any text. If, when creating variables during an export, the question filters are copied, this text will be copied in the variable file. In NIPO Diana, this text is placed above the table with straight runs and cross tabulations.

**Example**

```
*QUESTION 15 *CODES 96
Do you own a DVD player?
```

```
1: Yes
2: No
```

```
*QUESTION 16 *CODES 97 *IF [ Q15 , 1 ]
Did you use your DVD player yesterday?
```

```
1: Yes
2: No
```

In this example, question 16 will only be asked if the answer to question 15 is code 1.

6.28.1.1 See Also

\*IF (condition)..... 162

## 6.29 \*INCLUDE

### Purpose

Transfers data within a questionnaire.

### Syntax

```
*INCLUDE <Qn|array> <Qm| [expression] | [range]>
```

### Description

This command is also allowed under condition. One or more answer codes are combined in the receiving data field.

### Arguments

*Qn*

The question reference of the data field where the codes must be merged. For more codes to be inserted in a question, the question must be multiple coded (`*CODES *MULTI`). On single-coded questions, the existing code is overwritten with the highest code in the second parameter.

*Qm*

All codes of this question will be merged in the specified data field. This question may be a single (`*CODES`) or a multiple codes question (`*CODES *MULTI`).

*expression*

The result of the expression represents a code value. This code is marked in the data field.

*range*

Specifies a range of more than one code, separated by semicolons, that should be marked in the question. Use `TO` to specify a range.

### Remarks

- Use the `*COPY` command to overwrite original contents. Use `*EXCLUDE Qn Qn` to clear all answers from a multiple-coded question.
- Contrary to the `*COPY` command the `*INCLUDE` command does not give warning messages when data positions are used more than once, except when the `*INCLUDE` command is specified before the actual question definition.
- When copying text or an `*ALPHA` question into an `*ALPHA` question, use `*COPY` rather than `*INCLUDE`.
- When a question already contains the codes that were included, nothing changes.
- When including more than one code or including a range in a question that is not `*MULTI`, only the *last* code that was specified is included.

### Example 1

```
*QUESTION 1 *CODES 71L10 *MULTI
*INCLUDE Q1 [3]
*INCLUDE Q1 [4]
*INCLUDE Q1 [5]
*INCLUDE Q1 [7]
```

The contents of the question will be: 2, 3, 4, 5 and 7. This may also be specified as:

```
*INCLUDE Q1 [3;4;5;7]
```

Or as:

```
*INCLUDE Q1 [3 TO 5;7]
```

**Note:**

Do not separate codes with commas.

**Example 2**

```
*QUESTION 1 *CODES 71L10 *MULTI
*QUESTION 2 *CODES 121L8 *MULTI
*INCLUDE Q1 Q2
```

The answers of Q2 are merged into the answers of Q1. For example, if Q1 contained codes 1 and 4 and Q2 contains codes 1, 3 and 7, Q1 will contain codes 1, 3, 4 and 7.

**Note:** For `*INCLUDE` the right (read) argument can have a variable reference, but not the left (destination) argument. For example, the following is valid:

```
*INCLUDE Q{M3} [Q{*? variablewithId}]
```

6.29.1.1 See Also

<code>*COPY</code> .....	118
<code>*EXCLUDE</code> .....	132
<code>*ID</code> .....	160

## 6.31 \*INIT

### Purpose

**For ONLINE surveys only.**

To change the behavior of the [resumed](#) interview.

### Syntax

```
*INIT  
<ODIN code>  
*END
```

### Description

\*INIT block is run as is when the interview is [resumed](#).

### Remarks

- Only one \*INIT block per survey is allowed.
- You cannot jump out of the \*INIT block.
- You cannot jump into the \*INIT block.
- You cannot have a question with question types in the \*INIT block.
- Respondent cannot move back into the \*INIT block.
- When resuming an interview (with script that had an \*INIT block), respondent cannot move back to the session that was done before the pause.
- The \*INIT block should be defined before the 1st question in the script.

For the detailed explanation and examples of how to use the \*INIT block, please see [NIPO Academy 30](#).

## 6.32 \*LABEL

### Purpose

Defines a text label for the question's export.

### Syntax

```
*LABEL <text>
```

### Description

This command must be specified on a \*QUESTION with a question type definition. This command gives the possibility to attach an export label to a question that replaces the question text. The label doesn't have to be unique.

### Arguments

text

The text that must be used as the question label. The maximum length of a label name is 65 characters. Label names containing spaces must be enclosed by double quotes.

### Example

```
*QUESTION 1 *CODES L1 *VAR Sex *LABEL Gender
Int. type the respondents gender

1:Male
2:Female

*QUESTION 2 *NUMBER L2 *VAR Age *LABEL "Age of respondent"
Could you please tell me your age?
```

### Export result in NIPO Diana / Nvision Script

```
*Sex *SNG 61L1: Gender
1:Male
2:Female

*Age 62L2: Age of respondent
```

### Export result in IBM SPSS

```
VARIABLE LABELS
  Sex 'Gender'
  Age 'Age of respondent'
```

#### 6.32.1.1 See Also

\*VAR.....284

## 6.33 \*LANGUAGE

### Purpose

Defines a code section in a specific language.

### Syntax

```
*LANGUAGE "<name>[ , LTR|RTL]"
```

### Description

With this command, you can create one or more sections that contain text in another language than the default language. The first \*LANGUAGE section must start at the end of the script containing the actual questionnaire logic. A limited number of commands are available in a \*LANGUAGE section for mostly formatting purposes. See further below for details. All other logic must be placed in the main section.

If a question number is not available in another language, the text from the main section is displayed. If a question is only available in a translation, it is never displayed. The codes are displayed as they are specified in the language, so make sure the original code labels and their translations correspond.

During the interview the interviewer can select the appropriate language, and it is changed using the \*SWILANG command.

The name of the currently active language is contained within the system text variable `LANGUAGE`.

### Arguments

`name`

This is the name of the language section that NIPO ODIN refers to when a language is selected from the menu. Language names must be specified between double quotes (for example "Nederlands").

`LTR`

Indicates language direction Left-to-Right (default).

`RTL`

Indicates language direction Right-to-Left (Hebrew and Arabic).

### Allowed commands

The following commands may be used in a \*LANGUAGE section:

- \*QUESTION with question number but without type, position, conditions and other options.
- \*?
- \*\*
- \*BUT (see example 1)
- \*FONT (definition)
- \*FONT (switching)
- \*LIST (definition)

- \*USELIST
- \*PICT (question option)
- \*PICT (codes option)
- \*PAGE (use variables, see example 4)
- \*NUMBER (in \*FORM question) without position definition or options as place holders
- \*ALPHA (in \*FORM question) without position definition or options as place holders
- \*HEADING
- \*MERGE to merge language sections only

#### Note

If an entry you are trying to translate is missing in the \*LANGUAGE section, you can still translate it by using variables that you fill on the condition of the language (see example 3). In that example, we do this on the \*MATRIX question because the \*MATRIX question text has no entry. In the same way it can also be done for the documents in \*SHOWDOCUMENT. To translate pages and matrixes, you must give them a \*ID, so they have a hook in the \*LANGUAGE area block.

#### Example 1

```
*TEMPLATE "NfieldChicago"
*QUESTION 101 *CODES 61L1
Which language do you prefer?
Welke taal heeft uw voorkeur?

1:English *SWILANG ""
2:Nederlands *SWILANG "Dutch"

*QUESTION 111 *CODES 62L50 *MULTI *BUT 50 "None"
Which fruits do you like?

*USELIST "fruits"

*LIST "fruits"
1:Apple
2:Apricot
3:Avocado
4:Banana
6:Blackberry
7:Blackberry
8:Blackcurrant
9:Blueberry
10:Boysenberry

*LANGUAGE "Dutch"

*QUESTION 111 *BUT 50 "Geen"
Welke fruitsoorten vindt u aantrekkelijk?

*USELIST "fruits"

*LIST "fruits"
1:Appel
2:Abrikoos
3:Avocado
4:Banaan
5:Broodfruit
6:Blauwe bosbes
7:Braam
8:Zwarte bes
```

9:Zwarte bosbes  
10:Rode bosbes

## Example 2

Test questionnaire in three languages

\*TEMPLATE "NfieldChicago"

\*QUESTION 101 \*CODES L1

Which language do you prefer?

Welke taal heeft uw voorkeur?

Welche Sprache bevorzugen Sie?

1:English \*SWILANG ""

2:Nederlands \*SWILANG "Nederlands"

3:Deutsch \*SWILANG "Deutsch"

\*QUESTION 1 \*CODES L1

question in English

1: Yes

2: No

\*QUESTION 2 \*CODES L1

second question in English

1: Yes

2: No

\*QUESTION 3 \*CODES L1 \*IF [ Q101,2 ]

third question only in Dutch

1: Yes

2: No

\*END

\*LANGUAGE "Nederlands"

\*QUESTION 1

vraag in het Nederlands

1: Ja

2: Nee

\*QUESTION 2

tweede vraag in het Nederlands

1: Ja

2: Nee

\*QUESTION 3

de derde vraag is alleen in het Nederlands

1: Klopt

2: niet waar

\*LANGUAGE "Deutsch"

\*QUESTION 1

Frage auf Deutsch

1: Ja

2: Nein

\*QUESTION 2

Zweite Frage auf Deutsch

1: Ja

2: Nein

**Example 3: Mixed \*MATRIX in 2 Languages**

```

*TEMPLATE "NfieldChicago"
*TEXTVARS MatrixText, MatrixInstruction

*QUESTION 101 *CODES L1
Which language do you prefer?
Welke taal heeft uw voorkeur?

1:English *SWILANG ""
2:Nederlands *SWILANG "Nederlands"

*QUESTION 10 *CODES 61L1 *DUMMY
These are the family members that joined you on your trip to Chicago:

1:Father
2:Mother
3:Brother
4:Sister

** Put matrix text and instruction into a *DUMMY question, and then into the variables, so that these texts can be translated, since
cannot use *MATRIX in *LANGUAGE section.

*QUESTION 20 *CODES 62L1 *DUMMY
1:Please carefully check to complete all questions
2:Please enter the following details...

*PUT MatrixText Q20,2
*PUT MatrixInstruction Q20, 1

*MATRIX 4 Q10 *FIELD 63L100 *UIRENDER "Mixed" *UIOPTIONS "instruction=*>MatrixInstruction"
*>MatrixText

*QUESTION 30 *ALPHA 1L20
Name

*QUESTION 40 *NUMBER 21L3 *MIN [18] *MAX [88]
Age

*QUESTION 50 *CODES 24L1 *UIRENDER "Horizontal"
Favorite food

1:Pizza
2:Sushi
3:Hamburgers
4:Taco
5:Curry

*QUESTION 60 *CODES 25L1 *UIRENDER "Select"
Driver's license
1:Yes
2:No

*ENDMATRIX
*END

*LANGUAGE "Nederlands"

*QUESTION 10
Dit zijn de familieleden die zich bij u hebben aangesloten tijdens uw reis naar Chicago:

1:Vader
2:Moeder
3:Broer
4:Zus

*QUESTION 20
1: Controleer zorgvuldig om alle vragen te voltooien
2: Voer de volgende gegevens in ...

```

```
*QUESTION 30
Naam
```

```
*QUESTION 40
Leeftijd
```

```
*QUESTION 50
Favoriete eten
```

```
1:Pizza
2:Sushi
3:Hamburgers
4:Taco
5:Curry
```

```
*QUESTION 60
Rijbewijs?
1:Ja
2:Nee
```

```
*ENDMATRIX
```

```
*END
```

#### Example 4 \*PAGE with translation

```
*TEMPLATE "NfieldChicago"
*TEXTVARS PageText
```

```
*QUESTION 101 *CODES L1
Which language do you prefer?
Welke taal heeft uw voorkeur?
```

```
1:English *SWILANG ""
2:Nederlands *SWILANG "Nederlands"
```

**\*\* Put page text into a \*DUMMY question, and then into the variable, so that this text can be translated, since cannot use \*PAGE in \*LANGUAGE section.**

```
*QUESTION 20 *CODES 62L1 *DUMMY
1:And now some questions about snacks.
```

```
*PUT PageText Q20, 1
```

```
*PAGE
*?PageText
```

```
*QUESTION 30 *CODES 61
Did you have a snack today?
```

```
1: Yes
2: No
```

```
*LANGUAGE "Nederlands"
```

```
*QUESTION 20
1: En nu enkele vragen over snacks.
```

```
*QUESTION 30
Heb je vandaag een snack gehad?
```

```
1: Ja
2: Nee
```

#### Example 5: \*SAMPLEDATA translated

Use `PROPERTY` to save gender into sample data, rather than `*SAVE` on the question, to be independent of the language of the categories.

```
*SAMPLEDATA sGender

*QUESTION 101 *CODES L1
Select your gender.

1:male *PROPERTIES "gender=Male"
2:female *PROPERTIES "gender=Female"

*PUT sGender [?PROPERTY(Q101, ansQ101, "gender")]

*LANGUAGE Dutch
*QUESTION 101
Kies uw geslacht.

1:man
2:vrouw
```

### IDs as used in translations

IDs can also be used to translate bits of the questionnaire that are cumbersome to do now without them. The ID in the main section can be used to add a translation in the language section.

It is also supported for questions.

The following commands now can have a translation using an id:

- **Page** -- \*PAGE \*ID "pageId"
- **Matrix** -- \*MATRIX \*ID "matrixId"
- **Question** -- \*QUESTION questionNumber \*ID "idName"

### Page

The text of a \*PAGE can be translated as following:

```
*PAGE *ID "pageId"
Esta es una pagina de texto

*LANGUAGE "Netherlands"
*PAGE *ID "pageId"
Dit is een pagina met tekst
```

### Matrix

The text of a \*MATRIX (the bit before the questions) can now be translated as following:

```
*FONT 1 "10 Verdana (255 0 0)"
*VARS text

*QUESTION 10 *CODES 61L1 *DUMMY *ID "my_id"
1:One
2:Two
3:Three

*MATRIX 3 Q {my_id} *ID "mid"
Text
*QUESTION 20 *CODES 1L1 *ID "my_id2"
1:yes
2:no
3:cant remember
*ENDMATRIX

*LANGUAGE "Nederlands"

*MATRIX *ID "mid"
*FONT 1 Tekst *? text
*ENDMATRIX
```

*Please note there is no support to refer to the matrix in expressions using the {id} syntax.*

### Question

The text of a question can now also be translated using the id:

```
*QUESTION 1 *ID "qid"
This is question 1

*LANGUAGE "Nederlands"

*QUESTION *ID "qid"
Dit is vraag 1
```

### 6.33.1.1 See Also

*ID.....	160
*MATRIX .....	182
*PROPERTIES.....	214
*SHOWDOCUMENT.....	250
*SWILANG .....	262

## 6.34 \*LIST (definition) and \*ENDLIST

### Purpose

Defines an answer list.

### Syntax

```
*LIST <n| "name">
```

### Description

- Defines a code list. This command must be specified at the beginning of a line. The code list consists of all the following lines up to the first new command at the beginning of a line.
- If the list is going to be used with a text question (\*ALPHA) there may be a text between square brackets after the code text, which is used as data to store when the code is selected during the interview.
- If the list is going to be used with a numerical question (\*NUMBER) or a text question (\*ALPHA) the list may contain the same code more than once. This way you can have multiple labels be coded under the same code value, for example to cover for alternative names, changed brand names or common spelling errors.

Lists provide a great way to reuse code lists, to condense the questionnaire and to maintain code lists that are repetitively used throughout the questionnaire.

### Arguments

*n*

A positive integer and is unique within one sub-questionnaire.

*name*

The name of the list. This name must be unique within one (sub-)questionnaire.

### Remarks

- In the list definition a code may be followed by the answer option \*NOCON. The use of this answer option specifies that the specific code always will be displayed when the list is used. With this option it is for example possible to always display the answer category "Don't know".
- For each code in the code list and when the list is used in combination with a \*CODES question, the following code options are allowed: \*OPEN (codes option), \*NOCON, \*NMUL, \*FONT (switching), \*GROUP and \*SWILANG.
- When the list is used in combination with \*NUMBER or \*ALPHA, only the code option \*NOCON is allowed. The meaning of the option is that this code will not disappear from the list when typing an answer. All other code options result in a syntax error.
- A blank line at the bottom of the list is considered part of the list and is displayed in the code list of the question that uses the list.

### Example

```
*LIST 1
1: Citroën
2: Fiat
3: Ford
4: Hyundai
5: Mazda
6: Mitsubishi
```

```

7: Nissan
8: Opel
9: Peugeot
10: Renault
11: Suzuki
12: Toyota
13: Volkswagen
14: Volvo
15: Other *NOCON *OPEN

16: Don't know *NOCON

```

**\*ENDLIST****Syntax**

```
*ENDLIST
```

**Description**

Can be used (optionally) to explicitly mark the end of a `*LIST` definition.

**Arguments**

None.

**Remarks**

Lists, like many other ODIN concepts, do not have a well-defined termination token. Nfield assumes an ODIN command ends where the next ODIN command starts, or at the first line feed.

For the sake of readability, commands are usually separated by empty lines in a script.

In most cases, these empty lines are ignored by Nfield. However, when referring to multiple lists in a single question, empty lines in a list definition will create an implicit heading, thereby splitting the merged list into groups. If the merged list is then sorted or randomized, the sort order will be applied within each of these groups separately, rather than on the merged list as a whole.

An explicit `*ENDLIST` command is an easy way to prevent these issues.

**Example**

<p>The following script :</p> <pre> *TEMPLATE NfieldChicago  *LIST "Fruits" 1:Banana 2:Dragon fruit 3:Apple 4:Cherry </pre>	<p>will generate the following result:</p>
---	--

<pre> *LIST "Veggies" 5:Broccoli 6:Aubergine 7:Carrot  *QUESTION 10 *CODES 61L7 *MULTI *SORT en-us Which fruits and veggies have you eaten in the last 7 days? *USELIST "Fruits" *USELIST "Veggies" </pre>	<p><b>Which fruits and veggies have you eaten in the last 7 days?</b></p> <div> <input type="text" value="Apple"/> </div> <div> <input type="text" value="Banana"/> </div> <div> <input type="text" value="Cherry"/> </div> <div> <input type="text" value="Dragon fruit"/> </div> <div> <input type="text" value="Aubergine"/> </div> <div> <input type="text" value="Broccoli"/> </div> <div> <input type="text" value="Carrot"/> </div> <div> <input type="button" value="CLEAR"/> <input type="button" value="NEXT"/> </div>
<p>Inserting *ENDLIST into the script:</p> <pre> *TEMPLATE NfieldChicago  *LIST "Fruits" 1:Banana 2:Dragon fruit 3:Apple 4:Cherry *ENDLIST  *LIST "Veggies" 5:Broccoli 6:Aubergine 7:Carrot  *QUESTION 10 *CODES 61L7 *MULTI *SORT en-us Which fruits and veggies have you eaten in the last 7 days? *USELIST "Fruits" *USELIST "Veggies" </pre>	<p>will result in the following:</p> <p><b>Which fruits and veggies have you eaten in the last 7 days?</b></p> <div> <input type="text" value="Apple"/> </div> <div> <input type="text" value="Aubergine"/> </div> <div> <input type="text" value="Banana"/> </div> <div> <input type="text" value="Broccoli"/> </div> <div> <input type="text" value="Carrot"/> </div> <div> <input type="text" value="Cherry"/> </div> <div> <input type="text" value="Dragon fruit"/> </div> <div> <input type="button" value="BACK"/> <input type="button" value="CLEAR"/> <input type="button" value="NEXT"/> </div>

## 6.34.1.1 See Also

\*LIST (question option).....179

\*USELIST .....282

## 6.35 \*LIST (question option)

### Purpose

Uses answer codes from a list.

### Syntax

```
*LIST <n| [expression] | "name">
```

### Description

This command is always used in combination with \*QUESTION and a question type definition and must be specified after these commands on the same line. An earlier defined list of codes will be used as answer codes for the question. This command may be used in the following three question types:

- In combination with a \*CODES question optionally with \*MULTI, the codes and the texts from the list are used. The answer codes stored in the DAT-file are decided by the code numbers in the list.
- In combination with a \*NUMBER question the answer must be typed. The list of available options narrows down at each character entered the number, and only code labels with a matching pattern (both labels starting with or containing the text) are displayed. When finally, one is selected, the code number belonging to the answer is stored as a number in the DAT-file. The code numbers are not displayed in a type-ahead list and cannot be entered by the interviewer or respondent.

`n|expression`

A positive integer or expression that indicates an existing list. If the expression is a numeric variable the expression has to be placed between square brackets. If the expression is a text variable, the expression has to be placed between double quotes.

`name`

The name of an existing list.

### Example 1

```
*LIST 1
1: Citroen
2: Fiat
3: Ford
4: Hyundai
5: Mazda
6: Mitsubishi
7: Nissan
8: Opel
9: Peugeot
10: Renault
11: Suzuki
12: Toyota
13: Volkswagen
14: Volvo
15: Other *NOCON *OPEN
16: Don't know *NOCON

*QUESTION 1 *CODES 61L2 *LIST 1
What is the make of your car?
```

In this example the earlier defined list is used for the answer codes.

**Example 2**

```

*LIST Cars
1: Citroen
2: Fiat
3: Ford
4: Hyundai
5: Mazda
6: Mitsubishi
7: Nissan
8: Opel
9: Peugeot
10: Renault
11: Suzuki
12: Toyota
13: Volkswagen
14: Volvo

*QUESTION 2 *CODES 63L17 *MULTI
What brands of cars do you know?

*USELIST Cars

16: Other #1 *OPEN
17: Other #2 *OPEN

18: Don't know any brand *GOTO 9999

```

**Example 3**

```

*LIST Cars2
01: Alfa Romeo
02: B.M.W.
02: BMW
03: Citroen
04: Fiat
05: Ford
06: Hyundai
07: Mazda
08: Mitsubishi
09: Nissan
10: Opel
11: Peugeot
12: Renault
13: Rolls Royce
14: Suzuki
15: Toyota
16: Volkswagen
16: VW
17: Volvo

18: Other *NOCON
19: Don't know

*TEXTVARS lname
*PUT lname "Cars2"

*QUESTION 3 *NUMBER 81L2 *LIST "? lname"
What is the make of your car?

```

In this example, the list displayed is determined by a text variable and therefore the name must be placed between double quotes (") when you refer to it in the question.

The "Other" code is always displayed because it contains the \*NOCON command.

6.35.1.1 See Also

*LIST (definition) .....	176
*USELIST .....	282

## 6.36 \*MATRIX

### Purpose

Matrix questions are combined questions. They are another way we can collect more data, while displaying less screens as well as improving the interviewer/interviewing experience. Matrix questions are very easy to set up. You begin your matrix with the command `*MATRIX` and you end with `*ENDMATRIX`. Similar to using `*BLOCK` but the behavior is different.

### Syntax

```
*MATRIX n Qn [W|N] [*FIELD <pos>L<length>] [*ROT|*RANDOM|*ORDER] Question
text
```

<any question type including all the options that a question can have.  
Excluding buttons, nested matrix, form question, multiple questions on a page  
type block.>

```
*ENDMATRIX
```

### Description

Create multiple questions on a page in the form of a matrix or transposed matrix.

### Arguments

n	This is a positive value indicating the number of statements (rows) in the matrix.
Qn	Codes question containing the statements. Default all statements will be included to be shown.
W	Optional: only answers mentioned in Qn will be displayed
N	Optional: only answers not mentioned in Qn will be displayed.
*FIELD <pos>L<length>	Defines the data positions.
*ROT *RANDOM *ORDER	Defines the order in which the statements are to be shown.

### Expression Operator for Matrix

S	Refers to the statement in a Matrix	QxSn	Answer of question x for the nth statement
---	-------------------------------------	------	--

A list will contain definitions and no answers we refer to a question and not a list. We need a parameter to specify how many statements there are, so we can determine the data positions correctly.

**Note:** \*NOCON or \*NMUL will have no effect inside the \*MATRIX. If you need to exempt a statement from the randomization inside \*MATRIX, you need to use the \*ORDER command.

### Example 1 Single Matrix

The list of statements is formed from the categories' text fields of Qn. The column headers are formed by the question text and categories' text of the questions residing within a Matrix block. In other words, the Parent column header will be equal to the text of the question and the Child column headers will be made up of the categories' text of that question.

The screenshot shows a survey interface with a title "We picked 10 highlights of Chicago. Please rate them." Below the title is a table with 10 rows, each representing a Chicago landmark. The first column lists the landmarks, and the next five columns represent a rating scale from "Strongly agree" to "Strongly disagree". The landmarks listed are: Millennium Park, Art Institute, Michigan Avenue and the Magnificent Mile, Navy Pier, Wrigley Field, Shakespeare Theater, Museum of Science and Industry, Field Museum of Natural History, Lyric Opera, and Willis Tower SkyDeck. At the bottom of the interface are "CLEAR" and "NEXT" buttons.

	Strongly agree	Agree	Undecided	Disagree	Strongly disagree
Millennium Park					
Art Institute					
Michigan Avenue and the Magnificent Mile					
Navy Pier					
Wrigley Field					
Shakespeare Theater					
Museum of Science and Industry					
Field Museum of Natural History					
Lyric Opera					
Willis Tower SkyDeck					

```
*TEMPLATE "NfieldChicago"
*QUESTION 10 *CODES 61L2 *DUMMY
1:Millennium Park ** category text becomes the 1st statement
2:Art Institute ** category text becomes the 2nd statement, etc.
3:Michigan Avenue and the Magnificent Mile
4:Navy Pier
5:Wrigley Field
6:Shakespeare Theater
7:Museum of Science and Industry
8:Field Museum of Natural History
9:Lyric Opera
10:Willis Tower SkyDeck

*MATRIX 10 Q10 *FIELD 63L10
We picked 10 highlights of Chicago. Please rate them. ** this becomes the header

*QUESTION 20 *CODES 1L1

1:Strongly agree ** category text becomes the answer box column header
2:Agree ** category text becomes the answer box column header, etc.
3:Undecided
```

4:Disagree  
5:Strongly disagree

\*ENDMATRIX

\*END

### Example 2 Mixed Matrix

\*TEMPLATE "NfieldChicago"

\*TEXTVARS Othermovie

\*LIST "movies"

1: Godfather I                      \*\* category text becomes the 1st statement

2: Turkish Fruit                    \*\* category text becomes the 2nd statement

3: Titanic                            \*\* category text becomes the 3rd statement

\*QUESTION 1 \*CODES 61L4 \*MULTI

Which of these movies would you like to watch?

\*USELIST "movies"

4: Other \*OPEN \*SAVE Othermovie                      \*\* open answer becomes the 4th statement

\*QUESTION 901 \*CODES L4 \*MULTI \*DUMMY

\*USELIST "movies"

4: \*?Othermovie

\*INCLUDE Q901 Q1

\*UIRENDER "CODES=Horizontal"                      \*\* all the CODES questions will be rendered horizontal

\*MATRIX 4 Q901 W \*FIELD 65L48 \*UIRENDER "Mixed"

Which of these have you seen already?                      \*\* this becomes the main header

\*QUESTION 2 \*CODES L1

Seen                                  \*\* this becomes the column header

1: yes                                \*\* category text becomes the answer box column header

2: no                                 \*\* category text becomes the answer box column header

\*QUESTION 3 \*CODES L1 \*IF[Q2, 1]

Please rate them                      \*\* this becomes the second column header

1: Very Bad - 1                      \*\* category text becomes the answer box column header

2: 2                                 \*\* category text becomes the answer box column header

3: 3                                 \*\* category text becomes the answer box column header

4: 4 - Very Good                      \*\* category text becomes the answer box column header

\*ENDMATRIX

\*UIRENDER "CODES="                      \*\* to reset the CODES questions back to original

\*\* Using the expression operator

\*QUESTION 5 \*OPEN 100L1 \*IF[Q1, 1 & Q2S1, 2]

\*\* will only ask this question if Godfather I was not watched but mentioned as one of the movies that the interviewer would like to watch

Why did have you not watched the Godfather I?

This is what question 1 looks like:

### Which of these movies would you like to watch?

☒ Godfather I

☒ Turkish Fruit

☒ Titanic

☒ Other

CLEAR

NEXT

Powered by NPD

And this is the Matrix:

### Which of these have you seen already?

	Seen		Please rate them			
Godfather I	<input checked="" type="checkbox"/> yes	<input type="checkbox"/> no	Very Bad - 1	2	3	<input checked="" type="checkbox"/> 4 - Very Good
Turkish Fruit	<input type="checkbox"/> yes	<input checked="" type="checkbox"/> no				
Titanic	<input checked="" type="checkbox"/> yes	<input type="checkbox"/> no	Very Bad - 1	2	<input checked="" type="checkbox"/> 3	4 - Very Good
Matrix	<input checked="" type="checkbox"/> yes	<input type="checkbox"/> no	Very Bad - 1	2	3	<input checked="" type="checkbox"/> 4 - Very Good

BACK

CLEAR

NEXT

Powered by NPD

## Example 3 Multiple Choice Matrix

**There is a lot to do during the day and night, but what do you think of...**

Please rate all statements

✓ Chicago History Museum	Interesting	Fun	Expensive	Must see	N/A
✓ Brookfield Zoo	Interesting	Fun	Expensive	Must see	N/A
✓ Grant Park	Interesting	Fun	Expensive	Must see	N/A
✓ 360 Chicago	Interesting	Fun	Expensive	Must see	N/A
✓ Water Tower Place shopping centre	Interesting	Fun	Expensive	Must see	N/A

Powered by NPS

\*TEMPLATE "NfieldChicago"

\*QUESTION 10 \*CODES 61L1 \*DUMMY

List of attractions.

1:Chicago History Museum

2:Brookfield Zoo

3:Grant Park

4:360 Chicago

5:Water Tower Place shopping center

\*MATRIX 5 Q10 \*FIELD 62L25 \*UIOPTIONS "instruction=Please rate all statements"

There is a lot to do during the day and night, but what do you think of...

\*QUESTION 320 \*CODES 1L5 \*MULTI

1:Interesting

2:Fun

3:Expensive

4:Must see

5:N/A \*NMUL

\*ENDMATRIX

\*END

## Example 4 Using expression operator for \*MATRIX to refer to a statement inside the loop

\*VARS LoopCounter

\*REPEAT 5

\*PUT LoopCounter [?R]

\*QUESTION 1 \*CODES L1

Answer for loop\*?LoopCounter

1:First Answer

2:Second Answer

\*ENDREP

\*QUESTION 2 \*CODES L1 \*IF [Q1S2,1]

Under condition

1:ok

In this example, we are referencing the question inside the loop (Question 1) outside the loop (in Question 2). We want to show Question 2 only if we got an answer 1 ("First Answer") to Question 1 during the second time the loop was run. The way we can do it is by using the S (expression operator for \*MATRIX). So Q1S2 means the answer to Q1 given during the second run of the loop.

**Example 5 Using expression operator for \*MATRIX to refer to a statement inside the \*MATRIX**

This example is similar to example 4, but here we are showing Question 2 only if the respondent chooses option 1 (Interesting) of the second matrix row (Brookfield Zoo).

```
*TEMPLATE "NfieldChicago"

*QUESTION 10 *CODES 61L1 *DUMMY
List of attractions.
1:Chicago History Museum
2:Brookfield Zoo
3:Grant Park
4:360 Chicago
5:Water Tower Place shopping center

*MATRIX 5 Q10 *FIELD 62L25 *UIOPTIONS "instruction=Please rate all statements"
There is a lot to do during the day and night, but what do you think of...

*QUESTION 320 *CODES 1L5 *MULTI

1:Interesting
2:Fun
3:Expensive
4:Must see
5:N/A *NMUL

*ENDMATRIX

*QUESTION 2 *CODES L1 *IF [Q320S2,1]
Under condition
1:ok

*END
```

6.36.1.1 See Also

*BLOCK.....	108
*ID.....	160
*ORDER .....	206
*REPEAT ... *ENDREP .....	230

## 6.37 \*MAX

### Purpose

Sets a maximum.

### Syntax

```
*MAX <n| [expression]>
```

### Description

This command is always used in combination with \*QUESTION and must be specified after the question type definition on the same line.

- In combination with \*CODES \*MULTI it defines the maximum number of answers allowed.
- In combination with \*NUMBER it defines the maximum value that may be entered.

### Arguments

n|expression

This is a positive integer or expression that gives the maximum answer value.

### Example 1

```
*QUESTION 91 *NUMBER L2 *MAX 65
What is your age?

(Maximum is 65)

*QUESTION 92 *CODES L1
Do you smoke?

1: Yes
2: No

*QUESTION 93 *NUMBER L2 *MAX [ Q91 ]
At what age did you start smoking?

(Maximum is current age)
```

### Example 2

```
*QUESTION 1 *NUMBER L1
How many cars do you have?

*QUESTION 2 *CODES L16 *MULTI *MAX [ Q1 ] *IF [ Q1>=1 ]
What brand(s) of car(s) do you have

(Maximum as many brands as you have cars)

1: Citroen
2: Fiat
3: Ford
4: Hyundai
5: Mazda
6: Mitsubishi
7: Nissan
8: Opel
9: Peugeot
10: Renault
11: Suzuki
12: Toyota
13: Volkswagen
14: Volvo

15: Other *NOCON *OPEN
16: Don't know *NMUL
```

In this example, no more answers can be given than the number of cars mentioned in Q1.

6.37.1.1 See Also

*MIN.....	193
*MULTI.....	195
*NUMBER.....	201
*RANGE.....	226

## 6.38 \*MERGE

### Purpose

Inserts a script fragment into questionnaire.

### Syntax

```
*MERGE {position} {FragmentFilename}
```

### Description

The specified script fragment is inserted on the place of this command. \*MERGE cannot be used in a questionnaire that is already merged itself.

### Arguments

position

The start position with which all fixed data fields in the merged questionnaire will be raised.

FragmentFilename

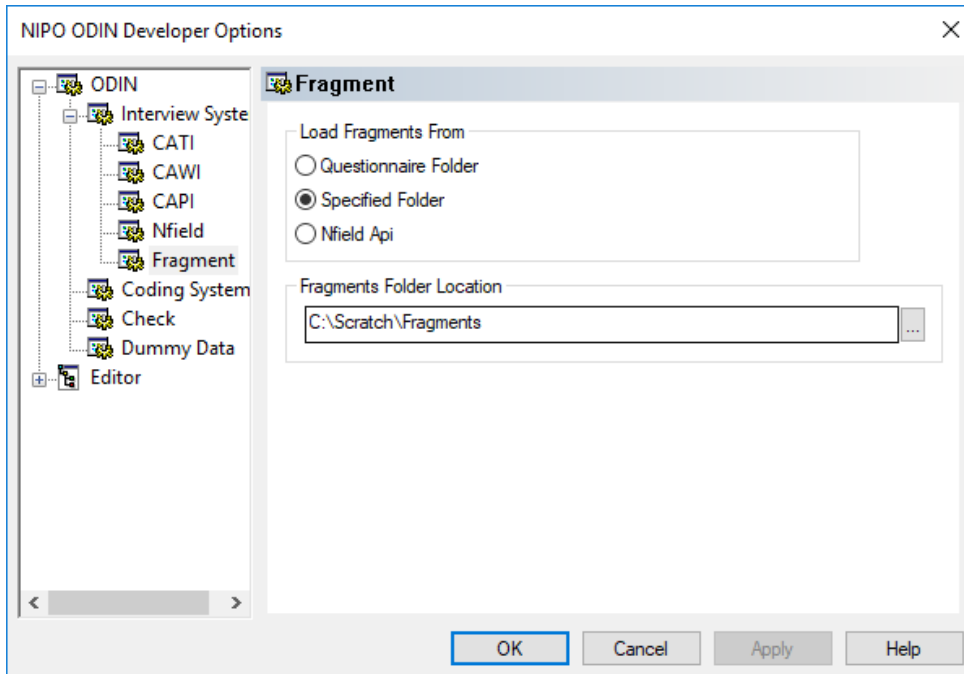
Name of the questionnaire fragment file that is to be merged.

### Remarks

- Upload the fragment file before questionnaire. The fragment file can **only** be uploaded via public API:  

```
POST /v1/Surveys/:SurveyId/ScriptFragments/:FragmentName
```

 After the fragment upload, you can upload the ODIN script with the fragment reference using the Nfield Manager or the public API.
- Fragment name is case sensitive
- The questionnaire to be merged must have fixed data fields (from position 1, instead of 61).
- No start position can be defined if the questionnaire is to be merged in a language section.
- The \*MERGE command cannot be used conditionally as it is evaluated when the questionnaire is loaded into memory.
- You can set up the folder location for file fragments in ODIN Developer, Settings/Options:



- Font definitions can also be put into a separate files to be merged.

#### Example 1

Let's say we have this file fragment:

File: Fruitlist

\*LIST "fruits"

1: Apple

2: Apricot

3: Avocado

4: Banana

6: Bilberry

7: Blackberry

....

We need to upload this file to Nfield.

In our questionnaire we merge this file and can use the list from it:

```
*MERGE Fruitlist

*QUESTION 9003 *CODES L3
What is your favorite fruit?

**Can now refer to the list defined in the script fragment
*USELIST "fruits"
```

**Note:** The `*MERGE` command can be used inside translations, for instance, in the example above you could have `*MERGE FruitListNL` inside the `*LANGUAGE Dutch`, and `*MERGE FruitListEN` inside `*LANGUAGE English` parts. So you would have to have 2 different list files for the 2 languages.

6.38.1.1 See also

An NIPO Academy video on how to manage long lists: <https://youtu.be/l3zzhhQD9EI>

## 6.39 \*MIN

### Purpose

Sets a minimum.

### Syntax

```
*MIN <n|[expression]>
```

### Description

This command is always used in combination with \*QUESTION and must be specified after the question type definition on the same line.

- In combination with \*CODES \*MULTI it defines the minimum number of answers allowed.
- In combination with \*NUMBER it defines the minimum value that must be entered.

### Arguments

n|expression

A positive integer or expression that gives the minimum answer value or the initial length or position.

### Example 1

```
*QUESTION 91 *NUMBER L2 *MIN 18 *MAX 65
```

What is your age?

(Minimum is 18, maximum is 65)

```
*QUESTION 101 *NUMBER L4 *MIN [ -999 ]
```

What was the gross profit for your company last year?

(Type the profit in Millions, if the company made a loss, use negative numbers)

### Example 2

```
*QUESTION 1 *CODES L18 *MULTI *MIN 3
```

What are your three favorite car brand(s)?

(Minimum 3 brands)

- 1: Citroen
- 2: Fiat
- 3: Ford
- 4: Hyundai
- 5: Mazda
- 6: Mitsubishi
- 7: Nissan
- 8: Opel
- 9: Peugeot
- 10: Renault
- 11: Suzuki
- 12: Toyota
- 13: Volkswagen
- 14: Volvo

15: Other #1 \*OPEN

16: Other #2 \*OPEN

17: Other #3 \*OPEN

18: Don't know \*NOCON \*NMUL

6.39.1.1 See Also

*MAX.....	188
*MULTI.....	195
*NUMBER.....	201
*RANGE.....	226

## 6.40 \*MULTI

### Purpose

Specifies a question as a multiple coded question.

### Syntax

```
*MULTI [ [pos] L<length> | <pos> ]
```

### Description

Specifies a question as a multiple coded question. This command may be used in combination with a closed question or an open question and must be specified after these commands on the same line. The length of the data field specification for \*CODES has to be at least as long as the highest code value in the set of presented codes. The answers are stored as a string of 0 and 1 values in the data file with the closed answers. Here answer code n corresponds with the n<sup>th</sup> position in the string. A 1 on that position means that the answer was given.

- If \*MULTI is used in combination with \*OPEN, the question will be regarded as a multiple answer question when coding open answers.
- It is also possible to use \*MULTI without \*CODES, but then the additional parameter for the position length per item must be included (see **Example 2** below).

### Arguments

pos

An optional data field specification where the order of the mentions is stored in the data file with the closed answers.

length

Stores the order of mentioned (the order in which the user picked the codes during the interview). This has to be as long as the number of picked codes to be saved, multiplied by the number of digits for the highest code. If the length is shorter codes will be truncated. For example, if highest code is 12, and you need to store the first 3 order of mentioned (the first 3 answers the respondent mentioned), you need the length of: 2 (the number of digits in 12) multiplied by 3 = 6.

### Remarks

To specify a maximum number of answers to be mentioned, use \*MAX. Similarly, to specify a minimum number of answers to be mentioned, use \*MIN.

### Example 1

```
*QUESTION 1 *CODES 61L10 *MULTI 71L6
```

What PC makes do you know?

```
1: Acer
2: Compaq
3: Dell
```

4: Hewlett Packard  
 5: IBM  
 6: Philips  
 9: Other \*OPEN  
 10: Don't know \*NMUL

In this example, all mentions are stored in position 61L10. The order of the first three answers mentioned is stored in position 71L6. For instance, if the categories 5, 1, 6 and 4 are mentioned, position 71L2 will hold the value 05, position 73L2 will hold the value 01 and position 75L2 will hold the value 06.

### Example 2

```
*QUESTION 1 *MULTI L12 2
Which of the following colors do you like?

1: Red
2: Green
3: Purple
4: Orange
5: Blue
6: Aquamarine

10: None *NMUL
```

In this example we use \*MULTI without \*CODES, but with the additional parameter for the position length per item (2).

#### 6.40.1.1 See Also

*CODES.....	114
*NMUL.....	197
*OPEN (question type) .....	203

## 6.41 \*NMUL

### Purpose

Specifies that the code in question cannot be given as answer in combination with other codes.

### Syntax

\*NMUL

### Description

This command is always used after a code definition and must be specified behind this code on the same line. Specifies that the code can not be given as answer in combination with other codes.

### Remarks

This command is only effective for questions that allow multiple answers.

### Example

```
*QUESTION 1 *CODES 61L10 *MULTI
What PC makes do you know?

1: Acer
2: AST
3: Compaq
4: Dell
5: Hewlett Packard
6: IBM
7: Philips
8: Tulip
9: Other *OPEN
10: Don't know *NMUL
```

In this example, selecting code 10 will automatically de-select other codes and vice versa.

## 6.42 \*NOCON

### Purpose

Specifies that the code in a question doesn't come under control.

### Syntax

\*NOCON

### Description

This command is always used behind a code definition and must be specified after this code on the same line. If the codes of the current question are displayed under control by means of \*CONTROL then this code is ignored and always displayed. This command is also used for the question options \*RANDOM and \*ROT. It excludes all codes starting from the code that have \*NOCON as option from randomization, inversion and rotation, respectively.

If used in combination with \*LIST and \*NUMBER or \*ALPHA the code is always displayed regardless of the text entered as the narrowed-down search.

### Example

```
*QUESTION 1 *CODES 61L10 *MULTI
What PC brands do you know?
(Unaided awareness)

1: Acer
2: AST
3: Compaq
4: Dell
5: Hewlett Packard
6: IBM
7: Philips
8: Tulip

9: Other *OPEN
10: Don't know any makes *NMUL

*QUESTION 2 *CODES 71L10 *MULTI *CONTROL Q1 N
Which of the following PC makes do you know?

(Aided awareness)

1: Acer
2: AST
3: Compaq
4: Dell
5: Hewlett Packard
6: IBM
7: Philips
8: Tulip

10: None of the above *NMUL *NOCON
```

In this example, code 10 in question 2 is always displayed, even when it is mentioned in question 1.

6.42.1.1 See Also

\*CONTROL.....116

\*LIST (definition)..... 176

\*RANDOM..... 221

\*ROT .....243



## 6.43 \*NON

### Purpose

Permits non-response.

### Syntax

\*NON

### Description

This command can be specified on any \*QUESTION or on a \*ALPHA or a \*NUMBER. Non-response is permitted at this question or field. It is permitted to press ENTER or **OK** without giving an answer.

### Example

```
*QUESTION 1 *NUMBER 61L4 *NON
What is your postcode?
```

In this example, it is possible to press ENTER or **OK** without filling in the postcode.

#### 6.43.1.1 See Also

*ALPHA .....	104
*FORM .....	141
*NUMBER .....	201
*QUESTION .....	220

## 6.44 \*NUMBER

### Purpose

Defines a numerical question.

### Syntax

```
*NUMBER [pos]L<length>[.fraction]|<pos>
```

### Description

This command is always used in combination with \*QUESTION and must be specified behind this command on the same line. Defines a question as a numerical question and expects a number as an answer.

### Arguments

*pos*

The data field specification where the given answers is stored in the data file with the closed answers.

*length*

The length of the data field. The length of the data field defines the number of digits of the maximum value of the question. In case of a floating point value, the length of the data field is length+fraction.

*fraction*

The number of decimals that is allowed to be entered in a floating-point value. The fraction is stored in the data field without the decimal separator. Which separator (point or comma) is to be used by the interviewer or respondent depends on the regional configuration settings.

### Remarks

- The answer is right-aligned and stored with leading zeros in the answer field.
- The answer can consist of an integer or a floating-point value, as defined in the data field specification.
- The decimal point in the syntax is never stored in the answer record.
- The maximum value can be set with the \*MAX or \*RANGE command, but is also limited by the number of positions in the data field.
- The minimum value can be set with the \*MIN or \*RANGE command.
- Negative values can only be entered when the \*MIN or \*RANGE command is defined with a negative value.
- Negative values are stored with a preceding minus sign in the data file. Note that you may require an extra position in the field definition.
- Positive values are stored without sign.

### Example

```
*QUESTION 1 *NUMBER 61L3.2 *MAX 100
What percentage of your income do you spend on clothing?
```

In this example, you can enter a value with two decimals, for instance 60.75. In the data field 06075 is stored in 5 positions.

6.44.1.1 See Also

*ALPHA.....	104
*CODES.....	114
*LIST (question option).....	179
*MAX.....	188
*MIN.....	193
*OPEN (question type) .....	203
*RANGE.....	226

## 6.45 \*OPEN (question type)

### Purpose

Defines an open question.

### Syntax

```
*OPEN [pos]L<length>|<pos>
```

### Description

This command is always used in combination with `*QUESTION` and must be specified after this command on the same line. It defines a question as an open question that expects an alpha-numerical answer. This open answer can be of virtually unlimited length. The answer is stored in a separate data file for open answers (`survey.o`). Although the answers are not stored in the data file (DAT-file) with the closed answers, you must reserve data positions in the DAT-file. It is recommended but not required to reserve positions that will be used for merging the coded open-ended answers.

### Arguments

`pos`

The start of the data field reserved in the DAT-file. This position is also be used as identification to match O-file open-ended answers with a DAT-file data position, to enable coding open answers.

`length`

The length of the data field.

### Remarks

- A text box appears in which you can type the complete answer.
- Optionally the command `*MULTI` may be added. This has no influence on the interviewing process. When coding open answers the question will be treated as a multiple-coded answer question.

### Example 1

```
*QUESTION 1 *OPEN 61 *NON
Do you have comment or remarks regarding this questionnaire?
```

In this example, you can enter any text. This text is stored in the O-file. No codes are stored in the DAT-file, so the related data field (61) remains empty. It is however used to reserve some space for coding open-ended answer. Because of the `*NON` command, it is also possible not to enter anything in which case no open ended answer are stored.

### Example 2

```
*QUESTION 2 *OPEN 62L100 *MULTI *BUT 100 "No suggestions"
Do you have any suggestions on how to improve this product?
```

In this example, you can enter any text. This text is stored in the O-file. The positions reserve space to code open-ended answers afterwards (maximum 100 categories, multiple). If the button **No suggestions** is pressed, no open answer is stored, but code 100 is marked in the data file.

6.45.1.1 See Also

*ALPHA.....	104
*CODES.....	114
*NUMBER.....	201
*OPEN (codes option).....	205
*REC.....	228

## 6.46 \*OPEN (codes option)

### Purpose

Specifies a code as an open code.

### Syntax

\*OPEN

### Description

This command is always used after a code definition and must be specified after this code on the same line. Specifies a code as open code and expects, if the code is part of the answer, a literal answer that can be of virtually unlimited length. The entered answer is stored in a separate data file for open answers (`survey.o`).

### Remarks

- A text box appears in which you can type the complete answer.
- This codes option is not allowed for code 0 in single-coded questions.

### Example

```
*QUESTION 1 *CODES L1
Do you live in Amsterdam or somewhere else?

1: Yes, in Amsterdam
2: No, somewhere else *OPEN
```

In this example, if answer 2 is selected a location can be entered. This text is stored in the O-file, while the code (2) is stored in the DAT-file.

#### 6.46.1.1 See Also

```
*CODES.....114
*OPEN (question type) .....203
```

## 6.47 \*ORDER

### Purpose

Set the order of codes in a code list or the order of execution in a \*REPEAT loop or matrix.

### Syntax

\*ORDER <var | QnM | QnR>

### Description

You can define the order of codes in a \*CODES question and the order of a \*REPEAT loop according to the contents of an array variable or the order of mentions of another question.

- Both variable names and questions with order specifier (QnM or QnR) may be used in the \*ORDER command.
- \*ORDER may be used in a \*REPEAT, \*QUESTION or \*MATRIX statement.
- Invalid orders in the argument with \*ORDER are ignored. This means if an order number is used twice or if an index in the variable is empty, all remaining code are displayed in regular ascending order.
- \*ORDER cannot be used together with \*FORM, \*RANDOM or \*ROT.
- The order of mentions in a question are available through QnMi where n is the question number and i is the index of the mention.
- The display order of a random-question is available through QnRi, where n is the question number and i is the index of the random order.
- The operators QnM and QnR are available, independent of whether the order is saved or not.
- A syntax check produces a warning message if the variables M or R are defined. If these variable names are used, the operator QnMi and QnRi are not available.
- The items in a \*REPEAT block or code list which are not in the \*ORDER command, are displayed in the order as if there was no \*CONTROL command. This means that if you only want to show the items mentioned in a previous \*CODES question in the order they were mentioned, you have to use both a \*ORDER and a \*CONTROL command.
- \*ORDER <ArrayVariable> on a \*QUESTION displays the categories in the order in which they are in the array, by label.

### Arguments

var

This is a variable name of an array variable (\*VARS name[n]). The values in this variable set the order in which the codes are displayed or the order in which the \*REPEAT loop is processed.

QnM

This is order of mentions of a previous \*MULTI question.

QnR

This is the order in which a previous question with \*RANDOM or \*ROT was displayed.

### Example 1

```
*QUESTION 1 *CODES 61L5 *MULTI 66L4
Which brands do you know?
```

(Int. do NOT help!

Type the answers in the same order as given by the respondent)

1: brand A  
2: brand B  
3: brand C  
4: brand D

5: None of these \*NMUL \*NOCON

\*QUESTION 2 \*CODES 71L5 \*MULTI 76L4 \*CONTROL Q1 W \*ORDER Q1M  
Which brand do you use most often

(Brands are presented in the order that was entered in Q1)

1: brand A  
2: brand B  
3: brand C  
4: brand D

5: None of these \*NMUL \*NOCON

### Example 2: \*ORDER Command with Least Filled Quota

Show quota levels in order of least filled using the array returned by the \*GETLSTQLIST as the argument for an \*ORDER command.

\*\*Least filled command with ordering  
\*TEMPLATE NfieldChicago

\*SAMPLEDATA ModelNr  
\*TEXTVARS ModelLevel[4]  
\*VARS NOfReturnedModelLevels

\*GETLFQLIST NOfReturnedModelLevels ModelLevel ModelNr

\*\*Show least filled order using the array returned by the \*GETLFQLIST as the argument for an ORDER command  
\*QUESTION 10 \*CODES 61L1 \*ORDER ModelLevel

1:Model1  
2:Model2  
3:Model3  
4:Model4

\*END

**Note:** \*HEADING/\*GROUP command cannot work in combination with \*ORDER command.

#### 6.47.1.1 See Also

Least Filled Quota.....	83
*CODES.....	114
*FORM .....	141
*GETLFQLIST .....	145
*RANDOM.....	221
*REPEAT ... *ENDREP .....	230
*ROT .....	243

## 6.48 \*PAGE

### Purpose

Defines a page of text.

### Syntax

\*PAGE

### Description

Defines a page of text. This command should be at the beginning of a line or under condition using \*IF. The text consists of all following lines up to the first new command that is at the beginning of a line, excluding \*\*, \*? and \*FONT (switching). During the interview this text is displayed. To proceed only ENTER or **OK** need to be pressed. The text usually contains an explanation for the next question or questions.

### Example 1

```
*PAGE
And now some questions about snacks.

*QUESTION 1 *CODES 61
Did you have a snack today?

1: Yes
2: No
```

In this example, a short introduction to the next (set of) question(s) is displayed.

### Example 2

```
*IF [ TMP = 1 ] *PAGE This text will appear on a separate page
```

After an \*IF statement, you may immediately specify an \*PAGE with text.

### 6.48.1.1 See Also

\*ID..... 160  
\*QUESTION ..... 220

## 6.49 \*PICT (question option)

### Purpose

Displays a picture or allows user to play an audio file.

### Syntax

```
*PICT <filename>
```

### Description

Displays a picture or allows user to play an audio file on the screen. The \*PICT command may also be used in combination with \*PAGE.

Pictures in the following formats may be used:

- BMP
- JPG (normal and progressive encoding)
- MP3
- PNG

If MP3 format is used, a PLAY button is displayed, so the user can play that audio file.

### Arguments

*filename*

The file name of the image that must be displayed. This file should first be uploaded to Nfield survey. Note that screen resolution for respondents or interviewer workstations might vary. Always create a picture in the lowest resolution that is used for optimum results.

### Example

```
*TEMPLATE NfieldChicago

*q1

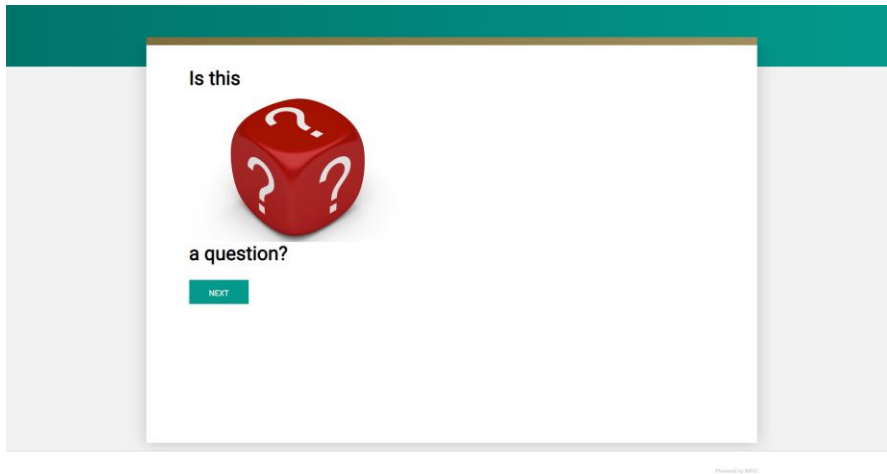
Is this

*PICT question.jpg

a question?
```

This example displays a question with the picture `question.jpg`.

## Result



### 6.49.1.1 See Also

*PAGE .....	209
*PICT (codes option).....	212

## 6.50 \*PICT (codes option)

### Purpose

Displays a picture on a code.

### Syntax

```
*PICT <filename>
```

### Description

This command is always used on a code definition and must be specified after this code on the same line. A picture is identified by the file name. The picture file must be uploaded to the survey first.

### Example

```
*TEMPLATE NfieldChicago









*QUESTION 10 *CODES 61L9 *MULTI *UIOPTIONS "columns=2"
What did you have this morning for breakfast?

1:*PICT "banana.png"Banana
2:*PICT "kiwi.png"Kiwi
3:*PICT "orange.png"Orange
4:*PICT "pear.png"Pear
5:*PICT "strawberry.png"Strawberry
6:*PICT "boterham.jpg"Sandwich
7:*PICT "smoothy.jpg"Smoothy
8:*PICT "muesli.jpg"Cereal

9:None of the above *NMUL
*END
```

## Result

What did you have this morning for breakfast?

 <p>Banana</p>	 <p>Kiwi</p>
 <p>Orange</p>	 <p>Pear</p>
 <p>Strawberry</p>	 <p>Sandwich</p>
 <p>Smoothy</p>	 <p>Cereal</p>

None of the above

CLEAR NEXT

Powered by NFI

**Note:** Accessibility options for pictures are regulated by the template used. For example, NfieldChicago template does offer accessibility options for pictures. Please see [NfieldChicago documentation](#) for more information.

6.50.1.1 See Also

\*PICT (question option) ..... 210

## 6.51 \*PROPERTIES

### Purpose

Used to define the custom (user) defined properties at various levels (question/categories) in ODIN script. These properties can be defined at questions, category, and lists.

### Syntax

```
*PROPERTIES "{ [key]=[value] } { ; [key]=[value] } { ; ... } "
```

### Description

Properties consist of one or more key/value pairs, separated by a semi-colon. Nfield considers '**fixed key/value pair**', which means the property names and their values cannot be variables and replaced with other variables or values. For example – in a multi-county setup, it is not possible to control a global brand list with one custom property by changing its values at run time.

### Remarks

- Properties cannot be defined in \*LANGUAGE sections, only on questions and categories in the default language section.
- Filtering on the key name is case-*insensitive*. Values of properties are case-sensitive.
- Custom properties are a useful way of defining and storing user specific information at question/categories/list level. This feature has high potential and use in automation where information/resources are picked up based on request/user input from global setup/repository. They are also useful in driving variable content in the script.
- Properties can be used to filter the categories shown in questions with the [\\*USELIST](#) command.
- A built-in function has to be called for fetching values of custom properties. This function needs few compulsory arguments. Below describes how it should be used in ODIN script –
  - `?property(Q[question number],[category number],"key")`
  - `property()` is a function to fetch the custom prop values. It cannot be used without passing arguments.
  - `Q[question number]` is a question from which custom prop value is to be fetched.
  - `[category number]` is a category name (response index) which will be fetched from the above-mentioned question. This is numeric index passed with categories. The category number can also be an expression.
  - `Key` is a name of custom property value of which is to be fetched in.

**Example 1**

```
*TEXTVARS moodtype
*PUT moodtype [?property(Q1, Answer, "Feeling")]
```

Since it is a function, it returns property value as text type which is to be stored into text variable before use. You can directly use the function as per your needs. The function can also be used as expression under conditional statements.

**Example 2**

```
*IF [?property(Q1, Answer, "mood") = "Bored"]
```

**Example 3**

Filtering a question based on custom property defined in other question

```
*VARS Answer
*QUESTION 1 *CODES 61L1 *PROPERTIES "segment=hotel" *SAVE Answer
How will you rate the service during your stay at hotel Marriot?
1: Very Good *PROPERTIES "emotion=Positive;mood=Happy"
2: Good *PROPERTIES "emotion=Positive;mood=Happy"
3: Average *PROPERTIES "emotion= Negative;mood=Bored"
4: Bad *PROPERTIES "emotion=Negative;mood=Bored"

*QUESTION 2 *OPEN 62L1 *IF [?property(Q1, Answer, "mood") = "Bored"]
What was that which made you feel bored?
```

Here `Question 1` is about rating the overall service at hotel Marriot. The answer (index) is saved in a variable (Answer) which was later used in `property()` to fetch the value of custom property **mood**.

Next question (`Q2`), is filtered on custom property **mood** and asked only if respondents have felt **bored** (i.e. value of custom property mood).

**Example 4**

Summarizing (netting) the responses of one question into other dummy (summary) variable at back of script.

```
*VARS Answer, propval
*QUESTION 1 *CODES 61L2 *SAVE Answer
Please enter region you live in.
1: Schleswig-Holstein *PROPERTIES "DE_Reg=1"
2: Hamburg *PROPERTIES "DE_Reg=1"
3: Niedersachsen *PROPERTIES "DE_Reg=1" 4: Bremen *PROPERTIES
"DE_Reg=1"
5: Nordrhein-Westfalen *PROPERTIES "DE_Reg=2"
6: Hessen *PROPERTIES "DE_Reg=3"
7: Rheinland-Pfalz *PROPERTIES "DE_Reg=3"
8: Saarland *PROPERTIES "DE_Reg=3" 9: Baden-Wuerttemberg
*PROPERTIES "DE_Reg=3"
10: Bayern *PROPERTIES "DE_Reg=3"
11: Berlin *PROPERTIES "DE_Reg=4"
12: Mecklenburg-Vorpommern *PROPERTIES "DE_Reg=4"
13: Sachsen-Anhalt *PROPERTIES "DE_Reg=4" 14: Brandenburg
*PROPERTIES "DE_Reg=4"
15: Thuringen *PROPERTIES "DE_Reg=4"
16: Sachsen *PROPERTIES "DE_Reg=4"
```

```

*QUESTION 2 *CODES 63L1 *DUMMY
1: North
2: West
3: South
4: East

*PUT propval [?PROPERTY(Q1, Answer, "DE_Reg")]
*COPY Q2 [propval]

*PAGE
propval = *? propval
Answer = *? Answer

```

Here `Question 1 (region)` is asked from respondent, and zone at question 2 is back-coded using custom properties defined at `Question 1`.

There could be many ways of implementing it using custom properties; this is just one way to fulfill the requirement of summarizing regions into zones.

#### Example 5

Use in multi-language projects the property for the data in the sample rather than the category text for quota control purposes.

```

*SAMPLEDATA sGender
*VARS ansQ102
*QUESTION 102 *CODES L1 *SAVE ansQ102
Select your gender.

1: male *PROPERTIES "gender=Male"
2: female *PROPERTIES "gender=Female"

*PUT sGender [?PROPERTY(Q102, ansQ102, "gender")]

*LANGUAGE Dutch
*QUESTION 102
Kies uw geslacht.

1: man
2: vrouw

```

#### 6.51.1.1 See Also

\*USELIST ..... 282

## 6.52 \*PUT

### Purpose

Puts data in variable.

### Syntax

```
*PUT <var> <"text" | [value] | Qn | Qn,code | [expression] >
```

### Description

Replaces the contents of a variable by data indicated in the second argument. This command is also allowed under condition. You can transfer the contents of a numeric variable to a text variable and vice versa.

When transferring the contents of a text variable to a numeric variable the text variable must contain a value. The following rules apply:

- Any fractions are transferred as well.
- The transfer stops at the first non-numerical character.
- If the first character is not numeral the result value is 0.
- When transferring the contents of a numeric variable to a text variable, it is rounded to the next integer, unless another format is specified with the `string manipulation` commands.

### Arguments

`var`

This is an earlier defined variable, which receives the data.

`text`

Direct text enclosed in quotes (single or double) with eventually embedded contents of a variable. If quotes are to be displayed in the text use the other quotes to embed the text.

`value`

A value enclosed in square brackets.

`Qn`

The answer to the question will be used as data (**Note:** it will not work if `Qn` is an `*OPEN` question. So, an open question should not be used here).

`Qn,code`

The code of a question is used as data. In a text variable the descriptive text of the code is used.

`Qn,var`

A code of question `n` is used as data. This code is defined in the variable `var`. In a text variable the descriptive text of the code is used, rather than the code number.

`Qn,?R`

A category of question `n` will be used as data. This code is defined by the repetition number. In a text variable the descriptive text of the code is used, rather than the code number.

`expression`

The result of the expression is stored in the variable.

**Remarks**

- A text value may range several lines. All lines up until the closing of the quotes are considered part of the text. Line-feeds are also considered part of the text. If a closing quote is missing for a command, an error occurs during the syntax check.
- A text may not contain NIPO ODIN commands other than the \*? and \*FONT (switching) directives.

**Sample table**

If a \*SAMPLEDATA variable is used, the last value stored in the variable is stored in the sample table when the interview ends (either definitely or indefinitely).

**Example 1**

```
*VARS age
*QUESTION 1 *NUMBER 61L2
How old are you?

*PUT age Q1
```

In this example, the answer of question 1 is stored in the variable age.

**Example 2**

```
*TEXTVARS paper
*QUESTION 2 *CODES 61L7 *MULTI

Which of the following newspapers do you know?

1: La Repubblica
2: La Stanza
3: The Mirror
4: The New York Times
5: Le Figaro
6: La Libération
7: None *NMUL

*REPEAT 6
*IF [Q2,?R] *ELSE *GOTO 4
*PUT paper Q2,?R
*QUESTION 3 *CODES 1

How often do you read *?paper?

1: Daily
2: Once a week
3: Once a month
4: Don't know

*QUESTION 4
*ENDREP
```

**Example 3**

```
*TEXTVARS sex
*QUESTION 7 *CODES 91

Gender?

1: Man *PUT sex "Men"
2: Woman *PUT sex "Women"
```

**Example 4**

```
*SAMPLEDATA Gender
*QUESTION 7 *CODES 91

Gender?

1: Man
2: Woman
```

\*PUT Gender [Q7]

The value 1 or 2 is stored in the `Gender` field in the sample table (requires the field to exist).

6.52.1.1 See Also

*?.....	99
*SAMPLEDATA .....	246
*TEXTVARS .....	268
*VARS .....	286

## 6.53 \*QUESTION

### Purpose

Defines a question.

### Syntax

\*QUESTION <n>|X

or

\*Q <n>|X

### Description

Defines a question with question number *n*. This command has to be at the beginning of a line.

The question consists of all lines up to the first command that is at the beginning of a line (excluding for \*\*, \*?, \*FONT and \*PICT). After the command \*QUESTION (or \*Q) there are often more commands on the same line. These commands on the question line define the specific conditions and options when answering this question.

### Arguments

*n*

A positive number that indicates the question number. Question numbers are made up of positive numbers ( $\geq 1$ ). No alphabetic or other characters are allowed (except for *x*, see below). Each question number has to be unique within the (sub-) questionnaire. You may use any question number in the range 1 to 2147483647 in any order - the order of the questionnaire is not set by the question numbers. The NIPO ODIN Developer allows you to renumber question numbers if necessary.

*x*

A placeholder value to be filled with a question number automatically later, by renumbering the questionnaire. It is not possible to set routing or filtering for placeholder question numbers.

### Remarks

Questions without question text and codes are considered dummy questions and are skipped during execution.

### Example 1

```
*QUESTION 1 *CODES L1
Int. type Gender of respondent.
```

```
1: Male
2: Female
```

```
*QUESTION 2 *NUMBER L2
What is your age?
```

```
*QUESTION 9999
These were all the questions. Thank you very much for your co-operation.
```

```
*END
```

In this example Question 1 is a precoded question, Question 2 is a numeric question and Question 9999 is a question without type, where only ENTER is required to continue.

### Example 2

```
*QUESTION 11 *CODES L1
```

```
*QUESTION 12 *NUMBER L2
What is your age ?
```

```
*IF [ Q12 <= 24 ] *COPY Q11 [ 1 ]
*IF [ Q12 >= 25 & Q12 < 35 ] *COPY Q11 [ 2 ]
*IF [ Q12 >= 35 & Q12 < 45 ] *COPY Q11 [ 3 ]
*IF [ Q12 >= 45 & Q12 < 55 ] *COPY Q11 [ 4 ]
*IF [ Q12 >= 55 & Q12 < 65 ] *COPY Q11 [ 5 ]
*IF [ Q12 >= 65 ] *COPY Q11 [ 6 ]
```

In this example Question 11 is a dummy question, but is created to be able to store the age in 6 distinct codes.

#### 6.53.1.1 See Also

*ALPHA.....	104
*CODES.....	114
*FORM .....	141
*NUMBER.....	201
*OPEN (question type) .....	203
*PAGE .....	209

## 6.55 \*QUOTA

### Purpose

Allows the quota in the Nfield Manager to be created automatically once the script with this command is uploaded and writes the answer to the question to the relevant \*SAMPLEDATA variables.

### Syntax

```
*QUOTA <sample_data_variable>
```

### Description

It will write the answer to the question to the relevant \*SAMPLEDATA variables, and will also create the quota definitions for these variables.

### Arguments

sample\_data\_variable

Sample data variable name, which will also be the used to create the quota variable name.

### Remarks

In the example script below, you can see that the \*QUOTA command has been used on Q1, Q3 and Q4. Just as \*SAVE, it will write the answers to the question to the relevant \*SAMPLEDATA variables. But it will do more: it will also create the quota definitions for these variables.

For example, the command on Q1 will create a quota variable named `Gender` with levels named 'male', 'female' and 'Rather not say'. It will also take into account if a question has the \*MULTI option and will then set the quota variable as a multi, provided the sample data variable has been defined as an array.

- The command cannot be used to create nesting quotas, nor can you set targets using it. Both of these will still have to be done manually through the Nfield Manager or through the public API.
- Unlike \*SAVE, \*QUOTA command is also executed when the \*INCLUDE command is targeting the related question. This makes question 3 still count towards quota, even though it is a dummy question.
- The \*QUOTA command will make Nfield Manager create a new quota definition or overwrite the existing one for each upload of the questionnaire script, unless the existing quota frame contains nesting quotas and/or any targets. As soon as the quota frame on a survey contains nesting quotas or targets it is locked, and cannot be changed through script upload any more. Removing targets and nesting will unlock the frame again.
- \*QUOTA always writes the label of the default language to the quota frame.
- Support for the quota command is not available in the Odin Developer yet. It will be included in the next release.

### Example 1

```
*TEMPLATE NfieldChicago
*SAMPLEDATA Gender, Age, Aided_Brandlist[10]
*QUESTION 1 *CODES 61L1 *ID Gender *QUOTA Gender
Gender?
1:male
```

```

2:female
3:Rather not say
*QUESTION 2 *number 62L3 *ID AgeNumber
How old (or young) are you?
*QUESTION 3 *CODES 65L1 *ID Agebrackets *QUOTA Age *DUMMY
1:0-17 *ID Bracket _0_17
2:18-40 *ID Bracket _18_40
3:41-65 *ID Bracket _41_65
4:66+ *ID Bracket _66_inf

*IF [Q{AgeNumber} < 18] *INCLUDE Q{Agebrackets}[{Bracket_0_17}]
*IF [Q{AgeNumber} > 17 & Q{AgeNumber} < 41] *INCLUDE Q{Agebrackets}[{Bracket_18_40}]
*IF [Q{AgeNumber} > 40 & Q{AgeNumber} < 66] *INCLUDE Q{Agebrackets}[{Bracket_41_65}]
*IF [Q{AgeNumber} > 65] *INCLUDE Q{Agebrackets}[{Bracket_66_inf}]

*QUESTION 4 *CODES 66L10 *MULTI *ID Aided_Awareness *QUOTA Aided_Brandlist
Do you know any of the below brands, even if by name only?
1:Brand-A
2:Brand-B
3:Brand-C
4:Brand-D
5:Brand-E
6:Brand-F
7:Brand-G
8:Brand-H
9:Brand-I
10:Brand-J

```

For more information, please watch [NIPO Academy 51a](#).

6.55.1.1 See Also

Quota ..... 67

## 6.56 \*RANDOM

### Purpose

Randomizes answer codes, repetitions or matrix statements.

### Syntax

\*RANDOM

### Description

This command can be used in combination with \*QUESTION, \*REPEAT and \*MATRIX and must be specified after these commands on the same line. The codes and the descriptions of a question are shown in random order on the screen. Repetitions are executed in random order.

### Arguments

pos

The data field specification where the used order is stored in the data file with the closed answers.

length

The length of the data field.

### Remarks

If you use \*RANDOM on a question the following applies:

- The number of codes of which the order is saved depends on the index variable used. To store the entire order, make sure that your index variable matches the number of codes used.
- With \*NOCON answer codes can be excluded from the randomization. \*NOCON excludes all remaining answer codes starting from the first answer code where it is specified.
- Also, with \*STOPRANDOM answer codes can be excluded from the randomization. Similarly, to \*NOCON, STOPRANDOM excludes all remaining answer codes starting from the first answer code where it is specified.

### Example

```
*QUESTION 1 *CODES 61L10 *MULTI *RANDOM
What PC makes do you know?

1: Acer
2: AST
3: Compaq
4: Dell
5: Hewlett Packard
6: IBM
7: Philips
8: Tulip
9: Other *OPEN *NOCON
10: Don't know *NMUL
```

In this example, all mentions are stored in position 61L10. The last two codes are not randomized along and will be displayed always at the bottom of the list.

6.56.1.1 See Also

*FORM .....	141
*GROUP.....	155
*NOCON .....	198
*ORDER .....	206
*REPEAT ... *ENDREP .....	230
*ROT .....	243
*STOPRANDOM.....	252

## 6.57 \*RANGE

### Purpose

Specifies a value range for numerical questions.

### Syntax

```
*RANGE <[n1|expression [TO n2|expression]]>[;n3 [TO n4];...]
```

### Description

This command specifies a range for numerical questions and numerical fields in a \*FORM question. You can specify a start value and an end value for the range by separating the two values with the keyword TO. Multiple values can be separated by a semicolon (;). Instead of a value, an expression may also be used.

### Remarks

You can only use an incremental range (1 TO 5 rather than 5 TO 1).

A semicolon must be used to separate two ranges. When using a comma, the system interprets this as a column-code reference.

The argument TO must be used to separate the extremes within a range. The dash (-) is interpreted as a minus sign and should not be used.

### Arguments

n1

The start value of the range.

n2

The end value of the range.

expression

An expression of which the result will be used as the start or end value.

### Example 1

```
*QUESTION 1 *NUMBER L2 *RANGE [ 18 TO 70 ; 99 ]
How old are you?

Int.: enter "99" for no answer.
```

In this example, the answer can be in the range of 18 to 70 and 99.

### Example 2

```
*QUESTION 1 *NUMBER L3 *RANGE [ -10 TO -1 ; 1 TO 10 ]
Specify a number in the ranges of -10 to -1 or 1 to 10.
```

In this example, the answer can be in the range of -10 to -1 and 1 to 10.

### Example 3

```
*VARS x,y
*PUT x [-3]
*PUT y [3]

*QUESTION 1 *NUMBER L2 *RANGE [x TO y]
Specify a number in the ranges of *?x to *?y.
```

In this example, the answer can be in the range of -3 to 3.

6.57.1.1 See Also

\*MAX..... 188

\*MIN..... 193

\*NUMBER..... 201



## 6.58 \*REC

### System

Nfield CAPI surveys only

### Purpose

Silently records (parts of) interviews.

### Syntax

```
*REC [[pos]L<length>|<pos>] ["Filename"] [::DELETE::] [::NOSAVE::] [QId]
```

### Description

With this command it is possible to record parts of an interview or even the whole interview. This command can be used at:

- The beginning of a line. When used at the beginning of a line a data field must be specified. This data field is used to storing the answers after coding. Specifying \*REC at the beginning of a line using a data field specification starts recording; specifying \*REC without a data field specification ends recording. Recording also automatically stops at the end of the script.

Recorded answers are saved to the mp3 files.

"Filename"

### Arguments

pos

This is the start of the data field where the data is written in the DAT-file.

length

This is the length of the data field.

::DELETE::

Deletes the current recording and all future recordings of the ongoing interview (it makes recording for the ongoing interview impossible, which is what is needed if the permission from the respondent to record the interview has not been granted). For more information on how it is done, please see section Silent Recording.

::NOSAVE::

Does not save the current recording.

QId

Question ID to map the silent recording in the Quality Control section with the question. It should be used at the start of the silent recording. This option is only available for CAPI surveys. For more information on how to use this, please visit the [CAPI course](#), section Quality Control/Quality Control Page.

### Example 1

```
*QUESTION 1 *CODES 648
We would like to record your answers. Do you agree?

1: Yes
2: No *GOTO 2
```

```
*REC 649L20

*QUESTION 2
Recording of the interview starts here.
```

### Example 2

```
*REC 648L20

*QUESTION 1 *CODES 668
Do you own a dog?

1: Yes
2: No

*QUESTION 2 *OPEN 669 *IF [ Q1 , 1 ]
What is the name of your dog?

*QUESTION 3 *CODES 670
Do you own a cat?

1: Yes
2: No

*REC
*QUESTION 4
This question is no longer recorded.
```

In this example, the answers on questions 1, 2 and 3 will be recorded and saved in the mp3 file.

#### 6.58.1.1 See Also

SILENT RECORDING ..... 341

## 6.59 \*REPEAT ... \*ENDREP

### Purpose

Start repetition block.

### Syntax

```
*REPEAT n
<commands>
*ENDREP
```

### Description

Defines the start of a repetition block. This command has to be on the beginning of a line. This command is always used in combination with \*ENDREP (end repetition block). A repetition block consists of a set of commands and questions that are considered to be a special component. When a repetition block is executed in the questionnaire, the system executes all commands within the repetition block as many times as is indicated by *n*. The answer fields within the repetition block are considered relative positions to the data field. When a repetition block is executed, the system uses the starting position defined by \*FIELD on the \*REPEAT statement as the starting point to determine where the answers have to be stored. For every repetition the answer fields belonging to the commands in the repetition block are stored in fixed fields one after another. It is possible to refer to the repetition number with the ?<sub>R</sub> operator.

### Arguments

*n*

This is a positive integer that gives the number of repetitions.

### Remarks

- You cannot jump out of a repetition block with the \*GOTO command. To terminate a repetition block, the \*END command can be used.
- Jumping from outside the repetition block into a question within a repetition block with the \*GOTO command is not possible.
- The repeat numbers in repeat blocks can be controlled by \*CONTROL, randomized by \*RANDOM, rotated by \*ROT, and ordered by \*ORDER.
- To refer to an answer of question *x* for the *n*th run of the loop from outside the loop, you can use the expression *QxSn* (see example 3).

### Example 1

```
*TEXTVARS paper
*QUESTION 2 *CODES 61L7 *MULTI
Which of the following newspapers do you know?

1: La Repubblica
2: La Stanza
3: The Mirror
4: The New York Times
5: Le Figaro
6: La Libération
7: None *NMUL
```

```
*REPEAT 6 *FIELD 68L6
*PUT paper Q2,?R

*QUESTION 3 *CODES 1 *IF [Q2,?R]
How often do you read *?paper?

1: Daily
2: Once a week
3: Once a month
4: Don't know

*ENDREP
```

In this example, question 3 is repeated 6 times. Only for those newspapers mentioned in Q2 the question is displayed.

### Example 2

```
*TEXTVARS paper
*QUESTION 2 *CODES 61L7 *MULTI
Which of the following newspapers do you know?

1: La Repubblica
2: La Stanza
3: The Mirror
4: The New York Times
5: Le Figaro
6: La Libération
7: None *NMUL

*REPEAT 6 *FIELD 68L6 *CONTROL Q2 W
*PUT paper Q2,?R

*QUESTION 3 *CODES 1
How often do you read *?paper?

1: Daily
2: Once a week
3: Once a month
4: Don't know

*ENDREP
```

This example is the same as the previous, but now the `*IF` statement has been replaced with a `*CONTROL` statement to the same effect.

### Example 3

```
*VARS LoopCounter
*REPEAT 5
*PUT LoopCounter [?R]

*QUESTION 1 *CODES L1
Answer for loop*?LoopCounter

1:First Answer
2:Second Answer

*ENDREP

*QUESTION 2 *CODES L1 *IF [Q1S2,1]
Under condition

1:ok
```

In this example, we are referencing the question inside the loop (Question 1) outside the loop (in Question 2). We want to show Question 2 only if we got an answer 1 ("First Answer") to

Question 1 during the second time the loop was run. The way we can do it is by using the `S` (expression operator, also used in `*MATRIX`). So `Q1S2` means the answer to `Q1` given during the second run of the loop.

**Please note** that in a `*REPEAT` the same question is used multiple times, which implies that there are multiple instances of the same question id. If you refer to a question used in a `*REPEAT` inside that `*REPEAT`, it will refer to question in the current iteration of that `*REPEAT`. Referring to a question used in a repeat outside of the `*REPEAT` will result in *undefined result*.

#### 6.59.1.1 See Also

Repeat Number .....	45
*CONTROL.....	116
*END .....	126
*FIELD.....	134
*MATRIX .....	182
*ORDER .....	206
*RANDOM.....	221
*ROT .....	243

## 6.60 \*REQUEST

### Purpose

Online surveys only.

To perform a HTTP request without any restrictions, including a strict response model.

### Syntax

```
*REQUEST <ResultCode> <Result> <RequestName>
```

### Arguments

ResultCode

A variable to put the result in. Result codes can be one of the following:

ResultCode	Description
1	A success status code.
-1	Timeout, the API did not respond within the time limit of 5 seconds
-2	You forgot to add a request configuration.
-5	The response was not a success code. If any content is returned, it will be in the Content property of the result.
-6	An exception occurred during the request. If any content is returned, it will be in the Content property of the result.
-7	An unknown error has occurred. If any content is returned, it will be in the Content property of the result.

Result

A result variable to put the http response content in, this must be a textvar.

RequestName

The name of the request configuration. Before we can use the \*REQUEST we have to configure headers, http method, endpoint, etc. This is done using the Request configuration object.

In the configuration you can define variables to be passed through the script. These variables are placed between brackets {}, and will be replaced with the content of that variable when executed. You can use variable in the URI or the body, but not in the headers.

Below a snippet of ODIN script that populates that SearchTerm variable, and then performs a request.

The request has a template like this:

```
{"SearchTerm": "{SearchTerm}"}
```

The request in ODIN script:

```
*VARS RequestResult
*TEXTVARS SearchTerm, RequestResponse

*PUT SearchTerm 'Odin is awesome'

*REQUEST RequestResult RequestResponse 'RequestName'
```

And the payload will be send to the endpoint will be

```
{
  "SearchTerm": "Odin is awesome"
}
```

The response will be saved in RequestResponse. This can be anything you want. The content can also be a string, a number, or a GUID.

### Configure the request

Before using *\*REQUEST*, we need to configure some things for it: headers, http method, endpoint, etc. This done using the Request configuration object which can be setup through the Nfield UI.

As a Domain Administrator, Power User, or Local Domain Manager it is possible to create API Request configurations that scripters will be able to use in their ODIN scripts with the *\*REQUEST* command. This is done on the APIs tab by (1) clicking on the *New Request* button, and following the steps below:

2

## General configuration

Select an appropriate request method.



GET

URI\*

The endpoint (URI) of the request.

Send

General

Headers

Test Values

Name\*

The name of the request as used in the \*REQUEST command in the ODIN script.

Description

Optional description of the request.

Help URI Optional help URI towards extensive documentation for the request.

- The `Name` is mandatory, and must be unique (per domain), and not be longer than 16 characters.
- The `Uri` is mandatory and cannot be longer than 1024 characters.
- The `HelpUri` cannot be longer than 1024 characters.
- The `Description` cannot be longer than 4096 characters.
- There are no limitations on the Header `Name` or `Value` properties.

3

## Headers configuration

GET

URI\*

The endpoint (URI) of the request.

Send

Header name

Header value

General

Headers

Test Values

Authorization

JSON Web Token



+ Add Header



- Click **"Add Header"** to add Headers.
- You can add any number of Headers by clicking **"Add Header"**.

4

## POST / PUT / PATCH configuration

For POST/PUT/PATCH request methods, you can configure the body of the request.

↓

POST URI\* [The endpoint \(URI\) of the request.](#) Send

General Headers **Body** Test Values

Body of the request

```
{
  "By default, the body of the request is expected in JSON format."
}
```

After configuring the endpoint, please always make sure to test it:

5

## Test Values configuration

GET URI\* <https://app.zipcodebase.com/api/v1/search?codes=90210&{Country}> Send

General Headers **Test Values**

Key	Value
Country	US

+ Add Test Value

- Click **"Add Test Value"** to add test variable values.
- You can add any number of test values by clicking **"Add Test Value"**.
- These test values are only used to test the endpoint setup, not for real interviews.

6

## Testing the endpoint

Example

GET

General Headers Test Values

+ Add Test Value

Click "Send" to test the endpoint.

Response Status: OK Size: 51 bytes Time: 251 ms

```
{
  "query": {
    "codes": [
      "90210"
    ],
    "country": "US"
  },
  "results": {
    "90210": {
      "postal_code": "90210",
      "country_code": "US",
      "latitude": "34.09010000",
      "longitude": "-118.40650000",
      "city": "Beverly Hills",
      "state": "California",
      "state_code": "CA",
      "province": "Los Angeles",
      "province_code": "037"
    }
  }
}
```

Example of Output

- The right-side pane is used to see the Output.
- By default, the output is in JSON format.

Here is an example of the ODIN script you can use to extract certain values from the response JSON and put them into variables.

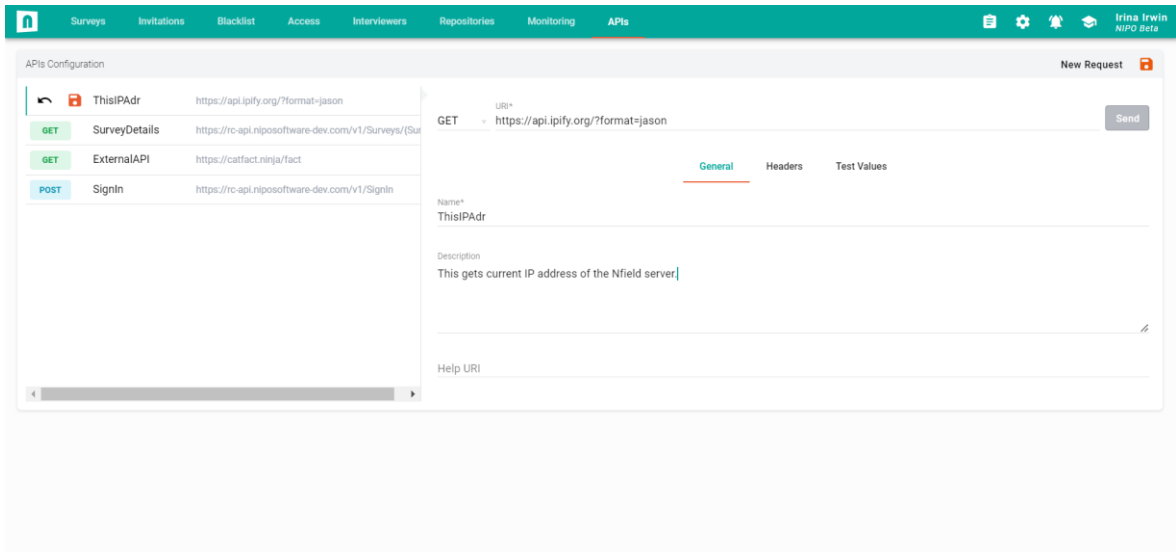
JSON response	INTO	ODIN script
<pre>{   "ResultCount": "3",   "Results": [     { "id": 1254378,       "Name": "Value1"     },     { "id": 894456,       "Name": "Value2"     },     { "id": 789125,       "Name": "Value3"     }   ],   "Metadata": {     "id": "9033fe8b-9e6c-4a5c-a28b-74206122b81c",     "Duration": 84   } }</pre>		<pre>*VARS Result, Count, Duration *TEXTVARS ResponseId, RequestResponse *REQUEST Result RequestResponse 'RequestName'  *PUT Count [?JSON(RequestResponse, 'ResultCount')] *PUT Duration [?JSON(RequestResponse, 'Metadata.Duration')] *PUT ResponseId [?JSON(RequestResponse, 'Metadata.Id')]  *TEXTVARS ResultName[10], TempText *VARS REPNUM *REPEAT 10  *IF [?R&gt;Count] *END *PUT REPNUM [?R-1] *PUT TempText "Results[*?REPNUM].Name" *PUT ResultName[?R] [?JSON(RequestResponse, TempText)] *ENDREP</pre>

## An example

We will create a request that will allow scripters to get a current IP address of the Nfield server (note that Nfield servers do not use static IP addresses. They have dynamic ones, so they will not be the same every time).

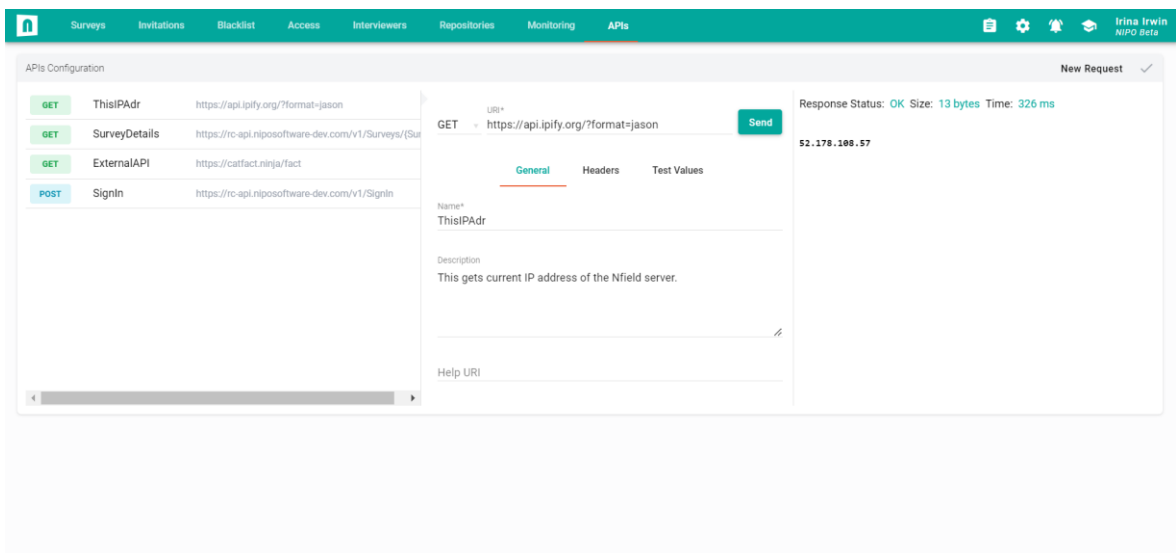
In the Nfield Manager, on the APIs page, click on *New Request*, then enter the following URI for a GET *https://api.ipify.org/?format=json*

Name should be the same as we will use in the script. In our example it is *ThisIPAdr*.  
For the optional description, please enter: “This gets current IP address of the Nfield server.”



Then click on the orange save icon.

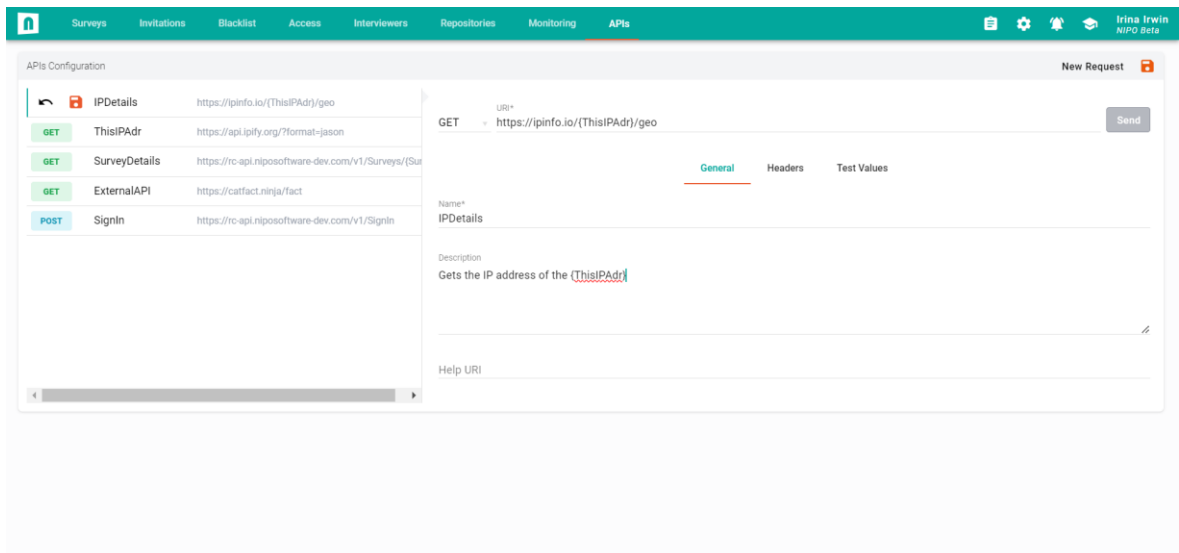
We now have a request that we can test by clicking on the *Send* button. This will return the current IP address of the Nfield server you are on. This is the result in my case:



The value you will see will of course be different. Please save that value somewhere since we will use it later in this example.

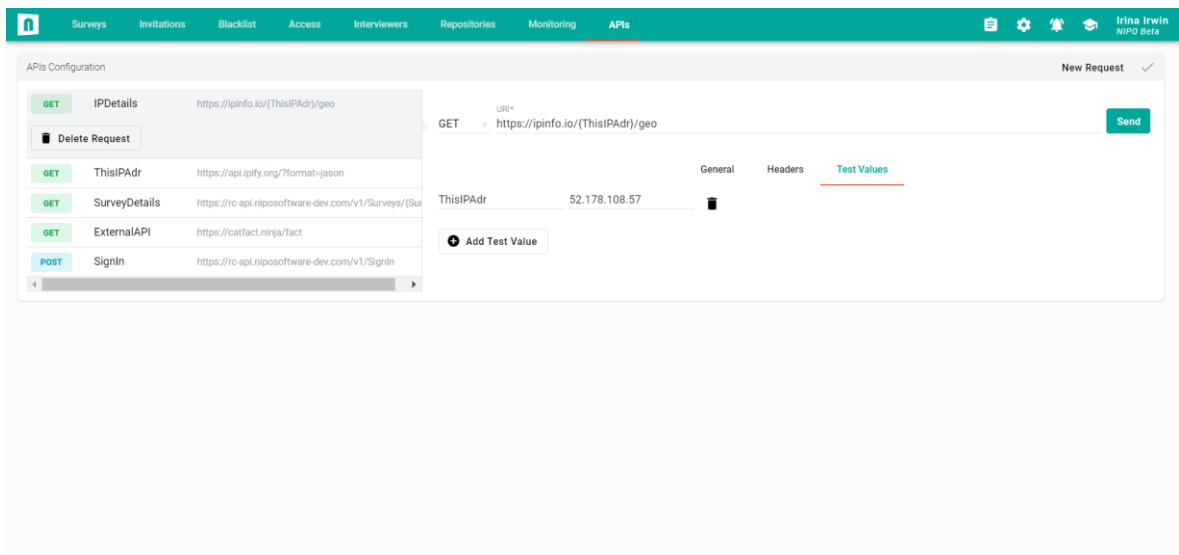
Let's make another request. It's called IPDetails. Click on the *New Request* button again, and enter the following into the GET URI: <https://ipinfo.io/{ThisIPAdr}/geo> *ThisIPAdr* is a variable in the script that will be filled by the script with the IP address that we want to get the details from.

The name used in the script is *IPDetails*. In the optional description you can put: “Gets the IP address of the {ThisIPAdr}”.

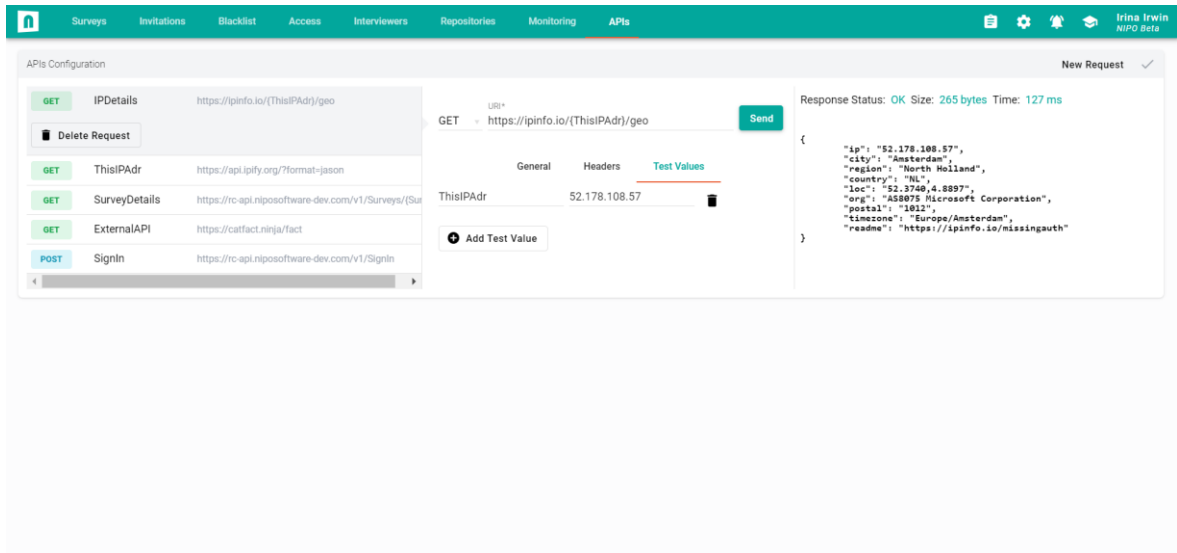


Save it.

To test this request we need to supply an IP address value to get details from. So click on the *Test Values*, then on *Add Test Value*, and enter *ThisIPAdr*, and then a valid IP address (for example, the one you saved when testing ThisIPAdr request):



Click on Send, and that will get the details of this IP address. Here is what I got:



Let us now use these requests in a script.

We will now create a new survey (for example, an Online survey called NfieldServer) and upload the following script to it:

```
*TEMPLATE NfieldChicago
*VARS RequestResult, IpLength
*TEXTVARS ThisIPAdr, IPInfo

*PAGE
This is using an API call to fetch the IP address of the Nfield server.
It then issues a second API call using the IP address from the previous step to fetch information from that IP address.

*REQUEST RequestResult ThisIPAdr 'NfieldServerIP'

** Here we use a JSON operator to extract the IP address from the JSON string that we now
** have in ThisIPAdr. See the list of operators in section Expression Operators of the
** Nfield Reference.
*PUT ThisIpAdr [?JSON(ThisIpAdr,"ip")]

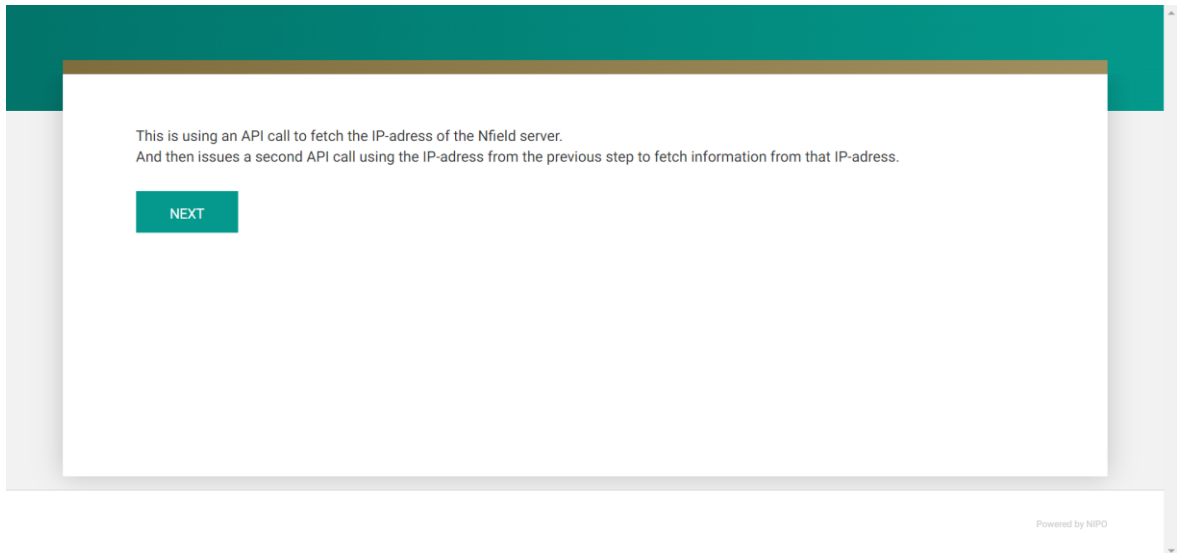
** Get the details of our IP address
*REQUEST RequestResult IPInfo 'IPDetails'

*PAGE

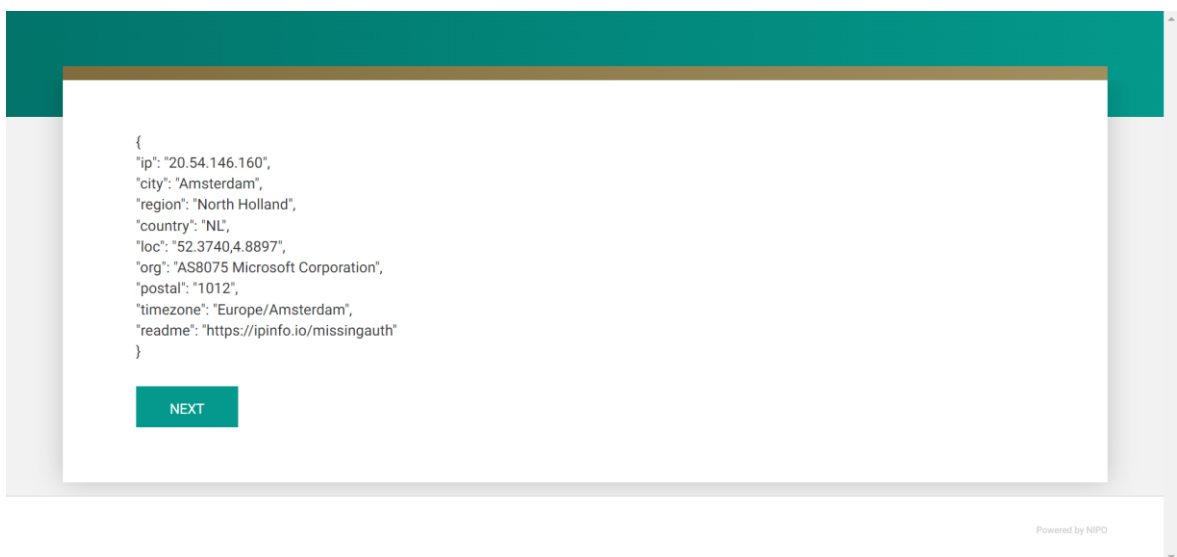
*?IPInfo

*END
```

Let's run the interview:



Click Next, and you get the current IP address and the details of you current Nfield server:



**IMPORTANT:** Please note that scripters are **OBLIGATED** to handle the request results in their script -- all the values the request gets, including the possible errors. Otherwise, if you get problems during fieldwork with your request, your helpdesk will **NOT** be able to figure out what happened. There is no way to tell if you don't handle those result values, so you should handle any errors that the request might get in your script.

For example, if you get -1 (time-out) as error, you script will need to decide what happens: to fire the \*REQUEST again or to suspend the survey and ask your respondent to try again later. If you don't handle this error, the survey will continue with the next question and an empty result for your request.

**For more information** on *\*REQUEST* command please see our [NIPO Academy #48](#) on this subject.

## 6.61 \*RETURN

### Purpose

Returns from subroutine.

### Syntax

\*RETURN

### Description

Returns from a subroutine before the physical end of the subroutine is reached. This command is also allowed under condition.

### Example

```
*TEXTVARS PAPER, OTHERPAPER

*SUBROUTINE "NEWSPAPER"
*QUESTION 101 *NUMBER L1 *MAX 6

    How many of the last 6 issues of *?PAPER did you read?

*IF [Q101 = 0] *RETURN
*QUESTION 102 *CODES L1
    Did you read *?PAPER yesterday?

    1: Yes
    2: No

*ENDSUB

*QUESTION 1 *CODES L8 *MULTI *SAVE PAPER
Which newspapers do you read?

    1: La Repubblica *GOSUB NEWSPAPER
    2: La Stanza *GOSUB NEWSPAPER
    3: The Mirror *GOSUB NEWSPAPER
    4: The New York Times *GOSUB NEWSPAPER
    5: Le Figaro *GOSUB NEWSPAPER
    6: La Libération *GOSUB NEWSPAPER
    7: El Pais *GOSUB NEWSPAPER
    8: Other *OPEN *GOSUB NEWSPAPER
```

#### 6.61.1.1 See Also

\*GOSUB.....151  
 \*SUBROUTINE ... \*ENDSUB .....260

## 6.62 \*ROT

### Purpose

Displays answer categories rotated.

### Syntax

\*ROT

### Description

This command is always used in combination with a \*CODES question, or \*REPEAT or \*MATRIX blocks and must be on the same line as the question definition. The codes and their labels are shown in rotated order on the screen, starting randomly from any of the available answer codes. Then the next codes follow in sequence, up to and including the highest code. Next the lowest codes follow up to code where it randomly started.

When you do a rotation, it chooses a random starting point, and from that point onwards it shows the rest of the list in rotated order.

### Arguments

pos

The data field specification where the order of the displayed categories is written in the closed answer file.

length

The length of the data field.

### Remarks

- With \*NOCON answer codes can be excluded from the rotation. \*NOCON excludes all remaining answer codes starting from the first answer code where it is specified.

### Example

```
*QUESTION 1 *CODES 61L10 *MULTI *ROT 71L8
Which car brands do you know?

1: BMW
2: Audi
3: Kia
4: Toyota
5: Honda
6: Nissan
7: Skoda
8: Peugeot
9: Other *OPEN *NOCON
10: Don't know *NMUL
```

In this example, all mentions are stored in position 61L10. In the answer field behind \*ROT the codes for the first 4 codes on the screen are stored. The last two codes are not rotated along and are always displayed at the bottom of the code list.

6.62.1.1 See Also

*GROUP .....	155
*NOCON .....	198
*ORDER .....	206
*RANDOM.....	221
*REPEAT ... *ENDREP .....	230

## 6.63 \*SAMPLEDATA

### Purpose

With this command variables in the NIPO ODIN questionnaire can be linked automatically with field names in the sample table, and the quota variables, URL parameters, or fieldnames in the CAPI address.

### Syntax

```
*SAMPLEDATA var1, var2, ...
```

### Description

The variable in the NIPO ODIN questionnaire is filled automatically with the contents of the matching database field when starting the interview. The fields in the database table are updated with the contents of the variable when closing the interview for a final (non-)response or an appointment. In every other aspect, a sample data variable works just like a regular NIPO ODIN text variable. Maximum character length of a \*SAMPLEDATA value is 1500.

You use \*SAMPLEDATA to refer to the parameters in a URL, both for the starting link and the exit link. For more information, please see the [Exit Links](#) chapter.

### Example 1

```
*SAMPLEDATA Region
*PAGE
According to our database the region is *?Region.
```

Assumes that the sample table for survey A1234 contains a field called `Region`.

### Example 2

```
*SAMPLEDATA Gender
*QUESTION 7 *CODES 91 *SAVE Gender
Gender?

1: Man
2: Woman
```

Saves the label text ('Man' or 'Woman') directly in the sample table field `Gender` (requires the field to exist).

### 6.63.1.1 See Also

```
*PUT.....217
*TEXTVARS.....268
*VARS .....286
```

## 6.64 \*SAVE (question option)

### Purpose

Saves the answer of a question in a variable.

### Syntax

```
*SAVE <var | array>
```

### Description

This command is always used in combination with `*QUESTION` and must be specified after this command on the same line. The answer on a question is saved in a `*VARS`, `*TEXTVARS` or `*SAMPLEDATA` variable. If it is a numeric variable the code value or the number is saved. If it is a text variable with a closed question with precoded answers the label of the code is saved. Saving an answer text is particularly of use if you want to insert this text in a subsequent question (dynamic text modification).

May not be used in a single question in combination with `*SAVE` as a code option. If used on a `*MULTI` question, it saves the lowest selected code (text) only.

### Remark

`*SAVE` as a question option is ignored if the question also uses `*SAVE` as a codes option.

### Arguments

`var`

Any (alpha)-numerical variable or any of the ten system variables. If the argument is omitted system variable 0 is used by default.

`Array`

Array of variables.

### Example 1

```
*TEXTVARS PARTY
*QUESTION 5 *CODES 81 *SAVE PARTY
Which party will you vote for in the next election?

1: Conservative union
2: Social democrats
3: Liberals
4: Greens
5: Socialists

6: Other party *OPEN
7: None *GOTO 7

*QUESTION 6 *CODES 82
Did you also vote for the *? PARTY in the previous election?

1: Yes
2: No

*QUESTION 7
...
```

### Example 2

```
*TEXTVARS Mydata

*QUESTION 1 *CODES 61L5 *MULTI *SAVE Mydata
Please pick some letters
1:A
2:B
3:C
4:D
5:E

*PAGE
*?Mydata
```

Let's say the user selects options 2, 3 and 4 – it will show “B”, which is the text of the lowest selected code.

#### 6.64.1.1 See Also

\*PUT.....217  
 \*SAVE (codes option).....249

## 6.65 \*SAVE (codes option)

### Purpose

Saves a code number or label in a variable.

### Syntax

\*SAVE <var>

### Description

This command is always used after a code definition and must be specified after this code on the same line. If it is a numeric variable the code value is saved as answer. If it is a text variable the descriptive text of the answer category is saved. If it is an open code the open-ended answer text is saved.

### Remark

If \*SAVE is used as a codes option, \*SAVE as a question option on the same question is ignored.

### Arguments

var

Any (alpha)-numerical variable or any of the ten system variables. If the argument is omitted system variable 0 is used by default.

### Example

```
*TEXTVARS localpaper

*QUESTION 1 *CODES L7 *MULTI
To what newspapers do you subscribe?

1: New York Times
2: The Guardian
3: Sunday Times
4: Washington Post
5: Herald Tribune
6: A local paper *OPEN *SAVE localpaper

7: Don't know *NMUL

*QUESTION 2 *CODES L1 *IF [Q1,6]
Do you read *? localpaper frequently?

1: Yes
2: No
```

#### 6.65.1.1 See Also

\*SAVE (question option) ..... 247

## 6.66 \*SHOWDOCUMENT

### Purpose

To create a Help button for your interviews.

### Description

When clicked, the Help button opens a PDF file and after reading it, the interviewer returns to the questionnaire again. The behavior is the same for CAPI and Online interviews.

### Remarks

- Available for Nfield CAPI and Online surveys.
- This feature is available for versions of the Nfield CAPI app v1.116.000 and higher.
- The PDF should be uploaded in the "Media Files" section of the questionnaire settings.
- Standard rendering of the help button is in front of the question text. It is also possible to render it as a button in the navigation bar using NfieldChicago template (see example).
- **Please make sure to test this functionality before deploying it to your fieldwork force as we have noticed that not all the PDF readers installed on the interviewing devices behave the same. If you have a wrong PDF reader installed, your interviewer might not be able to resume the interview. We recommend installing one version of PDF reader with clear instructions to your interviewers.**

### Syntax

```
*QUESTION <n>|X *SHOWDOCUMENT
```

```
"<name_of_pdf_help_file>;text=<text_to_display_on_button>"
```

### Arguments

name\_of\_pdf\_help\_file

Name of the pdf help file that will show up after the Help button is clicked.

text\_to\_display\_on\_button

Text to display on the Help button.

### Example

```
*TEMPLATE "NfieldChicago"

*QUESTION 1 *CODES 61L1 *SHOWDOCUMENT "data.pdf;text=Help page (pdf)"
Can you tell us your household income?

1:Under $30,000
2:$30,001 to $50,000
3:$50,001 to $75,000
4:$75,001 to $100,000
5:$100,001 or over

*END
```

### Result

This produces a question that looks like this (note that "Help page (pdf)" is a clickable link):

## Help page (pdf) Can you tell us your household income?

Under \$30,000
\$30,001 to \$50,000
\$50,001 to \$75,000
\$75,001 to \$100,000
\$100,001 or over

With the use of a simple theme it is possible to move and transform this help link into a proper button, and to place it at the bottom of the page, like in the example below:

## Can you tell us your household income?

Under \$30,000
\$30,001 to \$50,000
\$50,001 to \$75,000
\$75,001 to \$100,000
\$100,001 or over

How to do that can be found in [this section of the NfieldChicago documentation](#).

### 6.66.1.1 See Also

\*QUESTION .....220

## 6.67 \*SORT and \*STOPSORT

### Purpose

To order questions and matrixes alphabetically.

### Description

\*SORT can be used in combination with a \*QUESTION command or a \*MATRIX command to sort the answer categories or statements alphabetically. Categories will be sorted in ascending order according to the language and country rules as specified in the `languagecode-countrycode` parameter.

To stop the alphabetical sort, you can use \*NOCON on an answer category. That answer category and all following answer categories will not be sorted and instead will keep the order as they are in the script. \*NOCON will also exclude those items from other sorting commands and the \*CONTROL command. Unlike \*NOCON, the \*STOPSORT command only stops the alphabetical sorting (similarly to \*STOPRANDOM on \*RANDOM).

\*SORT cannot be used together with other sorting commands, like \*RANDOM and \*ORDER. As with those commands, inserting a line break in the list of answer categories will split the list into groups, which will then be sorted separately.

### Syntax

```
*SORT <languagecode-countrycode>
```

### Arguments

`languagecode-countrycode`

Language and country code. This parameter is mandatory and will be validated against [.NET culture codes](#). If the parameter consists of a code which is not valid or not supported, Nfield will use the default sorting language, which is **en-us (US English)**.

**Note:** If you are using \*SWILANG command to allow the respondent to change languages, you will want those languages to apply to your \*SORT commands as well. To do this, you need to add the culture code as an argument to the \*LANGUAGE command in your language section(s).

For example:

```
*LANGUAGE "Danish,LTR,da-DK"
```

The first element of this argument remains fully customizable (you can name the language any way you name it in the script), the second and third elements are the text direction (LTR or RTL) and culture code (da-DK). These are optional. If you leave them out, Nfield will revert to the default options (left-to-right and en-US respectively). If you include both text direction and culture code,

then the culture code *must* come after the text direction (so "Danish, da-DK, LTR" will not work).

**Note 2:** NIPO ODIN Developer version 5.18.0.29 (or lower) does not support this command.

#### Example 1 Simple category question

```
*TEMPLATE "NfieldChicago"

*LIST Countries
1: USA
2: France
3: India
4: China
5: South Africa
6: Other, namely *OPEN *STOPSORT

*LIST standard_buttons
98: Don't know
99: None of the above

*QUESTION 20 *CODES 65L24 *SORT en-us *MULTI *USEBUTTONS 65L1 "standard_buttons"
Which countries have you visited?
*USELIST "Countries "
```

#### Example 2 Matrix question

```
*TEMPLATE "NfieldChicago"

*QUESTION 30 *CODES 90L2 *DUMMY
*USELIST "Countries"

*MATRIX 12 Q30 *FIELD 92L15 *SORT en-us
For each of these countries, please choose which animal you associate it with.

*QUESTION 40 *CODES 1L1 *SORT en-us
1: Quetzal
2: Alpaca
3: Zebra
4: Phoenix
5: Axolotl
6: Dodo *NOCON
*ENDMATRIX
```

6.67.1.1 See Also

*MATRIX .....	182
*ORDER .....	206
*RANDOM.....	221
*SWILANG .....	262
*QUESTION .....	220

## 6.68 \*SPLITSTRING

### Purpose

To split a string into multiple parts based on specified delimiters. This command is used in scripts to process strings and store the results for further use. It is useful in scenarios where you need to break down a string into manageable parts.

### Syntax

```
*SPLITSTRING <resultCount> <resultString> <stringToSplit> <delimiter>
```

### Arguments

`resultCount`

The variable where the count of split parts will be stored, one of:

- variable
- array variable (count goes to element specified by index)

`resultString`

The variable where the split parts will be stored, must be an array text variable.

`stringToSplit`

The string that needs to be split, one of:

- "text" (\*? allowed)
- text variable
- array text variable (specify by index)

`delimiter`

The delimiter(s) used to split the string. Multiple delimiters can be specified, one of:

- "text" (\*? allowed)
- text variable
- array text variable
- array text variable (by index)

### Example 1 Basic split

In below example user input sentence is split by ',' (comma) character.

```
*TEMPLATE "NfieldChicago"

*VARS ResultCount
*TEXTVARS Result[10],Input

*QUESTION 10 *OPEN 61L100 *SAVE Input
Please list up to 10 fruits you have had for breakfast in the last month. Separate each item with a comma.
*SPLITSTRING ResultCount Result Input ","
*PAGE

Input: *?Input
ResultCount: *?ResultCount
Result:
*?Result[1]
*?Result[2]
*?Result[3]
```

**Example 2 Multiple delimiters**

```

*TEMPLATE "NfieldChicago"

*VARS ResultCount
*TEXTVARS Result[10],Input,Delimiters[3]

*PUT Delimiters[1] ','
*PUT Delimiters[2] '.'
*PUT Delimiters[3] '|'

*PUT Input 'I need to split. Not only by comma, but also by dot | and by pipe as well'

*SPLITSTRING ResultCount Result Input Delimiters
*PAGE
Input: *?Input
ResultCount: *?ResultCount
Result:
*?Result[1]
*?Result[2]
*?Result[3]

```

In this case the sentence is split by 3 characters, example sentence will result in such split:

ResultCount : 4 Result :

- [1][I need to split]
- [2][ Not only by comma]
- [3][ but also by dot ]
- [4][ and by pipe as well]
- [5][
- [6][
- ...

**Example 3 Count greater then declared result array**

```

*TEMPLATE "NfieldChicago"

*VARS ResultCount
*TEXTVARS Result[3], Input

*PUT Input 'apple,banana,orange,grape,watermelon'

*SPLITSTRING ResultCount Result Input ","
*PAGE
Input: *?Input
ResultCount: *?ResultCount
Result:
*?Result[1]
*?Result[2]
*?Result[3]

```

In this example, the input string contains five items separated by commas, but the `Result` array is only declared to hold three items. The `*SPLITSTRING` command will split the input string and store the first three parts in the `Result` array, while the `ResultCount` will reflect the total number of parts (which is 5 in this case).

**Remarks**

1. The `<stringToSplit>` and `<delimiter>` parameters cannot be empty.
2. Delimiters are case-sensitive.
3. The `ResultCount` parameter will always show the total number of strings generated by the command, even if the `Result` parameter defined to store less than that number. (So, if in the above example 1, the respondent enters not 10 but 12 fruits, `ResultCount` will show 12, but the last 2 strings will not be stored as a `Result`, since it has been defined to only hold 10 items).

**Note:** Current NIPO ODIN Developer does not support this command yet but will support it in a future release.

6.68.1.1 See Also

String manipulation routines ..... 53

## 6.69 \*STOPRANDOM

### Purpose

Specifies that the code and all following codes in a question do not come under randomization.

### Syntax

\*STOPRANDOM

### Description

This command is always used behind a code definition and must be specified after this code on the same line. This command is used for the question options \*RANDOM. It excludes all codes starting from the code that have \* STOPRANDOM as option from randomization.

### Example

```
*QUESTION 1 *CODES 61L10 *MULTI *RANDOM
What PC brands do you know?

1: Acer
2: AST
3: Compaq
4: Dell
5: Hewlett Packard
6: IBM
7: Philips
8: Tulip

9: Other *OPEN
10: Don't know any makes *NMUL *STOPRANDOM
```

In this example, code 10 in question 2 is always displayed last.

#### 6.69.1.1 See Also

\*RANDOM..... 221

## 6.70 \*STRAT

### Purpose

Stratification on output.

### Syntax

```
*STRAT <n|[expression]>
```

### Description

With this command a check can be made to see if a certain stratum has reached its limit. The stratification criteria are put in the sample table record during the interview, so-called stratification on output. If the stratum limit is reached, the questionnaire is continued with the question with the question number indicated by the argument.

### Arguments

n|expression

A positive integer that specifies an existing question number.

### Example

```
*SAMPLEDATA OwnsVCR
*QUESTION 2 *CODES 61 *SAVE OwnsVCR
Do you own a VCR?

1: Yes
2: No

*STRAT 99

*QUESTION 3 *CODES 62
Did you use your VCR yesterday?

1: Yes
2: No

*QUESTION 4
Thank you for your co-operation.

*END

*QUESTION 99
I don't have any more questions for you.

Thank you for your co-operation.

*ENDNGB
```

In this example, the answer of question 2 is transferred to the sample table record and the stratification levels are checked to see if the quota was reached. If so, a jump to question 99 is made.

#### 6.70.1.1 See Also

```
*END ..... 126
*ENDNGB..... 128
*ENDST ..... 131
```

## 6.71 \*SUBROUTINE ... \*ENDSUB

### Purpose

Defines a subroutine.

### Syntax

```
*SUBROUTINE name
<commands>
*ENDSUB
```

### Description

Defines the start of a subroutine. This command has to be at the beginning of a line. This command is always used in combination with \*ENDSUB (end subroutine). The subroutine can be jumped to at any time with \*GOSUB.

A subroutine consists of a set of commands and questions that the system considers as a special component. When a subroutine is called in the questionnaire the system executes the commands in the subroutine. It is as if the commands in the subroutine are inserted where the subroutine was called. The answer fields in a subroutine will be considered relative.

When a subroutine is called the system uses the position in the \*FIELD parameter belonging to \*GOSUB as the starting point to determine where the answers have to be put. A subroutine is considered a separate component of the questionnaire. It should be placed somewhere in the questionnaire before the first call to the subroutine.

In a subroutine you can call subroutines recursively.

Subroutines can also be called after a \*NEW command, when placed in the first sub-questionnaires.

### Arguments

name

The name of the subroutine.

### Remarks

It is not allowed to jump out of a subroutine by means of the \*GOTO command. To end subroutine execution, use \*RETURN instead.

### Example

```
*TEXTVARS BRAND

*SUBROUTINE OPINION
*QUESTION 1 *CODES L1
What do you think of the service of *? BRAND?

1: Very good
2: Good
3: Poor
4: Very poor

*ENDSUB

*QUESTION 2 *CODES L5 *MULTI
Which of the following gas stations have you ever visited?

1: Esso      *PUT BRAND "Esso" *GOSUB OPINION
2: Shell    *PUT BRAND "Shell" *GOSUB OPINION
3: Texaco   *PUT BRAND "Texaco" *GOSUB OPINION
```

4: BP	*PUT BRAND "BP" *GOSUB OPINION
5: Mobil	*PUT BRAND "Mobil" *GOSUB OPINION

#### 6.71.1.1 See Also

\*FIELD..... 134  
 \*GOSUB..... 151  
 \*RETURN ..... 243

## 6.72 \*SWILANG

### Purpose

Switches to another (predefined) language.

### Syntax

```
*SWILANG "[name]"
```

### Description

With this command it is possible to switch to another language during the interview. The name in the syntax has to match the name of the language section.

### Arguments

name

This is the name of the language section that NIPO ODIN refers to. To switch to the default language you can omit a name.

### Remark

- The double quotes around the section name are mandatory.
- In the documentation for the Nfield [Chicago](#) template, there is also description on how enable the users to be able to switch language anytime in the questionnaire. There is also a useful blog about this feature [here](#).

### Example

```
*QUESTION 1 *CODES 201
Interviewer: choose a language

1: Dutch *SWILANG "Dutch"
2: German *SWILANG "German"
3: French *SWILANG "French"
4: English *SWILANG ""

*QUESTION 2 *CODES 202
Do you have a dog?

1: Yes
2: No

*END

*LANGUAGE "Dutch"
*QUESTION 2
Heeft u een hond?

1: Ja
2: Nee

*LANGUAGE "German"
*QUESTION 2
Haben Sie einen Hund?

1: Ja
2: Nein

*LANGUAGE "French"
*QUESTION 2
Avez vous un chien?
```

1: Oui  
2: Non

6.72.1.1 See Also

\*LANGUAGE..... 169

<https://nfieldchicago.nfieldmr.com/accessibility/page-language/index.html>

## 6.73 \*TABLE

### Purpose

Enables export information for hierarchical data for the NIPO DSC for IBM SPSS, used by the IBM SPSS Data Collection Data Model as well as to the NIPO data model as used in the data repositories.

It also propagates the properties of codes of the controlling question of \*MATRIX or a \*REPEAT to the corresponding questions.

### Syntax

```
*TABLE ["<name> description"]
```

### Description

This command influences the NIPO Diana / Nvision Script export for use with the NIPO DSC for IBM SPSS. It adds a comment line (COM) to the variable definition which tells the NIPO DSC for IBM SPSS interface that the \*FORM or \*MATRIX questions, or \*REPEAT block must be used for a hierarchical data representation.

The \*TABLE command is required to have the data reflected from \*REPEATS and \*MATRIXES in the data repositories.

The \*TABLE command does not affect The COM line is ignored in NIPO Diana / Nvision Script.

This command can be used in two ways:

1. On a \*FORM question definition no label needs to be specified, as the block name is taken from the variable name or the variable \*LABEL. Hierarchical data of the fields of a \*FORM question can only be created if all fields in the \*FORM question are of the same type and size (for example, all \*NUMBERS of size 2, et cetera).
2. Within a \*REPEAT block the \*TABLE command is specified in advance of the first question in the repeat block and defines a name for the current iteration.
3. Within a \*MATRIX, the \*TABLE command is specified in on the same line as \*MATRIX command.

### Arguments

```
"<name> description"
```

For use in \*REPEAT blocks only. The name part of the text is required and consists of a single word that sets the name of the hierarchical block of data. The description contains a descriptive text that is different for each iteration within the \*REPEAT block. Change this text within a repeat block by displaying variable content using \*? (example below).

#### Example 1: \*REPEAT block with two questions

```
*TEMPLATE NfieldChicago
*TEXTVARS Brand

*REPEAT 4 *FIELD 101L24
*IF [?R = 1] *PUT Brand "Grolsch"
*IF [?R = 2] *PUT Brand "Hertog Jan"
*IF [?R = 3] *PUT Brand "Amstel"
*IF [?R = 4] *PUT Brand "Leeuw"

*TABLE "Beer *? Brand"
```

```

*QUESTION 5 *CODES 1L1
For *? Brand, please mark your overall taste experience:

1: Good
2: Above average
3: Average
4: Below average
5: Bad

*QUESTION 6 *FORM *TABLE *LABEL "Consumption"
Your average consumption of *? Brand on a weekly basis:

1: Glasses: *NUMBER 3L2 *NON
2: Bottles: *NUMBER 5L2 *NON

*ENDREP

```

This creates a comment line that tells the NIPO DSC for IBM SPSS to lay out the repeat block in a hierarchical data structure, where the block is named "Beer" and the nested brand block names are specified in the `*IF` lines. Inside each brand block there is a nested "Consumption" block.

#### Example 2: \*TABLE with \*MATRIX

```

** Use the NfieldChicago template
*TEMPLATE "NfieldChicago"

*QUESTION 10 *CODES 61L1 *DUMMY
1:BrandA
2:BrandB
3:BrandC
4:BrandD
5:BrandE
6:BrandF

*MATRIX 6 Q10 *FIELD 62L6 *TABLE "UsedBrandsMatrix"
Have you ever used this brand?

*QUESTION 20 *CODES 1L1 *VAR "UsedBrands"
1:Yes
2:No
3:I can't remember

*ENDMATRIX

```

Similarly, to `*REPEAT` example, this `*MATRIX` with `*TABLE` example creates a comment line that tells the NIPO DSC for IBM SPSS to lay out the matrix block in a hierarchical data structure, where the block is named " UsedBrandsMatrix " and the nested brand block names are specified in the `*IF` lines. Inside each brand block there is a nested " UsedBrands " block.

#### 6.73.1.1 See Also

\*LABEL..... 168  
 \*VAR..... 284

## 6.74 \*TEMPLATE

### Purpose

Switch to a specific template and optional theme.

### Syntax

```
*TEMPLATE "[templatename[;themename]]"
```

### Description

Switches to the selected template and theme at the next question or page, except for introductions (see remarks below). Templates and themes are typically selected at the start of the script. Using different templates in a single questionnaire is not recommended, but themes may be changed to appeal to certain demographics or characteristics of the respondent.

### Arguments

`templatename`

The full name of the template to use. The name is case-insensitive. The template must be available on your domain.

`themename`

The name of the theme to use. The name is case-insensitive. The theme must have been previously uploaded to Nfield.

### Remarks

- When no template is selected, a system factory styling is applied.
- Templates contain template-specific control rendering and rendering options. Script logic may rely on rendering controls that are specific to the template in use. Switching templates in a questionnaire may cause a template to malfunction because the rendering controls used by a question might not be available in the last selected template. Template switching during a questionnaire is therefore not recommended. To apply a different look and feel to a template, consider switching between themes.
- To return to the factory default template, use `*TEMPLATE ""`.
- To switch off a theme without using a new theme, call `*TEMPLATE` with the template name followed by a semi-colon only.

### Example

```
** Use the Nfield template
*TEMPLATE "NfieldChicago"

*QUESTION 1 *CODES 61
Do you own a car?

1: Yes
2: No

** Switch to a different theme
* TEMPLATE "NfieldChicago;softdrink"

*QUESTION 2
Thank you for your cooperation.
```

6.74.1.1 See Also

*UIOPTIONS (command) .....	270
*UIRENDER (question option) .....	272
*UIRENDER (command) .....	274
*UIRENDER (question option) .....	276

## 6.75 \*TEXTVARS

### Purpose

Defines one or more text variables or arrays.

### Syntax

```
*TEXTVARS <name>[,name2,...]
```

or

```
*TEXTVARS <name[size]>[,name2[size2],...]
```

### Description

Defines text variables or one-dimensional text arrays. This command has to be at the beginning of the line. These text variables or arrays can be used throughout the entire questionnaire to store data or retrieve data. In text variables or arrays all alphanumerical data can be stored. A text variable or array initially is empty.

---

#### Note:

For technical reasons it is not possible to use variables which name start with an **L**. Also, you cannot have a variable named **B**. As a rule, it is not recommended to use 1 letter variable names.

System variables do not need to be defined.

---

### Arguments

*name*

The name of the variable or array. Multiple variables or arrays can be created with a single

**\*TEXTVARS** command by including additional variable or array names separated by commas (Name2, Name3, and so on).

*size*

Specifies the size of the array to be created. Only one-dimensional arrays are supported. The variable will contain the specified amount of rows: you have created *size* variables with the same name.

### Example

```
*TEXTVARS UHELP[2]
*QUESTION 2 *ALPHA 151L20 *SAVE UHELP[1]
What is the name of the oldest child?

*QUESTION 3 *ALPHA 171L20 *SAVE UHELP[2]
What is the name of the second oldest child?

*PAGE
The oldest child is named *? UHELP[1]
The second oldest child is named *? UHELP[2]
```

6.75.1.1 See Also

*?.....	99
*PUT.....	217
*SAMPLEDATA .....	246
*SAVE (codes option).....	249
*SAVE (question option) .....	247
*VARS .....	286

## 6.76 \*UIOPTIONS (command)

### Purpose

Sets the options for all questions with the currently active rendering control of a question.

### Syntax

```
*UIOPTIONS "[name1]=[value1];[name2]=[value2];{...}"
```

### Description

This control sets the options for the questions with the currently active rendering control for the remainder of the questionnaire or until another option is selected.

### Arguments

```
name1=value1;name2=value2; ...
```

Refer to the template documentation for details for the names of the options and the possible values given to these options. Note that depending on the template design, the options may be case-sensitive. Do not use ' ' (space) after ';' (semicolon)!

### Remarks

- Ensure that the currently active template supports the options and that the values are valid. Unknown options are ignored.
- To reset default options for a \*UIOPTION control, use `*UIOPTIONS ""`
- When there are multiple types of \*UIOPTIONS in a script, new options will be appended to existing options, i.e. the new command will not completely replace the previous command. Only the changed types of the \*UIOPTIONS will be overwritten, and other (not mentioned in the new command) types of \*UIOPTIONS will be appended. This also applies to \*UIOPTIONS on a question. The previous options will be reverted when leaving the scope of the question.
- In case the \*UIOPTIONS value is an expression, the expression will be evaluated at the time the \*UIOPTIONS block is encountered, not at the time the options are applied to a block.
- \*UIOPTIONS can also be a variable.
- When switching templates, \*UIOPTION controls set for question types are NOT reset to the factory defaults. So, if we set some \*UIOPTIONS in one template, then switch to another template, if this new template supports these \*UIOPTIONS, they will be enabled. If it does not -- they will be ignored. If we revert to the first template, those \*UIOPTIONS will still be enabled.

### Example: Global \*UIOPTIONS vs question \*UIOPTIONS

The question options overwrite the global ones for that specific question (question 2). The global ones are reverted to on the next question (Q3 in our example). So, for the Q1 and Q3 we see the instruction “Common instructions: please type some text”, and for Q2 we see instruction “Question 2 instructions: please type a number between 0 and 99”.

```
*TEMPLATE "NfieldChicago"
*UIOPTIONS "Instruction=Common instructions: please type some text"

*QUESTION 1 *ALPHA 61L30
A text question

*QUESTION 2 *NUMBER 92L2 *UIOPTIONS "Instruction=Question 2 instructions: please type a number between 0 and 99"
Numeric question

*QUESTION 3 *ALPHA 94L30
Text question again

*END
```

#### 6.76.1.1 See Also

*TEMPLATE.....	266
*UIOPTIONS (question option).....	272
*UIRENDER (command).....	274
*UIRENDER (question option).....	276

## 6.77 \*UIOPTIONS (question option)

### Purpose

Sets the options for the current question with the currently active rendering control of a question.

### Syntax

```
*UIOPTIONS "[name1]=[value1];[name2]=[value2];{...}"
```

### Description

This question option sets the options for the current question with the currently active rendering control. It may be used in combination with the `*UIRENDER` command (see "`*UIRENDER (command)`") to set custom options as defaults for a question type or it may be used to set the rendering control options for one specific question or field only. Note that even without an explicitly selected rendering control, options may be set - in this case the options apply to the template's default rendering control for the question.

### Arguments

```
name1=value1;name2=value2; ...
```

Options that are understood by the currently selected rendering control for the current question type. Refer to the template documentation for details for the names of the options and the possible values given to these options. Note that depending on the template design, the options may be case-sensitive. Do not use ' ' (space) after ';' (semicolon)!

### Remarks

- Ensure that the currently active rendering control supports the options and that the values are valid. Unknown options are ignored.
- To reset default options for a rendering control, use `*UIRENDER "[questiontype]=[renderingcontrol]" *UIOPTIONS ""`
- Enforcing screen orientation for a question: you may want to control the way a question appears on the interviewer's device. In Nfield Manager you can set the screen orientation for your entire survey. Screen orientation for specific questions can be enforced by adding `orientation=landscape;` or `orientation=portrait;` to the `*UIOPTIONS` command on a question. This option can be set for questions separately; it overrides the Nfield Manager settings for the survey. You can - for instance - display large grids/matrix-questions always in landscape, making for a better interviewing experience.

### Example 1

```
*TEMPLATE "NfieldChicago"play to start the video clip.;minimum-duration=-1"

*QUESTION 40 *CODES 64L6 *MULTI *PICT "CarKnow.mp3" *UIOPTIONS "mandatoryplay=false"
** mandatoryplay=false: this option makes playing the audio before answering the question optional
Which of these car brands do you know?

1:Mercedes *PICT "Mercedes.mp3"
2:Chevrolet *PICT "Chevrolet.mp3"
3:Ford *PICT "Ford.mp3"
4:Volvo *PICT "Volvo.mp3"
5:Volkswagen *PICT "Volks.mp3"
6:Toyota *PICT "Toyota.mp3"
```

**Example 2 \*UIOPTIONS for playing a media file**

```
*QUESTION 6000 *CODES L1 *UIOPTIONS "type=play;media=advertorial.mp4;scenario=MediaWithIntro;intro-text=Press play to
start the video clip.;minimum-duration=-1"
Have you previously seen the advertorial that was just shown?
1: Yes
2: No
3: Don't remember
```

**Example 3: \*Form Fields with Default Values, 2 of Them Limited in Size, and Displaying Filled Character Count**

Powered by NPD

```
*TEMPLATE "NfieldChicago"
```

```
*QUESTION 10 *FORM *UIOPTIONS "instruction=Form with alpha and number fields. (* marks a required field)"
Please share your personal information with us:
```

```
1:First name*: *ALPHA 61L25 *UIOPTIONS "placeholder=First name;characterCount=true" **display char count
2:Last name*: *ALPHA 86L25 *UIOPTIONS "placeholder=Last name;characterCount=true" **display char count
3:Age*: *NUMBER 111L3 *MIN [0] *MAX [110] *UIOPTIONS "placeholder=Age"
4:Nationality*: *ALPHA 114L50 *UIOPTIONS "placeholder=Nationality"
5:Annual income:*NUMBER 164L6 *NON *UIOPTIONS "placeholder=Annual income"
```

```
*END
```

**6.77.1.1 See Also**

```
*TEMPLATE.....266
*UIOPTIONS (command) ..... 270
*UIRENDER (command)..... 274
*UIRENDER (question option)..... 276
```

## 6.78 \*UIRENDER (command)

### Purpose

Sets the rendering control (visual representation and input interface) for all questions of a specific question type.

### Syntax

```
*UIRENDER "[questiontype]=[renderingcontrol]"
```

### Description

This command sets how a question is displayed to the respondent. The rendering control is used for that question type for the remainder of the questionnaire or until another rendering is selected. It may be overridden for specific questions using the `*UIRENDER (question option)`. This command may also be used under condition. If no rendering control is explicitly selected for a question type, templates apply their default rendering for the question type. The rendering control is defined in the template. Refer to the template documentation for the names of the rendering controls. Because of this, it is not recommended to change templates within a questionnaire

### Arguments

`questiontype`

The question type for which to select the new rendering. May be `*NUMBER`, `*ALPHA`, `*CODES` or `*OPEN`. Note that the rendering also applies to `*NUMBER` and `*ALPHA` fields in `*FORM` questions.

`renderingcontrol`

The name of the rendering control. The control must have been defined in the currently active template, and it must be compatible with the selected `questiontype`. Question options and parameters such as the question text, length of the field, and values of `*MIN`, `*MAX` and `*RANGE` are passed to the rendering, but it is dependent upon the rendering whether or not these values are used in the representation of the question.

### Remarks

- To reset to the template's default rendering control, use `*UIRENDER "[questiontype]="`.
- When switching templates, rendering controls set for question types are reset to the factory defaults.
- If more than one template is used by the questionnaire, make sure the selected rendering control exists in the last selected template.

## Example

## You think you spent about \$1000 to 2000. On what did you spend it?

Numerical sliders with a minimum of \$1000 and a maximum of \$2000.

Lunch:



Day trips:



Dinner:



Evening entertainment:



Total:



CLEAR NEXT

```
*TEMPLATE "NfieldChicago"
**Set *FORM questions to a rendering control called "Sumslider" available in NfieldChicago template

*UIRENDER "*NUMBER=Sumslider"

*QUESTION 120 *FORM *UIOPTIONS "instruction=Numerical sliders with a minimum of $1000 and a maximum of $2000."
You think you spent about $1000 to 2000. On what did you spend it?

1:Lunch: *NUMBER 61L4 *NON *MIN [0] *MAX [1000]
2:Day trips: *NUMBER 65L4 *NON *MIN [0] *MAX [1000]
3:Dinner: *NUMBER 69L4 *NON *MIN [0] *MAX [1000]
4:Evening entertainment: *NUMBER 73L4 *NON *MIN [0] *MAX [1000]
5:Total: *NUMBER 77L4 *MIN [1000] *MAX [2000] *UIOPTIONS "field=total"

*END
```

### 6.78.1.1 See Also

\*PICT (codes option)..... 212  
 \*TEMPLATE.....266  
 \*UIOPTIONS (command) ..... 270  
 \*UIOPTIONS (question option)..... 272  
 \*UIRENDER (question option)..... 276

## 6.79 \*UIRENDER (question option)

### Purpose

Sets the rendering (visual representation and input interface) for a single question.

### Syntax

```
*UIRENDER "[renderingcontrol]"
```

### Description

This command sets the new rendering for a single question. This overrides any default rendering control set by the \*UIRENDER command (see "`*UIRENDER (command)`"). The selected rendering control must be available in the currently active template. If no rendering control is specified and no rendering control has been specified for this question type, templates apply their default rendering for the question type.

### Arguments

`renderingcontrol`

The name of the rendering control. The rendering must have been defined in the currently active template, and it must be compatible with the question on which the rendering is used. Rendering controls must have been documented by template's designer. Refer to the template's documentation for more information.

Question options and parameters such as the question text, length of the field, and values of \*MIN, \*MAX and \*RANGE are passed to the rendering control, but it is dependent upon the rendering control whether or not these values are used in the representation of the question.

### Remarks

- If more than one template is used by the questionnaire, make sure the selected rendering control exists in the currently active template.
- This question option may also be used on fields in a \*FORM question.

### Example: \*UIRENDER determines that the \*MATRIX contains the radio sliders

```
*TEMPLATE "NfieldChicago"

*QUESTION 10 *CODES 61L1 *DUMMY
Things to do as a tourist:
1:Eating
2:Accomodation
3:Transportation
4:Shopping

*MATRIX 4 Q10 *FIELD 62L4 *UIRENDER "Radiosliders" *UIOPTIONS "instruction=Please rate all statements"
What do you think of the price level in Chicago, regarding...

*QUESTION 710 *CODES 1L1
Costs

1:Very low
2:Low
3:Average
4:High
5:Very high

*ENDMATRIX

*END
```

**What do you think of the price level in Chicago, regarding...**

Please rate all statements

**Eating**

Very low Low Average High Very high

**Accommodation**

Very low Low Average High Very high

**Transportation**

Very low Low Average High Very high

**Shopping**

Very low Low Average High Very high

CLEAR NEXT

Powered by NPTD

#### 6.79.1.1 See Also

*TEMPLATE.....	266
*UIOPTIONS (command) .....	270
*UIOPTIONS (question option).....	272
*UIRENDER (command).....	274

## 6.80 \*USEBUTTONS

### Purpose

This command allows the scripter to specify a list of buttons that can be used to answer a question.

### Syntax

```
*USEBUTTONS [pos]L<length>|<pos> <"list_of_buttons">
```

### Description

This command allows the scripter to call on a list of buttons that can be used to answer a question. It has a separate field to store the answer independent from the question. Button properties are also supported if defined in the button list.

This command does not replace \*BUT command, but \*BUT has an issue: if the value written by the \*BUT command is also a valid answer to the question (for example, for a question “What is your

age?", and the `*BUT` command with a value of 99, there will be an ambiguity if 99 is answered). The `*USEBUTTONS` solves this by storing the button answer in separate field.

### Arguments

[pos]L<length>

The 1st parameter is the field where the button's answer will be stored. The 2nd one is the length. It needs to match the highest value of the button.

list\_of\_buttons

A list that contains codes for every button that the scripter would like to include. The list reference can either be a number or a name, where the name can optionally be surrounded by single quotes or double quotes. The list can contain properties.

### Remarks

- An important behavioral difference between `*BUT` and `*USEBUTTONS` is that the button answer will no longer be stored in the regular answer field. Expressions that refer to the answer through `'Qn(Fm)'` will consider those questions 'empty'. The one notable exception is the `'Qn,B'` construction (which is used to detect whether a question was answered). This will have to return '1' also when a button was used to answer the question.
- The related operator is `?BUTTON(Qn)`, where `Qn` is a question, returns the number of the button if the question was answered with a button, or -1 if the button was not used. This could be used, for example, with an `*IF` command:
 

```
*IF [?BUTTON(Q1) = 2] *GOTO 2
```
- Using both `*USEBUTTONS` and `*BUT` on a question is not allowed.
- We advise you to use the `*USEBUTTONS` command in combination with `*ALPHA` and `*NUMBER` to prevent ambiguity.
- Similar to `*BUT`, `*USEBUTTONS` has no interaction with `*CONTROL` and all buttons will always be shown.
- `*SAVE` also works identically - it will store the code into number variables and the text into text variables.
- The behavior of `*COPY`, `*INCLUDE` and `*EXCLUDE` is also identical to when `*BUT` is used. This means that if you want to check at runtime if a button is used from the `*USEBUTTONS` list, you need to use the argument `?BUTTON(Qn)`.
- Buttons are not supported on matrixes and blocks.

### Exports

- Upon export to .VAR a variable 'Vn\_B' will be created that contains the list of possible buttons/categories.
- The export to DIANA will link the question variable to the new button variable:
 

```
*V1_B *SNG 62L1: , Buttons
COM BUTFOR:V1
1:one
```
- The helper field construction will be used by all question types except the `*FORM` questions.

### Example

```
*TEMPLATE NfieldChicago
```

```

*VARS Age,HouseNr,But3,But4

*LIST standard_buttons
1:None of your business *PROPERTIES "DIMELE=_no_answer_"
2:Don't know *PROPERTIES "DIMELE=_dont_know_"

*QUESTION 3 *NUMBER 200L2 *SAVE Age *USEBUTTONS 202L2 "standard_buttons"
What is your age?

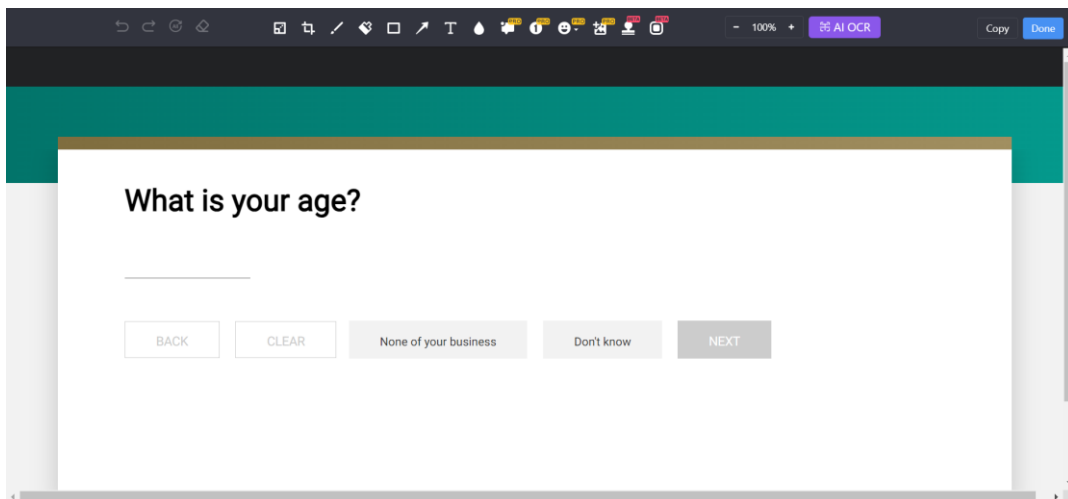
*IF [ ?BUTTON(Q3) = 1 ] *PAGE Very well, you are ageless.
*IF [ ?BUTTON(Q3) = 2 ] *PAGE You don't remember? Are you really that old?
*IF [ ?BUTTON(Q3) = -1 ] *PAGE You are *?Age years old and there is no doubt about it.

*QUESTION 4 *FORM *USEBUTTONS 205L1 "standard_buttons"
Please, fill in your address details:
1:House number: *ALPHA 215L10
2:Street name: *ALPHA 225L20
3:Town/City: *ALPHA 255L20
4:State: *ALPHA 275L2
5:ZIP Code: *NUMBER 277L5

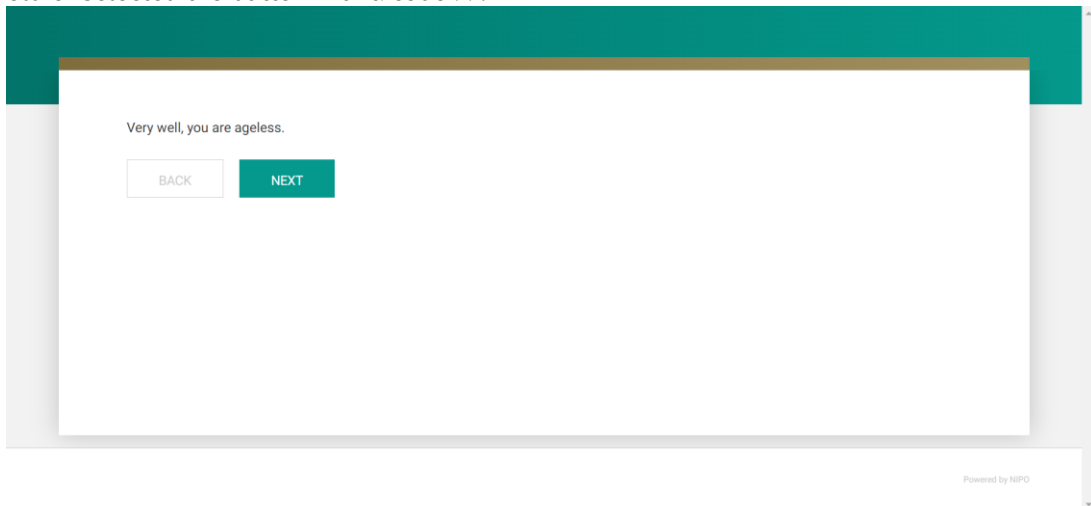
*IF [ ?BUTTON(Q4) = 1 ] *PAGE OK, so we're not welcome at your house.
*IF [ ?BUTTON(Q4) = 2 ] *PAGE Do you need a place to stay?
*IF [ ?BUTTON(Q4) = -1 ] *PAGE Your house number is *?HouseNr and there is no doubt about it.

*END

```



If the respondent selects **None of your business**, there is no confusion whether the person is 99 years old or selected the button with a code 99:

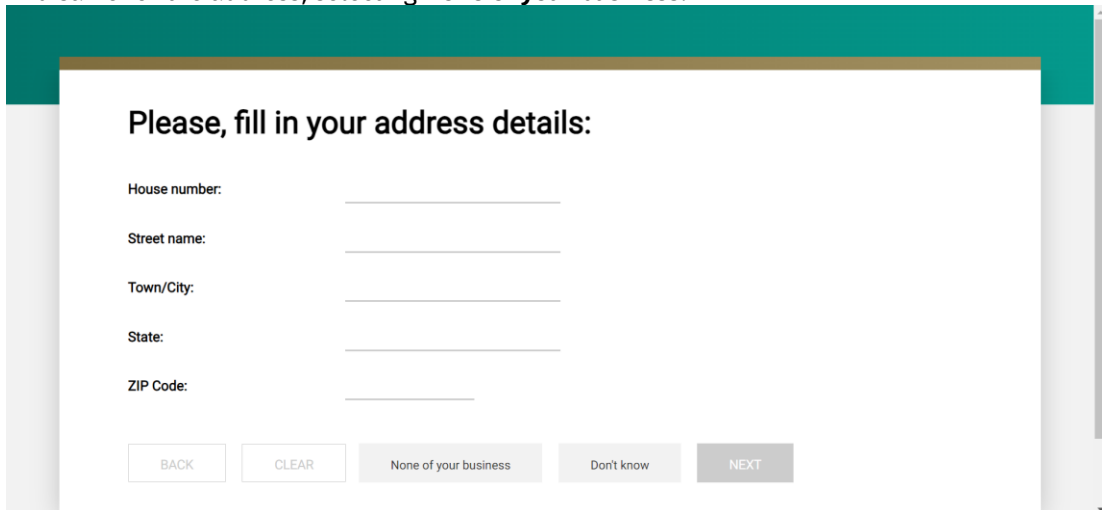


Very well, you are ageless.

BACK NEXT

Powered by NPO

And same for the address, selecting **None of your business**:



Please, fill in your address details:

House number: \_\_\_\_\_

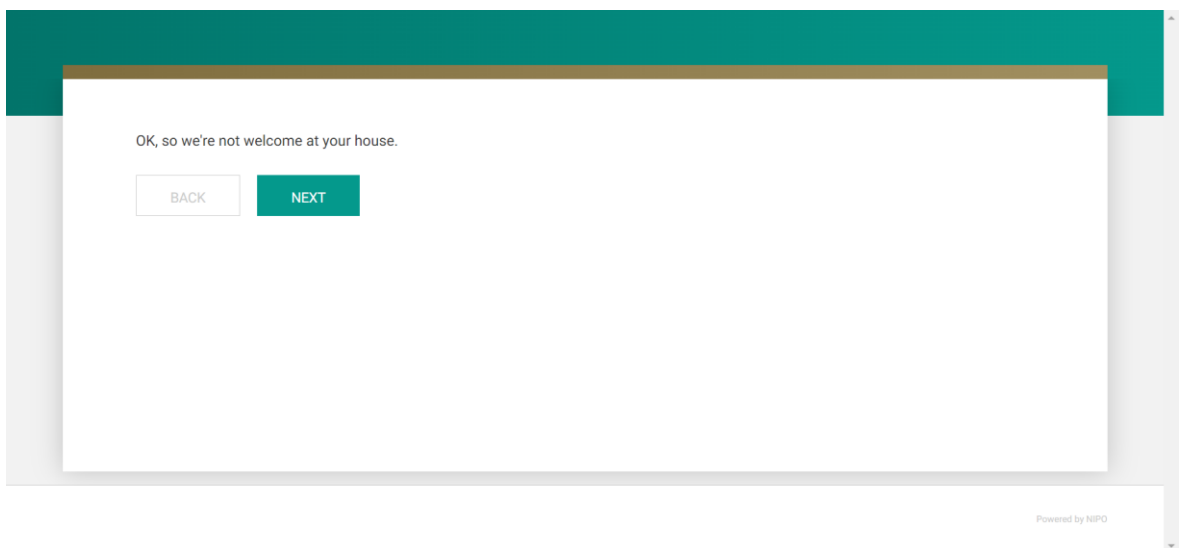
Street name: \_\_\_\_\_

Town/City: \_\_\_\_\_

State: \_\_\_\_\_

ZIP Code: \_\_\_\_\_

BACK CLEAR None of your business Don't know NEXT



OK, so we're not welcome at your house.

BACK NEXT

Powered by NPO

6.80.1.1 See Also

\*BUT ..... 111

## 6.81 \*USELIST

### Purpose

Use answer codes from a list.

### Syntax

```
*USELIST <n| [expression] | "name[,property=value[;property=value];...]">
```

### Description

This command is always used in combination with a closed question and uses a list as a substitute for (a part of) a code list. This is similar to the \*LIST command except that \*USELIST has to be used as (part of) a code list while \*LIST has to be used as a question option. More than one \*USELIST may be used in a single question.

### Arguments

`n|expression`

This is a positive integer or expression that indicates an existing list.

`name`

This is the name of an existing list.

### Remark

This command can only be used in the context of closed coded questions.

### Example 1

```
*LIST 1
1: Heineken
2: Amstel
3: Grolsch
4: Carlsberg
5: Tuborg

*QUESTION 1 *CODES L7 *MULTI
What brands of beer do you know?

*USELIST 1

6: Other *OPEN
7: Don't know *NMUL
```

In this example, the first five codes come from a predefined code list.

Display in a question only the categories available in their region.

```
*LIST "Outlets"
1:Outlet 1 *PROPERTIES "nl=Y;be=N;de=N"
2:Outlet 2 *PROPERTIES "nl=N;be=Y;de=Y"
3:Outlet 3 *PROPERTIES "nl=Y;be=Y;de=N"
4:Outlet 4 *PROPERTIES "nl=N;be=N;de=Y"
5:Outlet 5 *PROPERTIES "nl=N;be=N;de=Y"
6: Outlet 6 *PROPERTIES "nl=Y;be=Y;de=Y"

*QUESTION 101 *CODES L1
Choose your country

1:Belgium *PROPERTIES "country=be"
2:Germany *PROPERTIES "country=de"
3:Netherlands *PROPERTIES "country=nl"

*TEXTVARS sCountry
*PUT sCountry [?PROPERTY(Q101, Q101, "country")]
```

```
*QUESTION 201 *CODES L1
Which outlet did you visit?
```

```
*USELIST "Outlets, *? sCountry=Y"
```

It is possible to use an expression on `*USELIST`.

By writing an expression after the `*USELIST` command and before the `listname` parameter, Nfield will run the expression for each item in the list. If the expression results in true, it will include the item in the question (see the example below):

### Example 3

```
*LIST *Mylist
1: A *PROPERTIES "Type = a; Size = big; shape = round"
2: B *PROPERTIES "Type = b; Size = big; shape = square"
3: C *PROPERTIES "Type = c; Size = big; shape = round"
4: D *PROPERTIES "Type = d; Size = small; shape = round"

*QUESTION 1 *CODES 61L4 *MULTI
*USELIST [?property("Type") = a \ ?property("type") = b \ ?property("size") = small & ?property("shape") = round] "Mylist"
```

If we run this script, we will see that `Question 1` only displays codes 1 and 4 because the expression reads that properties (`type = A or type = B or Size = small`) and `shape = round`. Only codes 1 and 4 meet those requirements.

For more information, please watch our [Nfield Academy #44](#).

### 6.81.1.1 See Also

\*LIST (definition)..... 176  
 \*LIST (question option).....179  
 \*PROPERTIES..... 214

## 6.82 \*VAR

### Purpose

Defines a variable name for export to NIPO DIANA / Nvision Script variables, IBM SPSS, et cetera.

### Syntax

```
*VAR <name>
```

### Description

This command is always used in combination with `*QUESTION` and a question type definition and must be specified after these commands on the same line. Normally variable names in the NIPO Diana / Nvision Script variable file are based on the question numbers in your NIPO ODIN questionnaire. The `*VAR` command gives the possibility to create your own variable names during the export to NIPO Diana / Nvision Script, IBM SPSS, et cetera.

### Arguments

name

The variable should not contain spaces and should not start with a number.

### Remark

The variable name defines the export variable name of the question in the export. This is not the same as the variables created within your NIPO ODIN questionnaire with the `*VARS` and `*TEXTVARS` commands.

The variable names do not have to be unique in the NIPO ODIN questionnaire, but obviously your analysis tool will not be able to differentiate variables with the same name. The NIPO ODIN Developer syntax check produces a `Warning: Duplicate variable name message` if a variable name is not unique. This warning has no impact on running the questionnaire, and names may be changed without consequences at a later time.

Variable names within a `*REPEAT` block are suffixed with a repeat number. Variable names within a `*GOSUB` are suffixed with a number indicating the number of calls the routine was called until then. In some exports, multiple-coded variables are exported as multiple dichotomy variables, where the variable name is suffixed with a code label index number.

### Example

```
*QUESTION 1 *CODES L1 *VAR Gender
Int. type the respondents gender

1:Male
2:Female

*QUESTION 2 *NUMBER L2 *VAR Age
Could you please tell me your age?
```

Export to NIPO Diana / Nvision Script creates the following variables:

```
*Gender *SNG 61L1: Int. type the respondents gender
1:Male
2:Female

*Age 62L2: Could you please tell me your age?
```

Export to IBM SPSS creates the following variables:

VARIABLE LABELS  
 Gender 'Int. type the respondents gender'  
 Age 'Could you please tell me your age?'

6.82.1.1 See Also

\*LABEL..... 168  
 \*REPEAT ... \*ENDREP ..... 230  
 \*TABLE..... 264

## 6.83 \*VARS

### Purpose

Defines one or more numeric variables or arrays.

### Syntax

```
*VARS <name>[, name2, ...]
```

or

```
*VARS <name[size]>[, name2[size2], ...]
```

### Description

Defines numeric variables or one-dimensional number arrays. This command must be specified at the beginning of a line. Numeric variables or arrays can be used throughout the entire questionnaire to store data or retrieve data. In numeric variables or arrays all alphanumerical data can be stored. A numeric variable or array initially is empty.

### Arguments

*name*

The name of the variable or array. Multiple variables or arrays can be created with a single *\*VARS* command by including additional variable or array names separated by commas (name2, name3, and so on).

*size*

Specifies the size of the array to be created. The variable can store up to *size* values.

### Remark

- Only one-dimensional arrays are supported.

### Example

```
*VARS GHELP,HHELP[2]
*QUESTION 1 *NUMBER 93 *SAVE GHELP
How many children, living at home, are there in this
household?

*QUESTION 2 *NUMBER 94L2 *SAVE HHELP[1]
What is the age of the oldest child?

*QUESTION 3 *NUMBER 96L2 *SAVE HHELP[2]
What is the age of the second oldest child?

*PAGE
There are *? GHELP children living at home.

The oldest is *? HHELP[1] years.
The second oldest is *? HHELP[2] years.
```

#### 6.83.1.1 See Also

```
*?.....99
*PUT.....217
*SAMPLEDATA .....246
*SAVE (question option) ..... 247
*TEXTVARS.....268
```





## 7. File Structures and Database Tables

### 7.1 Data Files

All files in NIPO ODIN are in Unicode. All files have a fixed file structure. These structures are described in this section.

#### 7.1.1 Closed Answers File (DAT-file)

The closed answers file (DAT-file) contains the answers to all closed questions. For each interview (respondent) one record (line) is written in the file. The length of the records is fixed and depending on the highest position used in the NIPO ODIN questionnaire. The first positions (fields) in each record are reserved for the system.

##### Record description of the closed answers file

Position	length	Description
1-8	8	interview number
9-10	2	sub-questionnaire number
11-15	5	interview time in seconds
16-19	4	number of screens shown
20	1	Legacy
21-28	8	interviewer ID (CAPI only)
29-40	12	date and time last contact
41	1	0 (zero)
42-60	18	Legacy
61-...		answer data

#### 7.1.2 Open answers file (O-file)

The open-ended answers file (O-file) contains the answers to all open ended and semi-open questions. For each open answer that is actually asked to the respondent, one record (line) is written in the file so several records are stored for each interview (respondent). The amount of records depends on the routing and the number of open-ended questions in the questionnaire. The length of

the records is unfixed and can be as long as the open answer that was entered. The first positions (fields) in each record are reserved for the system.

**Record description of the open answers file (default format)**

position	Length	Description
1-8	8	interview number
9-10	2	sub-questionnaire number
11-15	5	starting position answer field in DAT-file
16-18	3	length reserved answer field in DAT-file
19-...	(depending on 16-18)	corresponding answer code number (or blank)
(depending on 19, etc.)		text of open answer

In cases where the amount of positions used for a single open-ended answer is larger than 999, or if your positions are larger than 999999, the O-file record format is extended:

**Record description of the open answers file (extended format)**

position	Length	Description
1-8	8	interview number
9-10	2	sub-questionnaire number
11	1	Contains * to indicate the record is extended
12-21	10	starting position answer field in DAT-file
22-31	10	length reserved answer field in DAT-file
32-...	(depending on 22-31)	corresponding answer code number (or blank)
(depending on 32, etc.)		text of open answer

Note that the extended O-file format may be recognized by a star in position 11.

## 7.2 Paradata

### 7.2.1 Introduction to Paradata

#### 7.2.1.1 What is Paradata

There are 2 types of data gathered during interviews:

1. Respondent data, such as respondent’s age, gender, his or her answers to interview questions, and related recordings and media files.
2. Technical data related to the interview, such as the id of the survey, survey version, start and end time of the interview, location, time zone, etc.

This second type of data is called Paradata.

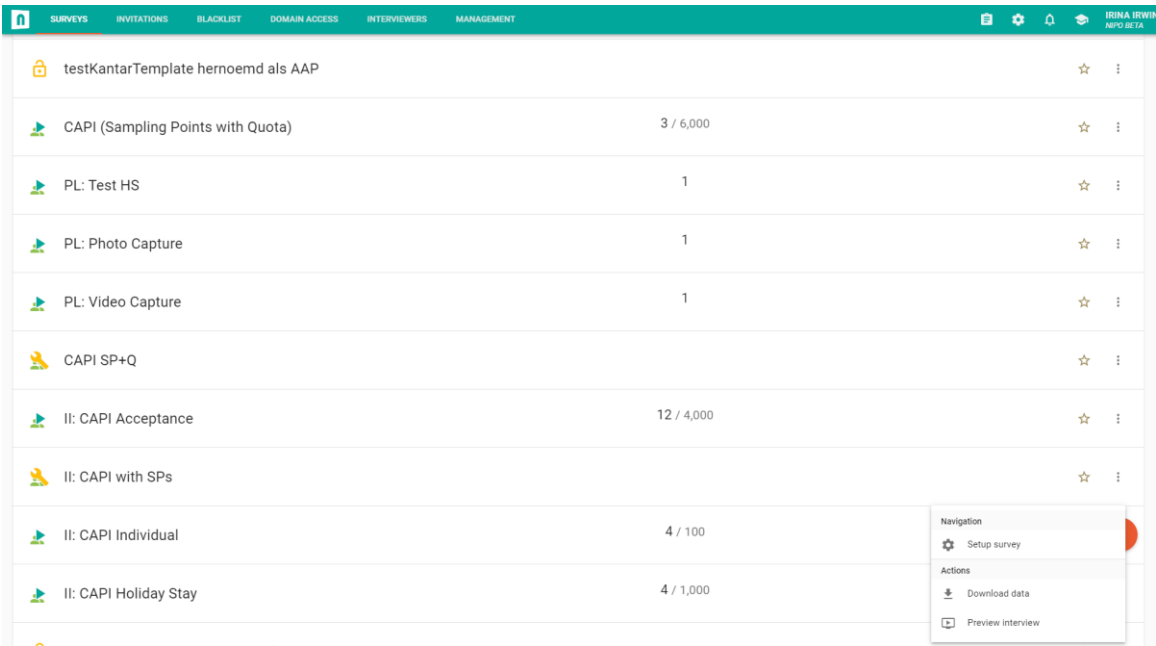
#### 7.2.1.2 What is Paradata Used For

Paradata is there for the Fieldwork Research executives to analyze.

Paradata can also be used by NIPO customer support personnel to help them to determine the cause of the customer’s problem.

#### 7.2.1.3 How to Create and Download Paradata

After the interviewers have done the interviews and uploaded their results to the Nfield Manager by syncing (in case of CAPI surveys), or after the online interviews were done by the users (in case of Online surveys), paradata can be downloaded from the Nfield Manager. To do that, one needs to select the survey, for which the data is being downloaded, in the survey list, and click Download Data button on the Details tab.



In the *Create Download Data* dialog that appears, one needs to deselect types of data not needed (by default, several types of data are selected), leaving only *Paradata*, optionally select version of the ODIN script you would like to download data for (the current one, or one of the previous ones), and click on *Create Download*. You can give the download a distinct name or leave the default (survey name).

Create download data

Interview status

☐ Successful
 ☐ Screen out
 ☐ Dropped out
 ☐ Rejected
 ☐ Test data

Type of data

☐ Closed answers
 ☐ Open answers
 ☒ Paradata
 ☐ Variables file

Date range

From

15 June 2020

To


22 June 2020

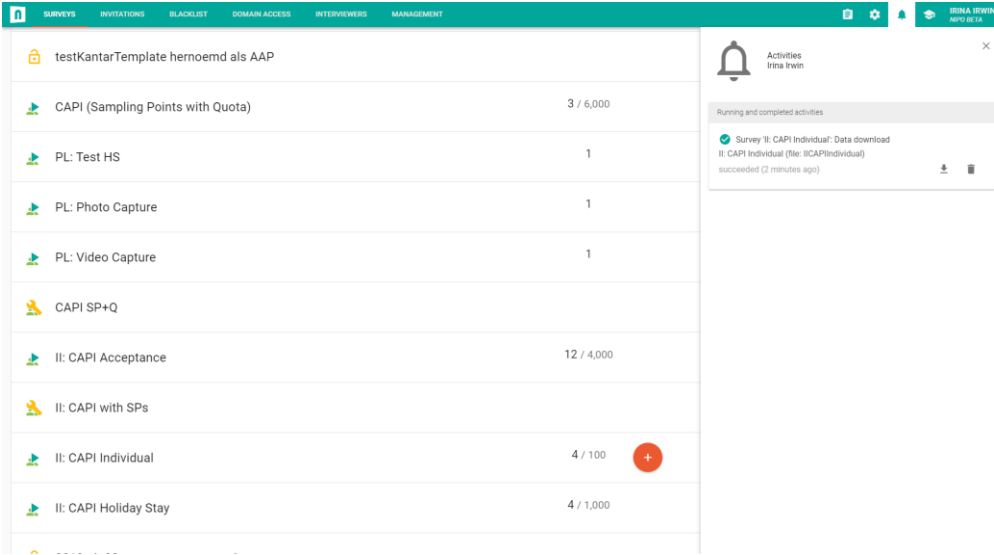
Survey version

June 22, 2020 10:34 AM (latest) ✕

CANCEL

CREATE DOWNLOAD

One can then download the resulting zip file at the Activities page accessed by clicking the icon  on the top right of the screen.

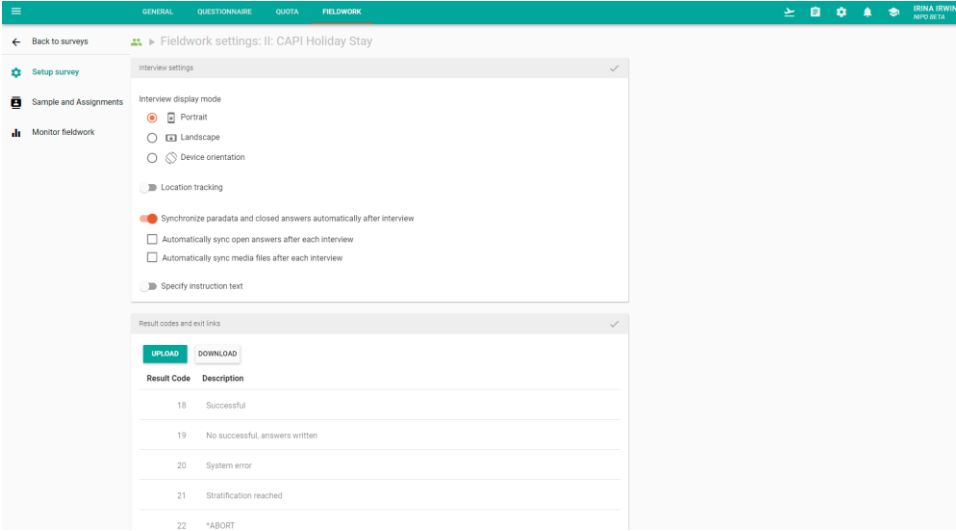


The screenshot shows the Fieldwork Manager interface. The main table lists surveys with their names and progress. The activities panel on the right shows a successful data download for 'Survey II: CAPI Individual'.

Survey Name	Progress
testKantarTemplate hernoemd als AAP	
CAPI (Sampling Points with Quota)	3 / 6,000
PL: Test HS	1
PL: Photo Capture	1
PL: Video Capture	1
CAPI SP+Q	
II: CAPI Acceptance	12 / 4,000
II: CAPI with SPs	
II: CAPI Individual	4 / 100
II: CAPI Holiday Stay	4 / 1,000

7.2.1.4 How to Set Paradata to Auto Sync after Each Interview

Fieldwork Manager can select an option in the Nfield Manager to make sure paradata gets automatically synced after each interview for the survey.



The screenshot shows the 'Fieldwork settings' for 'II: CAPI Holiday Stay'. The 'Interview settings' section is expanded, showing options for 'Interview display mode' (Portrait, Landscape, Device orientation), 'Location tracking', and 'Synchronize paradata and closed answers automatically after interview' (checked). Below this, there are checkboxes for 'Automatically sync open answers after each interview' and 'Automatically sync media files after each interview'. The 'Result codes and exit links' section is also visible, showing a table of result codes and descriptions.

Result Code	Description
18	Successful
19	No successful answers written
20	System error
21	Stratification reached
22	*ABORT

7.2.2 Paradata in CAPI Surveys

For different types of CAPI Surveys, the paradata types collected are also different.

Format of the paradata file is not in a true Jason file format, so please do not use a Jason viewer to view it.

### 7.2.2.1 CAPI survey without Sampling Points

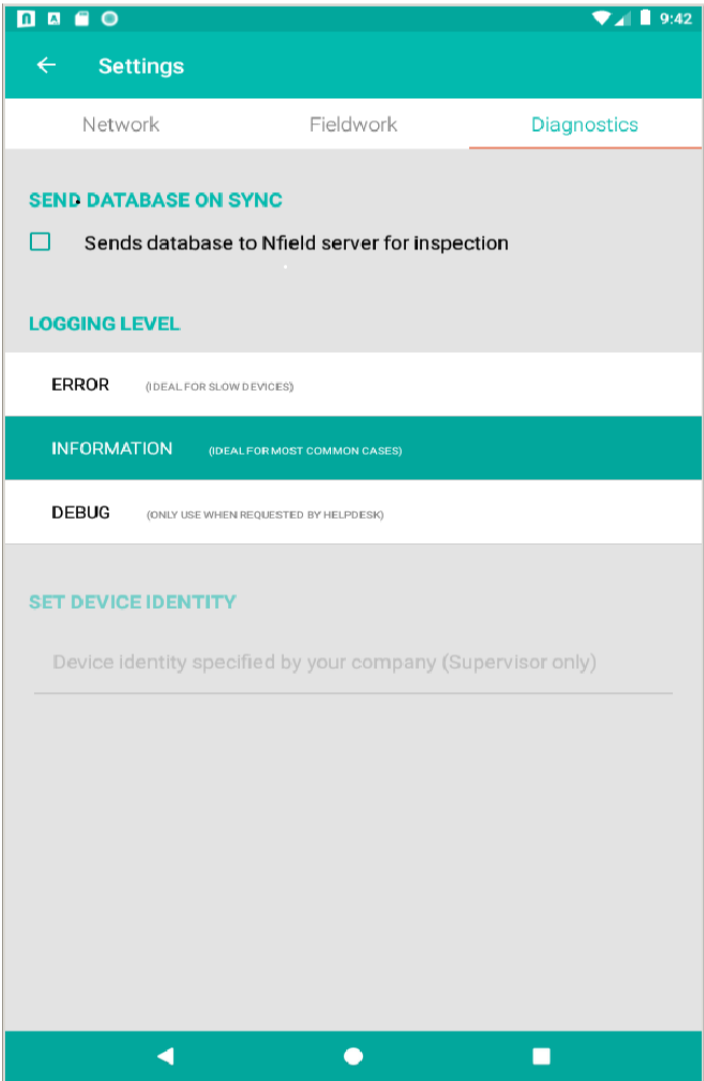
Below is an example of data collected for one interview. In the first column is the interview number, in the second are the paradata labels, and in the third one are the results per label.


1	ClientInformation	{"version":"1.101.000","deviceName":"santos10wifi","manufacturer":"samsung","operatingSystemVersion":"Android 4.2.2 (API level 17)"}
1	DeviceId	Device349
1	EndReason	18
1	InterviewEndTime	2023-11-15T10:03:58.305271Z
1	InterviewRecovery	[{"InterviewWriteTime\\":\\"2023-11-15T10:02:35.7654550Z\\",\\"InterviewRecoveryTime\\":\\"2023-11-15T10:03:45.9892820Z\\"}]"
1	InterviewStartTime	2023-11-15T10:02:29.39838Z
1	InterviewerId	3ADjBukA
1	LastSeenQuestionId	Q1003
1	LocaleId	en-GB
1	SampleData	[]
1	SurveyETag	6.36826442524323442
1	SurveyVersion	ETag(636826442524323442)
1	TestInterview	FALSE
1	TimeZone	CET
1	LocationInfo	{"latitude":"52.50313","longitude":"5.06610","accuracy":15,"IsGeolocationEnabled":true,"LocationValidationState":"Interview"}
1	InterviewQuality	1

**ClientInformation** refers to hardware/software information of the device, used for the interview.

**DeviceId** refers to the id of the device, on which the CAPI interview was performed. **Note** that we are not allowed to use the Android device ids to track users, as it is against the service agreement with Google, so please always enter an id for the device yourself.

One can enter a device id by logging into the CAPI client as an interviewer with a supervisor role, going to the Settings of the CAPI client, Diagnostics tab, and entering the Device ID there in the Set Device Identity field.



Only an interviewer with a supervisor role can enter/edit/delete a device id - this field is non-editable for regular interviewers using the device. Fieldwork Manager can give the supervisor role to that interviewer in the Nfield Manager by toggling the Supervisor Mode icon  into enabled state and saving.

User name	First name	Last name	Email address	Phone number	Fieldwork offices	Interviewer ID	Last sync
tbejeska	Thatch	Beljes	tbejeska@canalblog.com	657-871-2745	[Not assigned to fieldwo...	zzB6YJtr	
smacgettigenb0	Sharleen	MacGettigen	smacgettigenb0@dot.gov	737-367-3985	[Not assigned to fieldwo...	zZAsDhhZ	
wmatteo2w	Willetta	Matteo	wmatteo2w@craigslist.org	401-987-4591	[Not assigned to fieldwo...	Zz4Gq0Wr	
gclemensko	Gabbi	Clemens	gclemensko@state.tx.us	171-213-5443	[Not assigned to fieldwo...	ZvYXDrxV	
ckearm6a	Camella	Kearm	ckearm6a@fastcompany.cc	967-522-4425	[Not assigned to fieldwo...	ZKXVICH4	
asewardsc8	Antons	Sewards	asewardsc8@1688.com	574-668-8005	[Not assigned to fieldwo...	zWAbUPis	
Juggle supervisor mode	Jasmin	Coils	jcoilsd5@scientificamerica	597-981-7708	[Not assigned to fieldwo...	zyylsamE	
houstonmw	Hortensia	Suston	houstonmw@alexa.com	188-892-7222	[Not assigned to fieldwo...	ZUPzPMt1	
kkubica8u	Kathrine	Kubica	kkubica8u@iathis.com	849-589-3165	[Not assigned to fieldwo...	ZuC3loKt	
mdancey3k	Mirilla	Dancey	mdancey3k@toplist.cz	465-730-4708	[Not assigned to fieldwo...	zU4Jh2px	
mnathonhs	Miquela	Nathon	mnathonhs@chron.com	461-954-7870	[Not assigned to fieldwo...	zTKDHWU0	
wmeuseif	Winnah	Meuse	wmeuseif@hc360.com	802-590-8191	[Not assigned to fieldwo...	ZTFrs5Yy	
ggumaryhv	Gardener	Gumary	ggumaryhv@angelfire.com	936-532-1836	[Not assigned to fieldwo...	z5vKzYIX	
stakenton18	Saul	Lakenton	stakenton18@nba.com	406-137-4818	[Not assigned to fieldwo...	Zr3dawph	
nacottg8	Nikoletta	Acott	nacottg8@ow.ly	300-205-3865	[Not assigned to fieldwo...	ZQrs2RnT	
cgarmen8k	Corey	Garment	cgarmen8k@feedburner.co	674-820-2490	[Not assigned to fieldwo...	ZQqqWWEt	
rhannad	Dishon	Tann	rhannad@att.net	111-576-5084	[Not assigned to fieldwo...	ZTf8LUP24	

**EndReason** refers to the survey result codes.

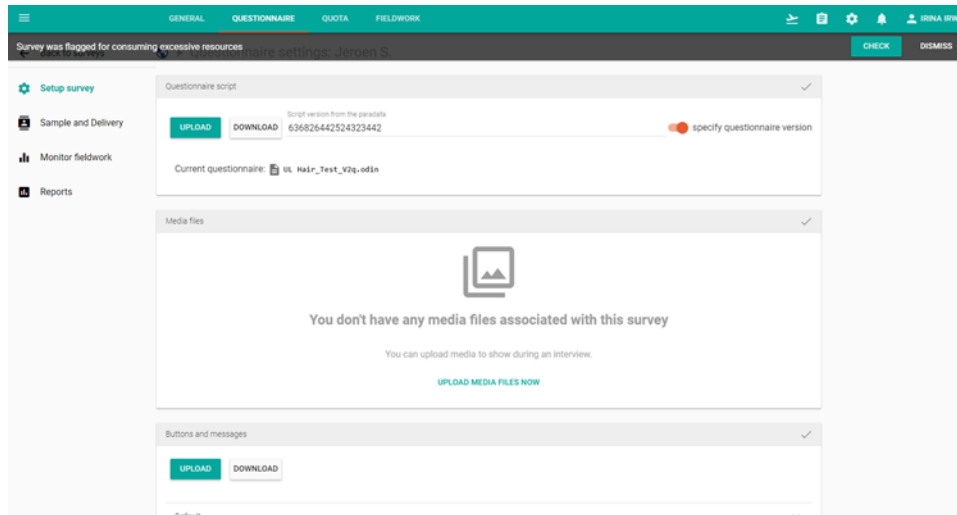
**InterviewStartTime**, **InterviewEndTime**, **startContactDate**, **endContactDate**, **startInterviewDate** **endInterviewDate**, give the time the original interview/contact/scheduled interview was started or stopped as a standard UTC “Zulu” date/time stamp. On the other hand, all the local times (for example, **localStartInterviewDate**, etc.) are the corresponding times in the time zone and the format of the device used for the interview.

**InterviewRecovery**: this value only appears if the app shuts down for any reason during the interview, and the interviewer decides to resume that interview on restarting the CAPI app. Where **InterviewWriteTime** is the time the CAPI app shut down, and the **InterviewRecoveryTime** is time the interview was resumed.

**LastSeenQuestionId**: the id of the last question viewed by the respondent This helps to analyze survey dropouts at the respondent level.

**LocaleId**: the id of the locale. For the list of standard locale ids, please check [here](#).

**SurveyETag** and **SurveyVersion** are the number of the survey version. Using these, one can download the specific version of the questionnaire used in the interview from the Nfield Manager, as in example in screenshot below.



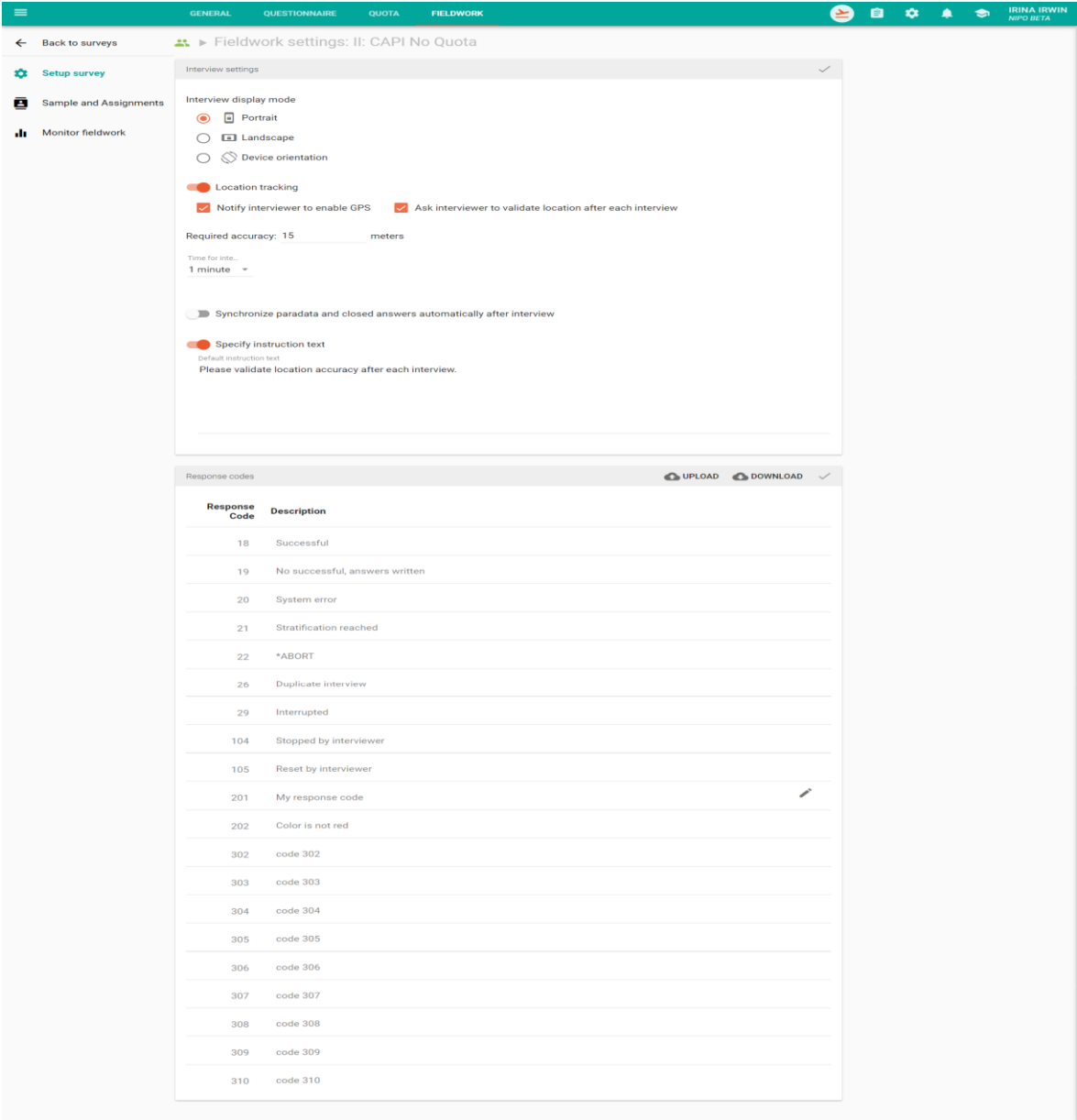
**TestInterview** shows whether the interview was a live one (value False) or a test one (True).

**Timezone** – an abbreviation for the time zone the interview is done in. For more info on time zone abbreviations, please see [this](#).

### Location Information fields

The requirements for location information gathering can be configured by the Fieldwork Manager in Nfield Manager by setting the Location Tracking to Active, and selecting the other relevant location tracing options (as in the screenshot below):

- notifying interviewer to enable his/her GPS if it's off.
- requiring him/her to validate location after each interview with accuracy to so many meters.



**LocationInfo** always holds the best location fix in terms of accuracy. Accuracy describes the estimated horizontal accuracy radius in meters of this location at the 68th percentile confidence level. This means that there is a 68% chance that the true location of the device is within a distance of this uncertainty of the reported location. Another way of putting this is that if a circle with a radius equal to this accuracy is drawn around the reported location, there is a 68% chance that the true location falls within this circle. This accuracy value is only valid for horizontal positioning, and not vertical positioning.

Please note that if the interview is very short (a couple of questions, for example), the app might not have enough time to capture and record the GPS coordinates in paradata. To make sure it will always happen, please check the “Ask interviewer to validate location after each interview” box.

**IsGeolocationEnabled** - this variable can have the values *true* or *false* and tells if the geo location is enabled on the interviewers device during the interview.

**LocationValidationState** (as part of **LocationInfo**), the possible values are:

- 'Interview' (which means the fix was achieved at the start of the interview);
- 'ForceWait' (the fix of location is coming after the interview because the Fieldwork Manager has configured the demand to check the location fix after each interview).

**InterviewQuality** – interviews can be checked and marked by the Fieldwork Executive in Nfield Manager/Quality Control tab. The numbers in this field stand for the following values:

- 0: interview has not been not checked,
- 1: approved,
- 2: unverified,
- 3: rejected

#### 7.2.2.2 CAPI survey with Sampling Points and Quota

1	ClientInformation	{"version":"1.101.000","deviceName":"santos10wifi","manufacturer":"samsung","operatingSystemVersion":"Android 4.2.2 (API level 17)"}
1	DeviceId	Device349
1	EndReason	18
1	InterviewEndTime	2019-01-09T15:40:01.227448Z
1	InterviewStartTime	2019-01-09T15:39:00.0000000Z
1	InterviewerId	3ADjBukA
1	LastSeenQuestionId	Q1003
1	LocaleId	en-GB
1	Quota	[{"levelId":"50507613-218a-48f5-b137-652dde8a7ff8","path":"Gender,Male"}, {"levelId":"356934ad-7f90-46af-930f-532fa36de4b8","path":"Age,18 - 24 years"}]
1	QuotaVariables	[{"Gender":"Male"}, {"Age":"18 - 24 years"}]
1	SamplingPointId	1
1	SurveyETag	6.36826442524323442
1	SurveyVersion	ETag(636826442524323442)
1	TestInterview	FALSE

1	TimeZone	CET
1	LocationInfo	{"latitude":"52.50313","longitude":"5.06610","accuracy":15,"IsGeolocationEnabled":true,"LocationValidationState":"Interview"}
1	SampleData	[{"name":"Gender","value":"Male"}, {"name":"Age","value":" 18 - 24 years"}]
1	InterviewQuality	1

The extra fields in this type of survey (in addition to the ones in the previous type) are:

**Quota, QuotaVariables, SampleData and SamplingPointId** refer to the quota and sampling points covered by the interview.

**Quota** specifies the **levelid** (a GUID value) of the quota selected, as well what quota was selected (**path**).

**QuotaVariables** are the Odin script variables entered in the quota frame (as in example below):

**SamplingPointId** is id of the sampling point from the Nfield Manager's survey Sample page (see example below).

Back to surveys

Sampling points: II: CAPI Acceptance

Setup survey

Sample and Assignments

Monitor fieldwork

Sampling points

ASSIGNED QUOTA

ASSIGNED INTERVIEWERS

DOWNLOAD TEMPLATE

UPLOAD

DELETE ALL SAMPLING POINTS

ADD

ADD FILTER

Id	Name	Description	Group Id	Minimum target	Fieldwork office
1	SP1			6000	Castricum
2	SP2			6000	Halfweg
3	SP3			6000	Headquarters
ams	SP4			6000	Amsterdam

7.2.2.3 CAPI survey with Sampling Points and Addresses

1	Address	<pre>{"surveyId":"d05a7b55-28a4-401f-bda1-5820bbb8179e","samplingPointId":"2","addressId":"Address002","addedDate":"2023-11-06T08:20:48Z","appointmentDate":null,"localAppointmentDate":null,"appointmentEndDate":null,"localAppointmentEndDate":null,"lastContactDate":null,"sampleData":[{"name":"gender","value":"Male"},{"name":"age","value":"Below 35"}],"contacts":[{"addressId":"Address002","samplingPointId":"2","surveyId":"d05a7b55-28a4-401f-bda1-5820bbb8179e","contactId":"d616eab5-150e-4a58-be9a-959305fcab23","interviewerId":"DTlw3fgY","startContactDate":"2023-11-06T08:40:19.179133Z","localStartContactDate":"11/6/2023 9:40:19 AM","startContactLocation":{"latitude":"52.50313","longitude":"5.06610","accuracy":16,"IsGeolocationEnabled":true},"startInterviewDate":"2023-11-06T08:40:19.179133Z","localStartInterviewDate":"11/6/2023 9:40:19 AM","endInterviewDate":"2023-11-06T08:40:29.292375Z","localEndInterviewDate":"11/6/2023 9:40:29 AM"}</pre>
---	---------	--

		Date":"11/06/2023 09:40:29","endContactDate":"2023-11-06T08:40:29.292375Z","localEndContactDate":"11/06/2023 09:40:29","responseCode":18,"isFinal":true]],{"details":"Pigeon Place 22, Groningen"}
1	AddressId	ccc2e21b-f323-4377-9289-f16e42605623
1	ClientInformation	{"version":"1.101.000","deviceName":"santos10wifi","manufacturer":"samsung","operatingSystemVersion":"Android 4.2.2 (API level 17)"}
1	DeviceId	Device349
1	EndReason	18
1	InterviewEndTime	2019-01-09T15:40:01.227448Z
1	InterviewStartTime	2019-01-09T15:39:00.0000000Z
1	InterviewerId	3ADjBukA
1	LastSeenQuestionId	Q1003
1	LocaleId	en-GB
1	Quota	[{"levelId":"RootLevelId","path":""}]
1	QuotaVariables	[]
1	SampleData	[{"name":"Gender","value":"Male"},{"name":"Age","value":" 18 - 24 years"}]
1	SamplingPointId	3
1	SurveyETag	6.36826442524323442
1	SurveyVersion	ETag(636826442524323442)
1	TestInterview	TRUE
1	TimeZone	CET
1	LocationInfo	{"latitude":"52.50313","longitude":"5.06610","accuracy":16,"IsGeolocationEnabled":true,"LocationValidationState":"ForceWait"}
1	InterviewQuality	1

**Address** is added in this type of CAPI survey.

Format of the address paradata is not in a true Jason file format. There is a way to convert paradata to a classic contact log using the Nfield Tool. For more information on this, please contact our [helpdesk](#).

Address contains the following fields that are specific to the address for which the interview was done (below, with example values):

1. "surveyId": "c78beb14-09ff-45cb-a34f-bb1717878403",
2. "samplingPointId": "bb83cd8b-0b22-4c9c-bf01-d7c1d830d7d8",
3. "addressId": "ccc2e21b-f323-4377-9289-f16e42605623",
4. "addedDate": "2019-01-09T15:39:45.785958Z",
5. "appointmentDate": "2019-01-17T12:15:00Z", ""
6. "localAppointmentDate": "1/17/2019 1:15:00 PM",
7. "appointmentEndDate": "2019-01-17T12:30:00Z",
8. "localAppointmentEndDate": "1/17/2019 1:30:00 PM",
9. "lastContactDate": "2019-01-09T15:39:24.364Z"
10. "sampleData": null,
11. "contacts": [{
  - 1) "addressId": "ccc2e21b-f323-4377-9289-f16e42605623",
  - 2) "samplingPointId": "bb83cd8b-0b22-4c9c-bf01-d7c1d830d7d8",
  - 3) "surveyId": "c78beb14-09ff-45cb-a34f-bb1717878403",
  - 4) "contactId": "8a4dc2e1-6df1-4c0b-85e0-d24b5b2efad0",
  - 5) "interviewerId": "3ADjBukA",
  - 6) "startContactDate": "2019-01-09T15:39:24.364Z",
  - 7) "localStartContactDate": "09/01/2019 16:39:24",
  - 8) "startContactLocation": {
    - a. "IsGeolocationEnabled": true,
    - b. "latitude": 52.344288,
    - c. "longitude": 4.912397,
    - d. "accuracy": 30},
  - 9) "startInterviewDate": "2019-01-09T15:39:47.228823Z",
  - 10) "localStartInterviewDate": "09/01/2019 16:39:47",
  - 11) "endInterviewDate": "2019-01-09T15:40:00.158149Z",
  - 12) "localEndInterviewDate": "09/01/2019 16:40:00",
  - 13) "endContactDate": "2019-01-09T15:40:00.158149Z",
  - 14) "localEndContactDate": "09/01/2019 16:40:00",
  - 15) "responseCode": 18,
  - 16) "isFinal": true}],
12. "details": "5 Kalvinstraat, Rotterdam"

**Contacts** describes the details of the contacts made by the interviewer at the address, including an initial contact, and the follow up appointment for a new interview time/date, if any.

**IsFinal** is set to True if this is the final interview at this address (whether successful or a definitely failed one), or to False if no definite rejection or success were achieved during the interview (for example, the respondent did not open the door).

Address **details** refers to address description as entered when address was created.

#### 7.2.2.4 CAPI survey with Sampling Points, Quota and Addresses

1	Address	{"surveyId":"c78beb14-09ff-45cb-a34f-bb1717878403","samplingPointId":"bb83cd8b-0b22-4c9c-bf01-d7c1d830d7d8","addressId":"ccc2e21b-f323-4377-9289-f16e42605623","addedDate":"2019-01-09T15:39:45.785958Z","appointmentDate":null,"localAppointmentDate":null,...
1	AddressId	ccc2e21b-f323-4377-9289-f16e42605623
1	ClientInformation	{"version":"1.101.000","deviceName":"santos10wifi","manufacturer":"samsung","operatingSystemVersion":"Android 4.2.2 (API level 17)"}
1	DeviceId	Device349
1	EndReason	18
1	InterviewEndTime	2019-01-09T15:40:01.227448Z
1	InterviewStartTime	2019-01-09T15:39:00.0000000Z
1	InterviewerId	3ADjBukA
1	LastSeenQuestionId	Q1003
1	LocaleId	en-GB
1	Quota	[{"levelId":"50507613-218a-48f5-b137-652dde8a7ff8","path":"Gender,Male"}, {"levelId":"356934ad-7f90-46af-930f-532fa36de4b8","path":"Age,18 - 24 years"}]
1	QuotaVariables	[{"Gender":" Male"}, {"Age":" 18 - 24 years"}]

1	SampleData	[{"name":"Gender","value":"Male"}, {"name":"Age","value":" 18 - 24 years"}]
1	SamplingPointId	3
1	SurveyETag	6.36826442524323442
1	SurveyVersion	ETag(636826442524323442)
1	TestInterview	TRUE
1	TimeZone	CET
1	LocationInfo	{"latitude":"52.50307","longitude":"5.06607","accuracy":20,"IsGeolocationEnabled":true,"LocationValidationState":"Interview"}
1	InterviewQuality	1

### 7.2.3 Paradata in Online Surveys

For Online surveys much less paradata gets collected.

1	Contacts	[{"StartTime":"2024-11-23T18:26:14.8864771Z","EndTime":"2024-11-23T18:28:10.1895501Z","ResponseCode":18,"IsFinal":true,"SuspendReason":""}]
1	InterviewStartTime	2019-01-10T13:16:39.0910529Z
1	LastSeenQuestionId	Q1003
1	Result	Successful
1	SurveyETag	6.36827229492480992
1	SurveyVersion	ETag(636827229492480992)
1	TestInterview	FALSE
1	EndReason	18
1	InterviewQuality	1

**LastSeenQuestionId:** the id of the last question viewed by the respondent This helps to analyze survey dropouts at the respondent level.

**EndReason** refers to the survey response codes.

**Result** is the code description for the code in EndReason.

The following paradata parameters appear if an interview was reset (**WasResetTo** value appears and is set to the value of the InterviewId of the replacement interview) or the interview data was deleted (**IsDeleted** appears and is set to TRUE). These parameters only appear when dropped out interviews are included in the download package.

1	WasResetTo	10
2	IsDeleted	TRUE



7.2.4 Paradata for the Quota Out (both Online and CAPI Surveys)


When a *\*STRAT* command check shows that stratum reached its maximum target and the responded is stratted out, we will now add a line to the paradata that describes the quota cells relevant for this interview, starting from with the total target, and then following the branch of the tree up to the first quota cell that fails the *\*STRAT* check. Only the result of the last failed *\*STRAT* check is stored.

Example

Let's assume we have the following quota frame:

Quota overview

Select Version June 20th, 2024 at 10:34 AM (latest)  

☐ Limit overshoot of active interviews 

	Quota overview	Minimum target	Maximum target	Successful interviews
Total target:			3	0
Region				
North			2	0
Drink				
Tea			1	0
Coffee			2	0
South			2	0
Drink				
Tea			1	0
Coffee			2	0

version etag: 638544692922581172

The first interviewer will answer region: *North* and drink: *Tea*. This interview will be completed successfully.

The second interviewer will also answer region: *North* and drink: *Tea*. This interview will fail the \*STRAT because the target for *Tea* in *North* is 1 which is already met because of the first interview.

In the paradata the following entry will be made:

```
2      StratificationRejectionReason
{Outcome:"NotAllowed", "Context":[{"Variable":"Drink", "Level":"Tea"}, {"Variable":"Region", "Level":"North"}]}
```

Let's say the third interviewer answers region: *North* and drink: *Coffee*. This is allowed because the max target for that cell is 2, which it has not reached yet.

If the fourth interviewer answers region: *North* and drink: *Coffee*, the interview will not pass the quota check. Not because *Coffee* in *North* has reached its target, but because region: *North* has reached its max target (since we have two completed interviews for region *North*).

In the paradata the following entry for this interview will appear:

```
4 StratificationRejectionReason {Outcome:"NotAllowed", "Context":[{"Variable":"Region", "Level":"North"}]}
```

You can see here that the context does not include the drink level, as the quota evaluation already failed on the parent level (region: *North*).

Let's say the fifth interviewer answers region: *South* and drink: *Tea*. This is allowed and the interview will be completed successfully.

The sixth interviewer will also answer region: *South* and drink: *Tea*. But this time it will fail, as the total target on the survey has been reached.

This interview will have the following entry in the paradata:

```
6 StratificationRejectionReason {Outcome:"NotAllowed", "Context":[]}
```

It has no context, because the evaluation already failed on the target set in the survey.

## 7.3 Audit Trail

### 7.3.1 Introduction to Audit Trail

This feature is designed to help users get more insights in what the respondent has done during an interview.

- The minimum version of CAPI app is 2.1.
- The Audit Trail logs will be delivered in the csv format with the following name:  
[surveyname]-auditlog.csv
- The Audit Trail logs will be downloaded together with the Paradata file (when Paradata option is selected in the *Data Download* dialog – see previous chapter).

### 7.3.2 Overview of an Audit Trail file

Below is an example of a downloaded interview Audit Trail log:

Interview number	Sequential Order	Question Id	ElapsedTime	NavigationAction	QuestionType	QuestionVariable	ActionDateTime
1	1	P228	2.48	Next	Page	Household	2020-03-17T14:52:23.5934980Z
1	2	Q10	4.81	Next	Question	Status	2020-03-17T14:52:28.5195050Z
1	3	Q20	4.61	Next	Question	Wealth	2020-03-17T14:52:33.2808470Z
1	4	R1Q2404	3.45	Back	Matrix	Household	2020-03-17T14:52:36.8565360Z
1	5	Q20	4.77	Next	Question	Transport	2020-03-17T14:52:41.7367750Z

Location	Versions	ExternalId
{"latitude":"53.50325","longitude":"6.06631","accuracy":18,"IsGeolocationEnabled":true}	{"Engine":"23.13.0.0","Parser":"1.16.017"}	Q1
{"latitude":"53.50325","longitude":"6.06631","accuracy":18,"IsGeolocationEnabled":true}	{"Engine":"23.13.0.0","Parser":"1.16.017"}	
{"latitude":"53.50325","longitude":"6.06631","accuracy":18,"IsGeolocationEnabled":true}	{"Engine":"23.13.0.0","Parser":"1.16.017"}	
	{"Engine":"23.13.0.0","Parser":"1.16.017"}	Q4
	{"Engine":"23.13.0.0","Parser":"1.16.017"}	

**Interviewnumber** - number of the interview. The log file is ordered on this column first and then on SequentialOrder.

**SequentialOrder** - the position of the action taken (by the respondent) from beginning to end. The sequential number reflects in which order the question were shown to this particular respondent in the interview. The number might not be the same for this question in another interview (with a different respondent, who is shown questions in a different order).

**QuestionId** - stores the *QuestionId* from the script:

- Q: We store the *QuestionIds* (as Q) only of the questions that are shown to and can be filled in by the respondents (so, for example, \*DUMMY questions are not shown). The number after Q is the question number in the ODIN script.
- P: A \*PAGE is also stored (as P). A number is assigned to each page.
- R: For a repeat question or a \*MATRIX (stored as R). In case of a \*MATRIX, we are storing the last question number within the matrix or block that a respondent could have filled in.
- S: for a subroutine. The number next to S (such as S1) shows which time this subroutine has appeared in the script, and not the order in which the respondent has seen this subroutine. So S1 means this sub has appeared the 1<sup>st</sup> time in the script, S2 – the 2<sup>nd</sup> time in the script, even though for this particular respondent this could be the 1<sup>st</sup> time this sub is called, etc.

As an example, in this script snippet we store Q2404:

```
*MATRIX 3 Q230W *FIELD 70L42 *UIRENDER "Mixed"
*? v240label

*QUESTION 2401 *ALPHA 1L12
Name

*QUESTION 2403 *CODES 13L1 *UIRENDER "Select"
Age Range
1:<18
2:18-24
3:25-34
4:35-44
5:45-54
6:55-64
7:65+

*QUESTION 2404 *CODES 14L1 *UIRENDER "Horizontal"
Driver's license
1:Yes
2:No

*ENDMATRIX
```

### ElapsedTime

This is the time in seconds with 2 decimal places.

- For a suspend via script we hardcode a 0 for *ElapsedTime*.
- For a suspend via app Back button option we store the actual screen time.

**Please note** that if you sum up all the elapsed times and compare them with the total time of the interview (obtained as a difference between the InterviewStartTime and InterviewEndTime from paradata), the total time of the interview will be a little bit higher because it also includes loading time of the pages and processing that takes place between questions.

### NavigationAction

The action done by the interviewer to move to the next page:

- Pressed Next button (Next)
- Pressed Back button (Back)
- Suspend (interview is suspended via the app's Back button or via script).

**QuestionType**

The following question types are recoded:

- QUESTION (ALPHA/NUMBER/CODES/OPEN/FORM)
- PAGE
- BLOCK
- MATRIX
- There is a special case when an interview gets suspended via script: in that case the *QuestionType* will be empty.

**QuestionVariable**

If you are using the command LABEL in your questionnaire it will fill this column with the value you have put there for the specific question.

**ActionDateTime**

The time stamp of the action, in UTC. The format is the same as date/time in the paradata file. If you would like to get a complete picture of your data quality, please also look at the paradata, which also contains the local time stamp and the location fix for each interview.

**Location (CAPI surveys only)**

GPS fix (location) of the question. It is stored if the survey manager turns on continuous location tracking setting for a CAPI survey in Nfield Manager.

**Versions**

Versions of the **engine** and the **parser**.

- The **engine** is the Nfield logic engine that runs the interview. It presents the first question, and then depending on the script and the answers, calculates what to show next, etc.
- The **parser** is the syntax checker that is used in the Odin Developer and the Nfield Manager.

**ExternalQuestionId**

Question identifier that was added using an [\\*ID](#) command.

**7.3.3 Suspending an Interview**

There are two ways of suspending an interview during fieldwork. They will result in different entries in the Audit Trail log.

**1. Suspend via the app's Back button**

We store an entry in the audit log when user clicks on "Save and Suspend". Once the user resumes the interview, the stopwatch will start counting again. We won't store the time between the suspension and the resume. Here is an example of an scenario and the stored Audit log:

- Start an interview. You are on Q50.

- Wait 4.23 seconds.
- Click *Next*. You are on R5Q701.
- Wait 8.31 seconds.
- Suspend (using the app *Back* button). You are outside the interview.
- Wait 10 minutes.
- *Resume*. You are on R5Q701.
- Wait 7.69 seconds.
- Click *Next*. You are on Q3 (next question).

Interview number	SequentialOrder	QuestionId	ElapsedTime	NavigationAction	QuestionType	ActionDateTime
1	8	Q50	4.23	Next	Block	2020-03-17T14:52:56.9594840Z
1	8	R5Q701	8.31	Suspend	Matrix	2020-03-17T14:53:05.3915940Z
1	8	R5Q701	7.69	Next	Matrix	2020-03-17T15:04:16.9738380Z

## 2. Suspend via script

When you suspend via script we store two actions:

- "Next", because they clicked next button.
- And "Suspend", because the interview is suspended.

As the suspend is triggered automatically immediately after the Next, the elapsed time would be less than 1 second. So the value will be hard-coded as 0. Here is an example of an scenario and the stored Audit log:

- Start an interview. You are on Q50.
- Wait 4.23 seconds
- Click Next. You are on R5Q701.
- Wait 8.31 seconds.
- Select in the questionnaire the response code which allows appointment. You are outside the interview.
- Wait 10 minutes.
- Resume. You are on R5Q701.
- Wait 7.69 seconds.
- Click Next. You are on Q3 (the next question).

Interview number	SequentialOrder	QuestionId	ElapsedTime	NavigationAction	QuestionType	ActionDateTime
1	8	Q50	4.23	Next	Block	2020-03-17T14:52:56.9594840Z
1	8	R5Q701	8.31	Next	Matrix	2020-03-17T14:53:05.3915940Z
1	8	R5Q701	0	Suspend		2020-03-17T14:53:05.3915940Z
1	8	R5Q701	7.69	Next	Matrix	2020-03-17T15:53:16.9738380Z

Note that the *QuestionType* is empty in this case, and the elapsed time is hardcoded with zero.

### 7.3.4 Killing the Nfield CAPI app During Interview

For a hard quit (application killed or crashed), we don't add the entry for "*Suspend*". The elapsed time between the previous action that was triggered and the time the application was killed will be lost. If the interview has the setting "*Autosave Interviewing*" enabled, the interviewer will be able to restore the interview. In this case the interview will be restarted from the last selected question (one screen earlier than in case of a suspension with the exit dialog). Once the interview is restored, the stopwatch will start counting again. Here is an example of an scenario and the stored Audit log:

- Start an interview. You are on Q1.
- Wait 2 seconds
- Click *Next*. You are on Q2.
- Wait 3 seconds.
- Hard kill the application. You are outside CAPI app.
- Wait 10 minutes.
- Restart the CAPI client and restore the interview. You are on Q1.
- Wait 4 seconds.
- Click *Next*. You are on Q2.

Interview number	SequentialOrder	QuestionId	ElapsedTime	NavigationAction	QuestionType	ActionDateTime
1	1	Q1	2	next	Block	2020-03-16T10:54:48.1238970Z
1	2	Q1	4	next	Block	2020-03-16T11:04:45.1259770Z

**Note:** For more detailed information on the Audit Trail feature, you can watch an Academy training session at [https://youtu.be/Ac\\_sYW5xKCM](https://youtu.be/Ac_sYW5xKCM).

## 7.4 Sample Table

### 7.4.1 Introduction to Sample Table

In Nfield you can upload a sample record file both CAPI and Online surveys. For each sample record, you can specify variables that you would then pre-fill before the survey starts, and use during the interview.

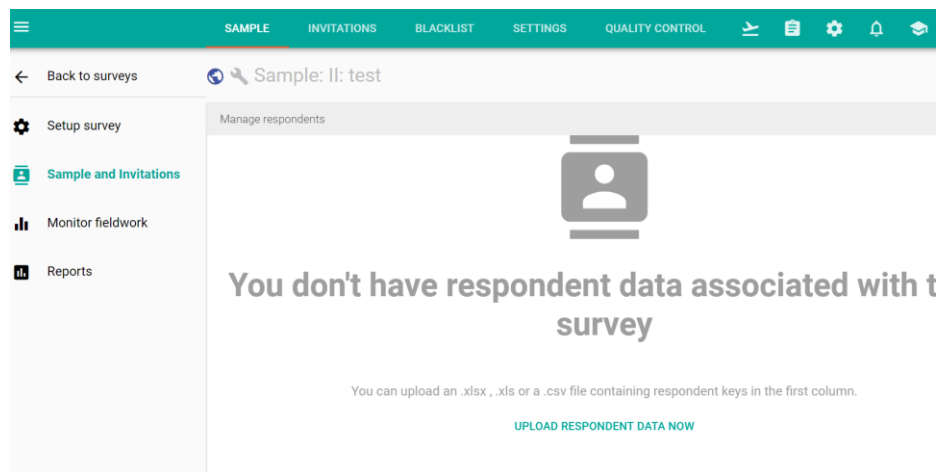
You can specify up to 800 variables for each sample record.

You need to save sample record file as an .xlsx, .xls or a .csv. This file should contain respondent keys (RespondentKey field) in the first column. All the other variables should be stored in columns named the same as the variables in the ODIN script.

Below is an example of a sample record file. The ODIN script for this survey could have variables FirstName, LastName, Email, or Age, but will not necessarily have (all of) these.

RespondentKey	FirstName	LastName	Email	Age
10	Mickey	Mouse	<a href="mailto:m.mouse@wdisney.com">m.mouse@wdisney.com</a>	25
20	Minny	Mouse	<a href="mailto:mi.mouse@wdisney.com">mi.mouse@wdisney.com</a>	23
30	Duffy	Duck	<a href="mailto:d.duck@wdisney.com">d.duck@wdisney.com</a>	27

This file can be then uploaded to your survey in the survey's Sample and Invitations/Sample tab:



One can perform the following actions on imported sample records (both on individual records, filtered subsets or all the records): Clear Profile, Block, Reset. Delete can also be performed, but see details below.

Interview Id	Respondent Key	Response Code	Result	Last Activity	Last Invitation Status	Invitation Count	Invitations
00000001		21	Screened out	8 days ago	No invitation sent	0	
00000002		18	Successful	8 days ago	No invitation sent	0	
00000003		18	Successful	8 days ago	No invitation sent	0	
00000004	11	0	Not used		Opened	1	sf
00000005	99j	0	Not used		No invitation sent	0	
00000006	101bcd	0	Not used		No invitation sent	0	
00000007	223	0	Not used		No invitation sent	0	
00000008	224	0	Not used		No invitation sent	0	
00000009	35	18	Successful	8 days ago	No invitation sent	0	

1. **Clear Profile:** clears all the Personally Identifiable Information (PII) data for the selected record(s) from the database. This is a way to keep the interview(s), but to anonymize the survey.
2. **Delete:** delete the interview as well. The whole interview data gets deleted, not only the PII data for it. There will still be sample record left evidencing that there was an interview, with an interview status, and the result *Deleted*. If you do it on a single record it will delete all the interview data, but if you do it on the subset of records or on all records, it will only delete records that are unused. If your selection contains even 1 used record, it will not delete anything.
3. **Reset:** creates a new record (a copy of the record) with a blank state (*Not Used*), and the interview can be still run for this respondent key. The old record stays, with status *Reset*. One can only reset records that are in a definite state (successful or screened-out interviews) and have a respondent key. To protect you from accidentally resetting an appointment, you cannot reset a dropped-out interview.
4. **Block:** you can block certain records, and if one tries to run an interview on the blocked respondent keys, the system will not allow it giving a message that the respondent is blocked. You can still send invitations to the blocked respondent.

### 7.4.2 Rules for Sample Table Headers

It is not allowed to create custom fields (sample table headers) with the same name as system fields (minus the TT):

- TTInterviewNumber
- TTStatusCode
- TTAvailabilityCode
- TTID
- TTClosedAnswerDataInBlob
- TTOPenAnswerDataInBlob
- TTContactTime
- TTLanguage
- TTDataUploadTime
- TTIsCounted
- TTSamplingPointId
- TTAddressId
- TTInterviewQuality
- TTRespondentKey
- TTResetRespondentKey
- TTETag
- TTImportDate
- TTImportedBackgroundData
- TTEmailUnsubscribed
- TTIsDeleted
- TTAddressDetails
- TTLocationTracking
- TTTelephoneNumber
- TTBlockedUntil
- TTUserModificationDate
- TTStartLink

So, for example, for a language column header, please do not use “Language” as it is the same as “TTLanguage” minus “TT”, but use “Lang” or “UserLanguage” or similar.

Also, column names that start with “TT”, special characters or size greater than 100 characters are not allowed.

The delimiting character for the sample columns should be unique and the heading line should be present. Only comma, semi colon and tab characters are allowed as delimiters, and only one of them can be used in a particular spreadsheet. This applies to both the sample uploaded using the Nfield Manager as well as through the API.

System columns are case sensitive. So 'AddressDetails' will map to the TTAddressDetails system field, while 'addressDetails' for instance does not. Non-system columns are case-insensitive. There cannot be both a column 'MyColumn' and a column 'myColumn' in the sample table. Both columns map to the same ODIN script variable '\*VAR mycolumn', or '\*VAR MyColumn', or '\*VAR myColumn', or '\*SAMPLEDATA MyCoLuMn'.

**Note:** A special case. If you upload sample with a column called TelephoneNumber (must match exactly), it is written to TTTelephoneNumber, and you must use TTTelephoneNumber in the script. For CATI this column is required when uploading sample.

### 7.4.3 Dealing with Customer Personally Identifiable Information (PII) data

If you want to be able to delete Personally Identifiable Information (PII) data independent from the other interview data (the non-PII interview data), for instance, to anonymize surveys in order to meet EU GDPR PII regulations, you need to store this PII data ONLY in the sample table.

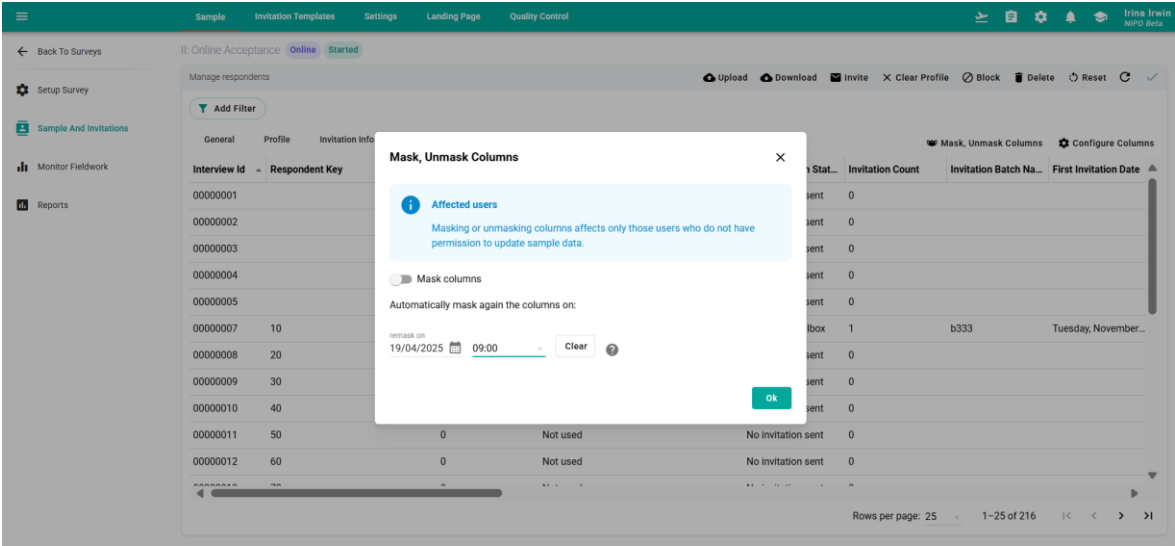
In the NIPO ODIN Scriptor Basic Course we show how to script this.

We also have the **Sample Data Obfuscation** feature to protect the PII data inside the Nfield Manager. It works as follows:

- **Default Protection:** Users with the SampleRecord.read permission will not see the original sample data in the sample table. Instead, they will be shown obfuscated text represented by a string of four characters, "xxxx," to ensure that sensitive information is kept secure.
- **Requesting Access to View Data:** If a user with SampleRecord.read permission needs to view the actual sample data, they can submit a request to a user with SampleRecord.write permission. This user will have the ability to unmask or reveal the original sample data.

You can unmask sample data for a specific number of days using a calendar setup:

1. Go to the Nfield Manager's *Sample* tab and click on the *Mask, Unmask Columns* button.
2. Set the *Mask columns* toggle off (to unmask).
3. Choose a date and time to mask the sample data again from that specific date on.



For more information on how NIPO meets GDPR requirements, please see the NIPO Academy video [here](#).

#### 7.4.4 Blacklist

For Nfield Online surveys you can maintain a blacklist. This is a list of respondents that have opted-out from participating in your surveys. Nfield allows you to list up to 1 million respondents in your blacklist. Please contact NIPO support at [helpdesk@nipo.com](mailto:helpdesk@nipo.com) if you have a requirement to exceed these limits.

#### 7.4.5 Locations of Nfield Services

Below are the storage locations in Azure Cloud of Nfield Services per geographical area at the time of the writing.  
We have primary storage location and geo-replication location (for back-ups).

##### \*\*\*\* EU deployment \*\*\*\*

Primary: EU west (Netherlands)  
Secondary: EU North(Ireland)  
PowerBi (reporting): EU west (Netherlands)

##### Local storage UK

Primary: UK south (London)  
Secondary: UK west (Cardiff)

PowerBi(reporting): N/A

**\*\*\*\*\* AM (Americas) deployment \*\*\*\*\***

Primary: East US (Virginia)  
Secondary: West US (California)  
PowerBi(reporting): East US(Virginia)

**Local storage Canada**

Primary: Canada Central (Toronto)  
Secondary: Canada East (Quebec)  
PowerBi(reporting): N/A

**\*\*\*\* APAC (Asia Pacific) deployment \*\*\*\***

Primary: East Asia (Hong Kong)  
Secondary: South East Asia (Singapore)  
PowerBi(reporting): South East Asia (Singapore)

**Local storage Singapore**

Primary: South East Asia (Singapore)  
Secondary: N/A (only 1 datacenter in Singapore, so no geo-replication available)  
PowerBi(reporting): South East Asia (Singapore)

**Local storage Australia** (for Australia and New Zealand).

Different datacenters for geo-replication of storage and databases due to (un)availability of some services in the different datacenters.

Primary: Australia Central (Canberra)  
Secondary storage: Australia Central2 (Canberra)  
Secondary Db: Australia East (Victoria)  
PowerBi(reporting): N/A

**\*\*\*\* CHINA deployment \*\*\*\***

Primary: China North (Beijing)  
Secondary: China East (Shanghai)  
PowerBi (reporting): N/A

**7.4.6 Local Data Storage Possibilities**

To comply with some countries' data storage rules, Nfield offers a possibility to store sample and survey data locally (inside the country), instead of in the Cloud. Local data storage options and their costs depend on the local circumstances. For more information please see this [NIPO blog item](#).



## 8. Interview Simulator

Interview Simulator feature in the Nfield Manager allows you to generate simulated random data for the survey, so that you can check if the questionnaire script and the data are valid prior to starting fieldwork.

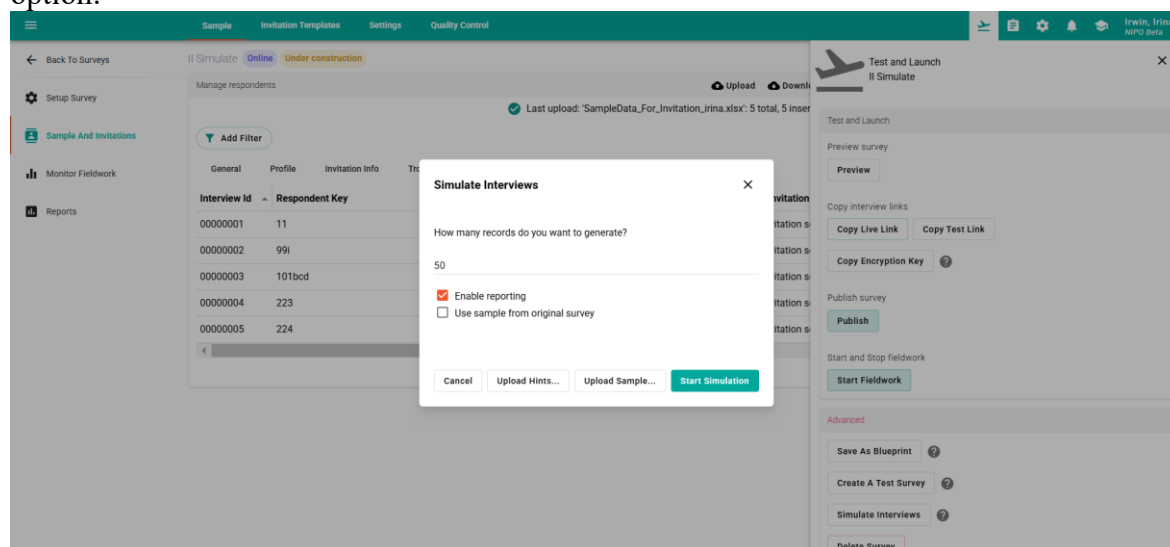
*How is this feature different from NIPO ODIN Developer's dummy data generation?*

The Interview Simulator supports quotas, stratification, \*GETLFLQLIST, reports, and other functionalities, some of which are not available in the NIPO ODIN Developer. With interview simulator hints you can steer randomization. Moreover, unlike the ODIN Developer's dummy data, the Interview Simulator runs within the Nfield Manager.

### 8.1 Running an interview simulation

Running simulations (simulated interviews) is only possible for surveys that have not been started yet, but have a published questionnaire.

For such a survey, on the *Test and Launch* tab, we can choose the *Simulate Interviews* option:



On this page you can say how many simulated interviews you would like to generate, if you would like the reporting to be also enabled, and if you want to use the pre-uploaded sample from the survey. You can also upload the sample here that will be only used for the simulation (will not be uploaded to the actual survey).

After clicking on *Start Simulation*, the simulation starts running, and its progress is shown in the *Activities* tab. When done, you can click on *View Simulation* in the *Activities* tab to be taken to the results. Your original survey is still in the same state (not started, no interviews have been run), and a copy of the survey with the simulated run has been placed into the tab *Tests* where you can look at it:

The screenshot shows the Interview Simulator interface. The top navigation bar includes tabs for Surveys, Groups, Invitations, Blacklist, Access, Interviewers, Repositories, Monitoring, and API. Below this, there's a sub-navigation bar with List, Trackers, Manage, Blueprints, Tests, and Search Respondents. The main content area displays a table of surveys:

Survey name	Created
II Simulate-simulation <span>Online</span> <span>Simulation</span>	1 minute ago
II P1S2-simulation <span>Online</span> <span>Simulation</span>	5 days ago
II P1S2-test <span>Online</span> <span>Manual</span>	5 days ago

On the right sidebar, there's a section for 'Activities' for user 'Irwin, Irina'. It shows a list of running and completed activities:

- Requested Interview simulation for survey 'II Simulate'
- View Simulation (less than a minute ago)
- Preparing fieldwork overview report (Requested fieldwork progress for 'II:P2S2' succeeded (over 2 years ago))

**Note:**

- Please remember that when running simulations, you are still using the same system resources as are used to run real interviews. So please be responsible about how many simulations you create.

**Questions and answers**

- How many interviews can an interview simulator generate at once?*  
We are allowing max 500 interviews per run.
- For a particular survey, how many times can you run an interview simulator?*  
There are no limitations per survey, but only the data and simulator survey for the the last saved. Each new run you do removes the previous run. So, only the most recent copy of survey and data is stored.
- On which surveys can I run the interview simulator?*  
Only on the surveys that are published, and for which fieldwork has not been started.
- How to run the simulation on surveys for which the fieldwork has already started?*  
You need to stop the fieldwork to run the simulation.
- Would quotas and stratification work?*  
Yes.
- Can we download the data from the interview simulator survey?*  
Yes- from the tab Monitor fieldwork->Progress->Download data or through the API.
- Are the results of an interview simulator survey available in reports?*  
Yes, if you have checked that option on creating the simulation.
- Can we run an interview simulator survey based on sample uploaded?*  
Yes, either using the original sample from the script, or uploading specific sample for the simulation (in the Simulate Interviews dialog).
- How long is the interview simulator survey is available?*

An interview simulator survey will be deleted 10 days from the creation date of simulation survey.

10. *Is simulated interviews available for CAPI surveys?*

Yes, you can create interview simulations for the CAPI surveys, but like all test interviews for CAPI started in the Nfield Manager, they will run online.

## 8.2 Hints

### 8.2.1 What are hints for interview simulation?

Hints are an external file to provide predefined answers.

For example, if you have a screener question where the majority of the options will lead to screen-outs, you can use hints to make sure that a decent amount of interviews will still pass this question.

To do that, you can specify the predefined responses using the Hints file.

- Hints (an external file) is optional.
- All question types are supported by hints.
- Hints are used to steer the simulated interview in a more intelligent way.
- The simulation will generate the answers based on the inputs specified in the hints file.
- If no hints are provided for a question, the answers will be randomly generated.

### 8.2.2 When to use Hints?

1. If the questionnaire script contains specific validation that a random generator will hit upon very seldom. For instance, an email address validation.
2. If you want to specify any predefined responses to control the routing.

### 8.2.3 Example

JSON Schema for the hints, and an example hints file

JSON schema	JSON example
<pre>namespace Nfield.Manager.Surveys.Interactions.Survey.Simulations {     internal static class HintsValidationSchema     {         public static string Schema =&gt; @"</pre>	<pre>{   "hintsName": "Demo",   "hints": [     {       "questionId": "Q1",       "answers": [</pre>

Q1 is a multi-choice question.

```

"$schema": "http://json-schema.org/draft-07/schema#",
"title": "Interview Simulator hints definition",
"type": "object",
"additionalProperties": false,
"properties": {
  "hintsName": {
    "type": "string"
  },
  "hints": {
    "type": "array",
    "items": {
      "type": "object",
      "additionalProperties": false,
      "properties": {
        "questionId": {
          "type": "string"
        },
        "noAnswer": {
          "type": "integer",
          "minimum": 0,
          "maximum": 100
        },
        "button": {
          "type": "integer",
          "minimum": 0,
          "maximum": 100
        },
        "buttonAnswers": {
          "type": "array",
          "items": {
            "type": "object",
            "additionalProperties": false,
            "properties": {
              "answer": {
                "type": "integer"
              },
              "weight": {
                "type": "integer",
                "minimum": 1
              }
            },
            "required": [
              "answer",
              "weight"
            ]
          }
        },
        "maxVisits": {
          "type": "integer",
          "minimum": 1
        }
      }
    }
  }
}

```

```

{
  "answer": "1,2",
  "weight": 100
},
{
  "answer": " 3",
  "weight": 50
},
{
  "answer": "4",
  "weight": 50
}
]
},

```

Q2 is a single-choice

```

{
  "questionId": "Q2",
  "answers": [
    {
      "answer": "1",
      "weight": 100
    },
    {
      "answer": " 2",
      "weight": 50
    },
    {
      "answer": "3",
      "weight": 50
    }
  ]
},

```

Q3 is a numeric question.

```

{
  "questionId": "Q3",
  "answers": [
    {
      "answer": "10",
      "weight": 25
    },
    {
      "answer": "20",
      "weight": 25
    },
    {
      "answer": "40",
      "weight": 25
    }
  ]
},

```

<pre>       ""excludedAnswers"": {         ""type"": ""string""       },       ""answers"": {         ""type"": ""array"",         ""items"": {           ""type"": ""object"",           ""additionalProperties"": false,           ""properties"": {             ""answer"": {               ""type"": ""string""             },             ""weight"": {               ""type"": ""integer"",               ""minimum"": 1             }           },           ""required"": [             ""answer"",             ""weight""           ]         }       },       ""required"": [         ""questionId""       ]     }   },   ""required"": [     ""hintsName"",     ""hints""   ] };  } } </pre>	<pre>       {         "answer": "60",         "weight": 25       }     ]   ] } } </pre>
---	---

### 8.2.4 Properties

Name	Description	Type	Additional information
hintsName	The name to identify the hints.	String	This property is required.
Hints	An array of hint objects. Where each element describes a single question hint.	Array	This property is required.
questionId	This is a unique identifier of a	String	This property is required.

	question as it is defined in the questionnaire script.		
noAnswer	Specifies a percentage of non-response answers. Possible value is from 0 to 100.	Number	This property is optional. This can be used on a question which is non-mandatory(*NON). min=0 max=100
Answers	An array of answer objects. The answer object has an answer definition and a weight property.	Array	This property is required.
Answer	The answer definition. This can be a single answer or an answer group or an answer range.	String	<p>This property is required.</p> <p>-----</p> <p>A single answer is a number or a text that specifies the desirable answer. For example: <i>"answer": "3" is an answer to a numeric question or to a category question, depending on question type.</i> <i>"answer": "text" is an answer to an alpha or open question.</i></p> <p>-----</p> <p>An answer group is a sequence of answers separated by comma. The group is only allowed for category question type which is a multi-choice.</p> <p><i>"answer": "2,3,5" selects multiple categories.</i></p> <p>-----</p> <p>An answer range is a continuous sequence where the start and the end are specified. A dash character is used as a range indicator. The range is only allowed for category question type which is a multi-choice.</p> <p><i>"answer": "3-7" selects a category range.</i></p>
Weight	The weight of the answer. This is a number value to specify a probability of the answer.	Number	This property is required.

## For more information

For more information on this feature, please watch our [NIPO Academy session 46](#).

### 8.2.5 Excluding specific response codes

```
{
  "hintsName": "hints name",
  "hints": [
    {
      "questionId": "Q1",
      "excludedAnswers": "3,4"
    }
  ]
}
```

Question id used in ODIN script

Response codes that need to be excluded

## 8.2.6 Total sum value for the question

It is possible to specify the total sum value for the question in the Hints file. The values are distributed randomly for each row until they add up to the total sum specified in the Hints.

```
{
  "hintsName": "hints name",
  "rules": {
    "singleQuestionSum": [
      {
        "questionId": "Q559",
        "sum": 10
      }
    ]
  }
}
```

→ Question id used in ODIN script inside a Matrix

→ Total Sum Value



# 9. Default Template for Nfield System Rendering Options

The default template for Nfield comes with a small but useful set of rendering options that are available in all templates. These options have been built-in because they require access to specific hardware controls on the tablet or mobile phone. The system rendering options support the following features:

- Capturing photos and using the device's camera.
- Capturing the respondent's answer to a question using the device's microphone.
- Play a sound, music or video clip using the device's media player.

This section briefly describes how to use these features.

**Please note** that templates may apply additional options to features; refer to the template documentation for more information.

## 9.1 Capture Photo

### Purpose

Take a picture using the device's camera. This may be used for any purpose, such as validating that the interview was taken with a genuine respondent, see a transport ticket that a respondent is carrying, or check the products that a customer has taken from a store.

### Description

The interviewer can tap the picture image, which opens up the camera. The picture taken is automatically saved with the respondent data. This is an option specific to the \*OPEN question type. Files are stored using the name <interviewnumber>\_<question>.jpg. For easy reference, this filename is also listed as the open-ended answer.

### Syntax

```
*UIOPTIONS "type=capture;capture=photo;photo-quality=high;take-label=<text>;retake-label=<text>"
```

### Rendering options

Name	Description	Example
capture	Set capture type. Currently only photo is supported.	capture=photo
retake-label	Label on the button that starts the camera, if a picture has previously been taken	retake-label=Retake picture
take-label	Label on the button that starts the camera	take-label=Take picture

photo-quality

Quality of the actual photo: low, medium, high, original

photo-quality=low

#### Remarks

- Photos are taken using the image quality and resolution settings configured on the device's camera, and that depending on the range of devices and configurations in the field you may receive pictures of various sizes. Your domain administrator needs to ensure the picture resolution settings of interviewer devices are set to an acceptable level.
- Photos may be large and take the interviewers a long time to upload. Take care to limit the number of photos taken in a single questionnaire.
- Capturing photos is not possible when testing the questionnaire on a browser on a desktop computer.
- Photo quality is dependent on the device. The "original" setting is the device's own setting. If you're working with a device with a fantastic HD camera, the "original" setting will render a much better quality than "high". The photo-quality setting is a good way to keep (photo) filesize down. Remember: all photos will have to be uploaded when doing a full sync with the Nfield servers. At present the default setting is "high".

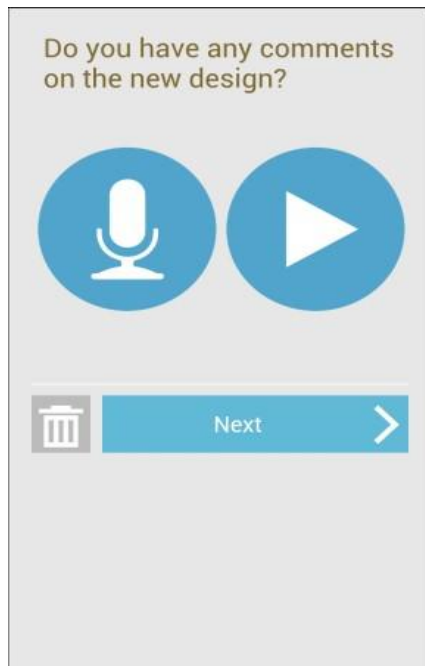
#### Example

\*QUESTION 10 \*OPEN 61L1 \*UIOPTIONS "type=capture;capture=photo;take-label=Take picture;retake-label=Take picture again"  
For interview validation purposes, can we please take a picture of your boarding pass?

#### 9.1.1.1 See Also

*OPEN (question type) .....	203
CAPTURE AUDIO .....	331
PLAY MEDIA .....	333
SILENT RECORDING .....	341

## 9.2 Capture Audio



### Purpose

Records audio (voice response) for a question. Voice response is preferred for open-ended answers because most mobile devices are not ideal for entering open-ended answers.

### Description

This option displays a record toggle button allowing the interviewer to select when audio recording should be started and stopped. Recording a question for the second time replaces the previous recording. There is also a playback toggle button which can be used to verify that the recording is adequately audible.

If the recording is intended to replace a typed open-ended answer, it is recommended to use `*NON` on the question. This allows both keyboard and voice input at the same time.

Audio clips are included in the download result data as `MPEG3` files. You may need to download a suitable codec or player for your operating system to be able to listen to the audio clips. We currently suggest using Apple QuickTime because it natively supports `MPEG3`.

The file format of the audio clips is

```
[interview number]_[question reference].mpeg3
```

For example, the recorded answer to question 6000 for interview number 2 would be called

00000002\_q6000.mpeg3. Note that interview numbers are always listed as eight-digit numbers.

### Applies to

For open\*`QUESTION` type only.

Syntax

```
*UIOPTIONS
"type=capture;capture=audio[;maximum-duration=<duration>]"
```

Rendering options

Name	Description	Example
maximum-duration	Sets a maximum time length for the recording, in seconds. The recording is truncated to fix the maximum.	max-duration=30

Remarks

- Capturing audio is not possible when testing the questionnaire on a browser on a desktop computer.
- Recording can be done on individual questions (respondent input) or the entire interview can be captured (interviewer and respondent)

This latter option is called Silent Recording.

Example (Capture audio on a single question)

```
*QUESTION 6000 *OPEN L1 *MULTI *UIOPTIONS "type=capture;capture=audio; maximum-duration=30" *NON"
Do have any comments on the new design?
```

9.2.1.1 See Also

\*OPEN (question type) ..... 203

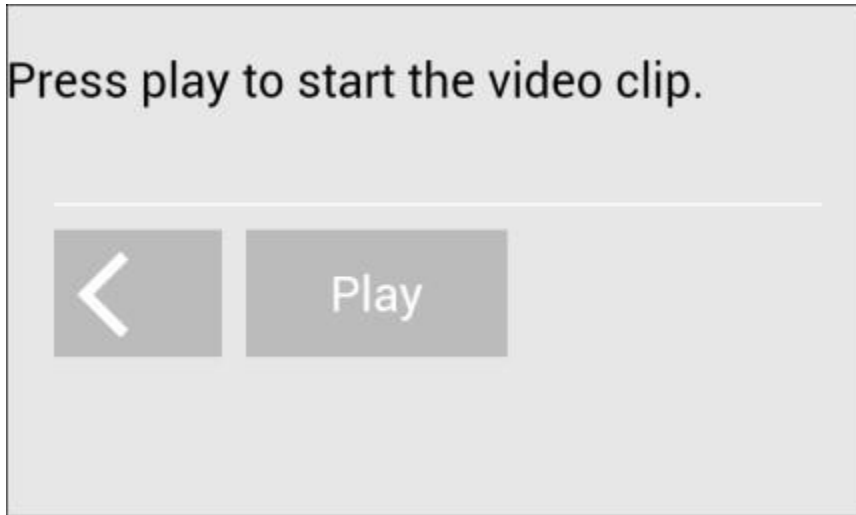
CAPTURE PHOTO..... 329

PLAY MEDIA ..... 333

SILENT RECORDING ..... 341



## 9.3 Play Media



### Purpose

Displays or plays multimedia such as movie clips or sound during a question.

### Description

Media playback is done using the device's designated playback app for this media format. The file types that may be used for playback depend on the device on which playback is executed, and on the operating system installed on the device. Recommended formats supported by most devices for audio and video playback are \*.MP3 and \*.MPEG4. For example, movie clips start a video player while sound clips start a sound player. Media playback can also display images of various formats. If the media clip is a video, playback may overlap the actual question and the question may not be visible until the playback app is closed.

When this rendering option is used, the selected `scenario` decides how media is presented and optionally introduced. Currently only one scenario type is supported.

### Applies to

Any `*QUESTION` type.

### Syntax

```
*UIOPTIONS "type=play;media=[file] [;scenario=<type>]
[;intro-text=<introduction text>] [;minimum-duration=<duration>]"
```

### Rendering options

`intro-text`

Text to display next to the play button when using the `MediaWithIntro` scenario.

**Example:** `intro-text=Press play to start the video clip.`

#### media

The full filename of the media file to be played. No path should be specified. The file to be played must have been uploaded to the survey prior to deploying the survey.

**Example:** advertorial.mp4

#### minimum-duration

The minimum waiting time until the question text is made visible, in seconds. Nfield ODIN cannot enforce playing the entire media clip (the interviewer may interrupt playback to return to the interview) but with this setting the question may be hidden for a given amount of seconds. Set to -1 to make this equal to the clip duration.

**Default:** -1

**Example:** minimum-duration=10

#### Scenario

Specifies how the media clip and accompanying screens are displayed. Currently only supports

**MediaWithIntro:** an introduction text is displayed, and the interviewer must press a play button.

**Example:** scenario=MediaWithIntro

#### Example

```
*QUESTION 6000 *CODES L1 *UIOPTIONS "type=play;media=advertorial.mp4;scenario=MediaWithIntro;intro-text=Press play to
start the video clip.;minimum-duration=-1"
Have you previously seen the advertorial that was just shown?
1: Yes
2: No
3: Don't remember
```

#### See Also

CAPTURE PHOTO.....	329
CAPTURE AUDIO.....	331
SILENT RECORDING.....	341

# 10. Only Relevant for Online Surveys

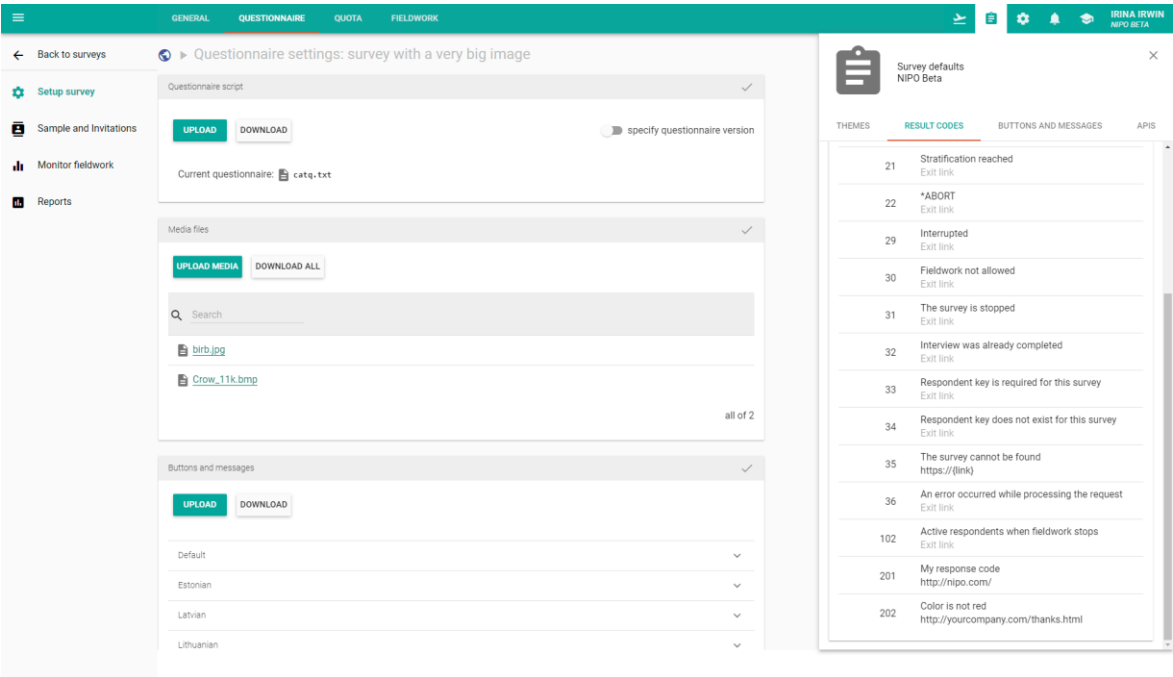
## 10.1 Exit Links

For each result code of the interview, you can specify an exit link (URL) that redirects the respondent to another page when he/she exits the interview. These links are called exit links.

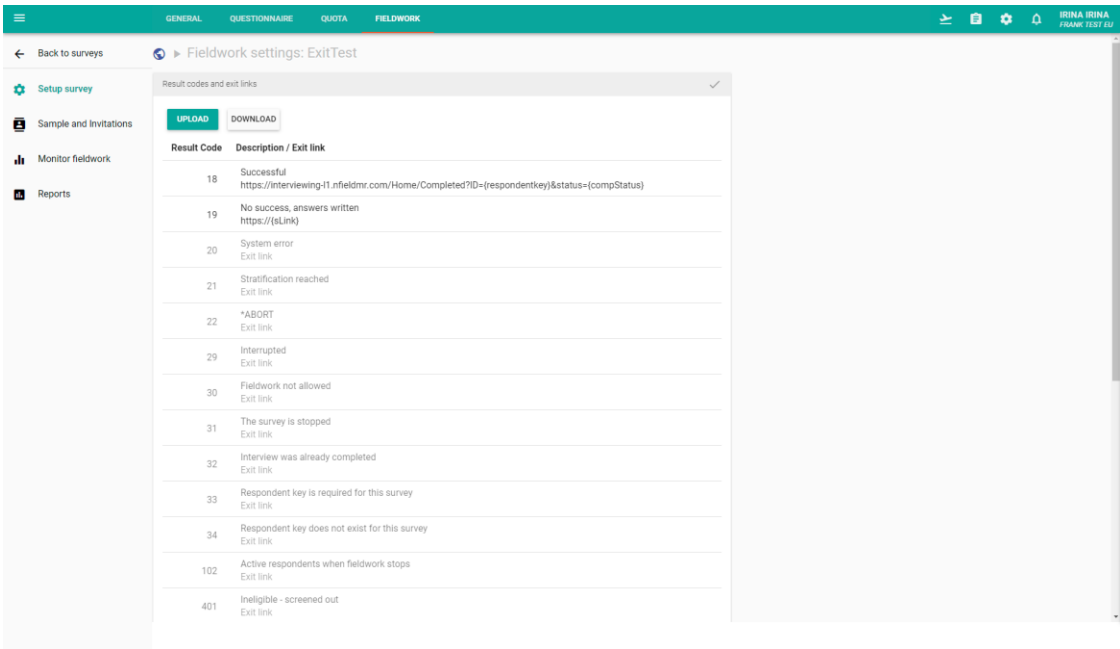
### 10.1.1 Placing the Exit Links

There are two places the exit links can be set up in Nfield Manager:

- 1. In Domain-wide settings, Reports tab which is only visible to Domain Administrators. The exit links set there will be valid for all the surveys in your domain:



2. In Survey settings, Fieldwork tab:



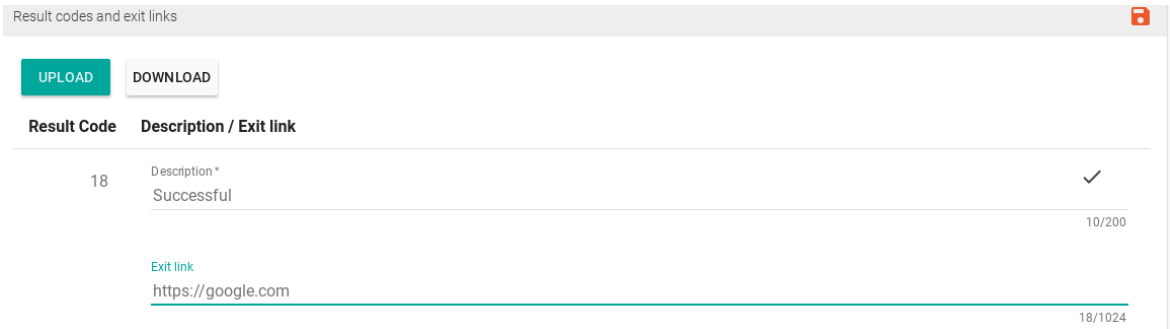
10.1.2 Types of Exit Links

In both places you can define generic links for each exit code (a website to go to). You can also add variables such as age, sex, length of interview etc. to the link.

Adding variables to the exit link is particularly useful when working with panel respondents who get points for completed interviews. You can then pass their points along in the URL.

10.1.3 Generic Exit Links Example

This is an example of a generic exit-link based on code “18”, “Successful”:



## 10.1.4 Variables in Exit Links Examples

### Example 1

Here is an example of using variables in exit links. We add *respondentkey* and a sample variable *compStatus* to the link:

The screenshot shows the 'Fieldwork settings: ExitTest' interface. It features a 'Result codes and exit links' section with 'UPLOAD' and 'DOWNLOAD' buttons. Below these buttons is a table with two columns: 'Result Code' and 'Description / Exit link'. The table contains one entry with Result Code '18' and Description 'Successful'. Below the description, there is an 'Exit link' field containing the URL: <https://interviewing-11.nfieldmr.com/Home/Completed?ID={respondentkey}&status={compStatus}>. The interface also includes a checkmark icon and a trash icon next to the entry.

Variables need to be defined in the script in the *\*SAMPLEDATA*, and then filled in, as in script below (the *respondentkey*, which was used in the original interview link, and is just passed along still needs to be defined in the *\*SAMPLEDATA*, but not filled in):

```
*SAMPLEDATA respondentkey, compStatusplay

*BLOCK

*QUESTION 1 *CODES 61L1
What is your gender?

1:Male
2:Female

*QUESTION 2 *NUMBER 62L2
What is your age?

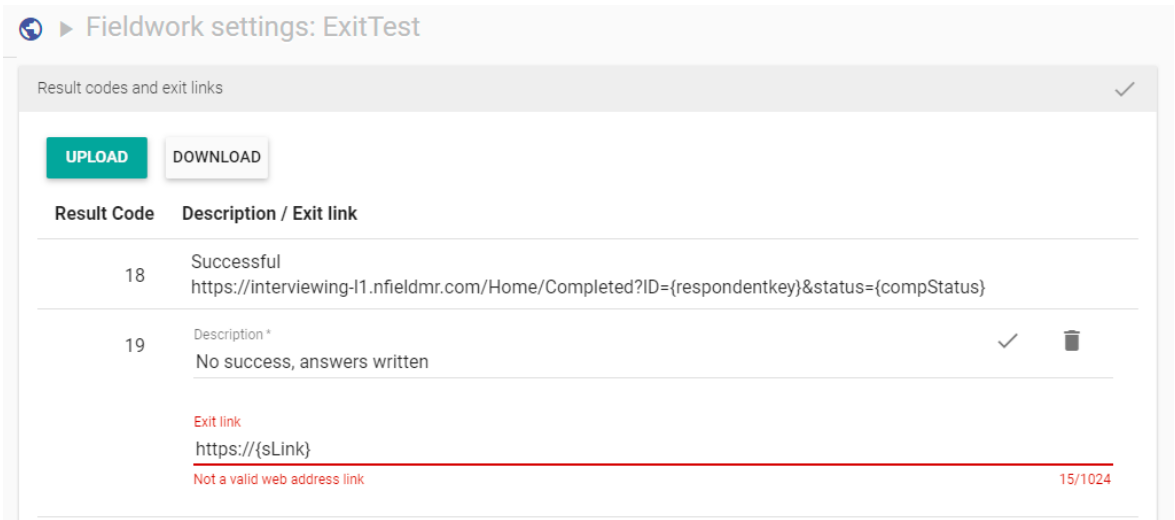
*QUESTION 3 *CODES 64L1 *IF [Q2 > 17]
Do you have a driver license?

1: Yes
2: No

*ENDBLOCK

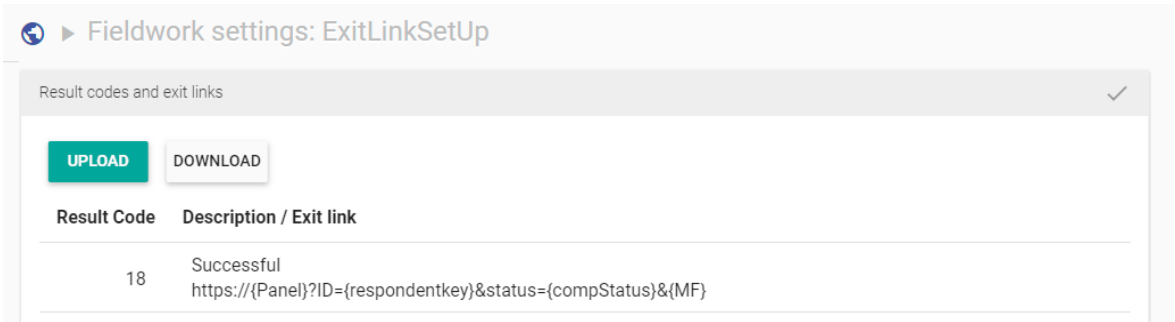
*PUT compstatus "Done"

*END
```



Example 2

Here is another example of variables used in the exit link (*Panel, respondentkey, compStatus, MF*). Note that in this case, while adding this link, you will see a warning that it is “Not a valid web address link”, since it is not a valid web address in this form (it only becomes a valid web address when filled in with correct variable values). You can still save the link.



Here is the script defining and filling in these variables:

```
*SAMPLEDATA respondentkey,compStatus, MF,Panel

*QUESTION 1 *CODES 61L1
Panel?

1:Panel A
2:Panel B
3:Panel C

*QUESTION 2 *Codes 62L1
What is your Gender

1:Male
2:Female
```

```

**Build exit link

*if [Q1,1 & Q2,1] *put Panel "PanelAAAA.com" *put MF "Gender=Male"
*if [Q1,1 & Q2,2] *put Panel "PanelAAAA.com" *put MF "Gender=Female"
*if [Q1,2 & Q2,1] *put Panel "PanelBBBB.com" *put MF "Gender=Male"
*if [Q1,2 & Q2,2] *put Panel "PanelBBBB.com" *put MF "Gender=Female"
*if [Q1,3 & Q2,1] *put Panel "PanelCCCC.com" *put MF "Sex=Male"
*if [Q1,3 & Q2,2] *put Panel "PanelCCCC.com" *put MF "Sex=Female"

*PUT compStatus "Done"

*END

```

In this example we have 3 panels (*Panel A*, *Panel B* and *Panel C*), which we fill in based on respondent's choice in Q1. The *respondentkey* was given to the respondent to run the interview, so that we just pass along. The *compStatus* we also set in the script. The *MF* is to indicate the gender of the respondent. Since in our example *Panel C* uses variable *Sex*, instead of *Gender* (as *Panel A* and *Panel B* do), we use a variable *MF* to store the appropriate variable name and value based on respondent's response to questions 1 and 2.

If we wanted to, we could save the whole link to another variable (for example, *exitLink*), and then just put that variable into the exit link as **Error! Hyperlink reference not valid.**

## 10.2 NIPO Status Page

NIPO has a Status page for Nfield at <https://status.nfieldmr.com>.

The Status page shows the availability for the four main services (Nfield Manager, Nfield public API, Nfield CAPI data synchronization service, Nfield Online interviewing) for each of the four availability regions of Nfield:



# Nfield Status

All Systems Operational

Nfield Americas ?	
Manager ?	Operational
API ?	Operational
CAPL (Sync) Service ?	Operational
Online Interviewing ?	Operational
Nfield Europe ?	
Operational	
Nfield Asia Pacific ?	
Operational	
Nfield China ?	
Operational	

## Past Incidents

Nov 6, 2023

No incidents reported today.

If in your day-to-day business you are experiencing problems with Nfield performance or availability please refer to this page before raising a support ticket with us. You do not need to raise tickets for problems that are already known.

## 11. Only Relevant for CAPI Surveys

### 11.1 Silent Recording

#### Purpose

Recording of the interview in the background (sound recording only, no video; both interviewer and respondent are being recorded).

#### Description

The silent recording can be enabled from the ODIN script with command `*REC` in the script.

#### Syntax

```
*REC [[pos]L<length>] ["Filename"] | [::DELETE::]
[::NOSAVE::]
```

- You specify a data position and optionally a filename.
- `*REC` using a data field specification starts recording; `*REC` without a data field specification ends recording.
- Recording also automatically stops at the end of the script.
- Options `["Filename"]` and `[::NOSAVE::]` should be specified at the start of the recording.
- Option `[::DELETE::]` should be at the end of the recording (when you stop the recording, but only if you have not specified `[::NOSAVE::]` at the start of the recording).

The recorded file will typically be in mpeg4 format, but that may depend on the actual device. The output files are named like this:

```
[interview number]_[file name-][position reference]-[index-number].mpeg3
```

- `interview number`: is the number of the relevant interview
- `file name`: the specified filename, only if specified
- `position reference`: data position specified at the `*REC` command
- `index-number`: starting with 1, but if the silent recording is interrupted by a specific audio capture question, the silent recording will continue in a new file with an increased index-number.
- `::DELETE::` deletes the recording and all other recordings after it in the script.
- `::NOSAVE::` does not save the current recording. By adding the parameter `::NOSAVE::` to a `*REC` command, you can indicate that the selected recording should not be saved. The recording will still start (showing a green dot in the app) but will not be uploaded to Nfield upon synchronisation. Unlike the `::DELETE::` parameter which stops and deletes all silent recordings, the `::NOSAVE::` will have no effect on other `*REC` commands.

#### Example

For interview 00000001, the ODIN syntax:

\*REC 61L1

will result in a file called 00000001\_61L1-1.mpeg3.

If there would be a separate audio capture in that silent recording, the next part would be stored in a file called: 00000001\_61L1-2.mpeg3.

If you also specify a filename, e.g. \*REC 67L1 "MyFile", the resulting filename would be 00000001\_MyFile-67L1-1.mpeg3.

### Notes

- The Nfield Online surveys do not support sound recording, only Nfield CAPI ones do.
- To prevent error messages in online test interviews, it is best practice to make the \*REC command conditional based on the system variable \_ISTEST, like this:

```
*IF [# _ISTEST] *REC 61L1
```

### Best Practices

- Always ask permission before starting silent recording.
- If permission is denied, stop and delete the recording.
- Only record the parts of the interview you need, since it is impossible to listen to recordings that are too long.

### Example 1

In this example, we first ask the user for permission to record parts of the interview. If permission is given, we record the only the most relevant parts of the interview (not the whole interview), otherwise we stop recording and delete whatever was recorded.

```
*TEMPLATE "NfieldChicago"

**start recording introduction
*REC 61L1 "Introduction"

*QUESTION 1 *CODES 62L1
For quality assurance we are recording part(s) of this interview.
Do you give permission to record part(s) of this interview?

1: Yes
2: No

**stop and delete recording if permission has been refused
*IF [Q1,2] *REC "::DELETE::"

*QUESTION 2 *ALPHA 124L15
What is your name?

*QUESTION 3 *CODES 139L1
What is your gender?

1:Male
2:Female

**End recording introduction
*REC

*QUESTION 4 *CODES 140L1
Part of the survey not to be recorded
```

```

1:ok

*QUESTION 5 *CODES 142L6 *MULTI
Which of these brands do you know?

1:Brand A
2:Brand B
3:Brand C
4:Brand D
5:Brand E
6:Brand F

*QUESTION 6 *CODES 148L1 *CONTROL Q5 W
Which of these brands is your favorite?

1:Brand A
2:Brand B
3:Brand C
4:Brand D
5:Brand E
6:Brand F

*QUESTION 7 *CODES 149L1
Another part of the survey not to be recorded

1:ok

*END

```

## Example 2

In this example, we save recording randomly once every 5 interviews. This way we save ourselves the trouble of saving and syncing interviews we couldn't all possibly listen to. This way you just listen to a number of recordings you agreed to in your quality assurance processes.

```

*TEMPLATE "NfieldChicago"

**start recording
*REC 61L1 "LongRecording"

*QUESTION 1 *CODES 62L1
For quality assurance we are recording part(s) of this interview.
Do you give permission to record part(s) of this interview?

1: Yes
2: No

**stop and delete recording if permission has been refused
*IF [Q1,2] *REC 63L61 "":DELETE:."

*QUESTION 2 *ALPHA 124L15
What is your name?

*QUESTION 3 *CODES 139L1
What is your gender?

1:Male
2:Female

*QUESTION 4 *CODES 142L6 *MULTI
Which of these brands do you know?

1:Brand A
2:Brand B
3:Brand C
4:Brand D

```

```

5:Brand E
6:Brand F

*QUESTION 5 *CODES 148L1 *CONTROL Q4 W
Which of these brands is your favorite?

1:Brand A
2:Brand B
3:Brand C
4:Brand D
5:Brand E
6:Brand F

*QUESTION 8 *codes 151L1
This is the end of this survey.
Thank you, do you have an additional remarks or questions?

1:No
2:Yes *open

**Save recording randomly once every 5 interviews.
*REPEAT 5 *FIELD 152L5 *RANDOM
*IF [?R = 1] *REC *END
*REC 1L1 "":DELETE::"
*END
*ENDREP

*END

```

### Example 3

In this example, we record into a file called TooShort. If the recording is longer than 30 seconds, we do not save it. Otherwise we do save it.

```

*TEMPLATE "NfieldChicago"

*VARS STOPWATCH[3]

*PUT STOPWATCH[3] [0]

*REC 61L1 "TooShort"

*QUESTION 1 *CODES 62L1
This is question 1 of the section

1:ok

*QUESTION 2 *CODES 63L1
This is question 2 of the section

1:ok

*QUESTION 3 *CODES 64L1
This is question 3 of the section

1:ok

*QUESTION 4 *CODES 65L1
This is question 4 of the section

1:ok

*IF [STOPWATCH[3] >= 30] *REC 66L1 "":DELETE::"
*IF [STOPWATCH[3] < 30] *REC

*PAGE
The rest of the survey

```

\*END

#### Example 4

In this example, if given permission, we record this interview, except for the question 60. To do that, we put a `::NOSAVE::` on `*REC` for that question. You use `::NOSAVE::` if you want an indication to the interviewer that the recording is being done all through the interview, but not saving every part of the interview so the interviewer does not know which part of the interview exactly will be checked.

\*TEMPLATE "NfieldChicago"

\*\*Start recording consent question  
\*REC 67L1 "Consent"

\*QUESTION 40 \*CODES 68L1

Do you give me permission to record the following parts of this interview for quality control purposes?

1: Yes

2: No

\*\*Stop and delete all recordings if permission is not given

\*IF [Q40.2] \*REC 70L1 "::

\*REC 72L1 "BrandAwareness"

\*QUESTION 50 \*CODES 74L6 \*MULTI

I will read out a number of brands. Please indicate for each of these brands if you have heard of them before.

1: Brand A

2: Brand B

3: Brand C

4: Brand D

5: Brand E

6: None of these \*NMUL

\*REC

\*REC 80L1 "::

\*QUESTION 60 \*CODES 82L6 \*MULTI \*CONTROL Q50 W

Which of those brands have you used yourself?

1: Brand A

2: Brand B

3: Brand C

4: Brand D

5: Brand E

6: None of these \*NMUL

\*REC

\*REC 89L1

\*QUESTION 70 \*OPEN 90L1

Could you describe what you would use those brands for?

\*REC

\*END

11.1.1.1 See Also

\*IF (condition)..... 162

CAPTURE PHOTO..... 329

CAPTURE AUDIO..... 331

PLAY MEDIA ..... 333

## 11.2 GPS Location Fix from Script

Nfield performs a location fix at the start of each CAPI interview by default. Optionally, interviewers can be instructed to validate this initial location fix at the end of the interview.

GPS location fix from script is only possible using the Chicago template.

It is also possible to perform location fixes *during the interview* which can be triggered and then accessed by the ODIN script. This allows Nfield users to ensure CAPI interviewers are at a designated location (and remain there) for specific parts of the interview. It also enables scripters to have Nfield validate this location data at runtime and base routing and/or dynamic content on this information.

It should be noted that performing GPS fixes takes its toll on the device's battery, especially when the location listener has to be switched on and off for each request. To optimize performance, we have chosen to introduce a "continuous tracking" mode, which will keep the location listeners switched on for the duration of the interview. This mode is switched off by default and can be enabled through a new survey setting, which can be set as follows:

In the **Nfield Manager** check the option **"Enable continuous tracking during interview"** in the Interview settings panel under the Fieldwork tab.

Interview settings

Interview display mode

☒ Portrait

☐ Landscape

☐ Device orientation

☒ Capture GPS location during the interview

☒ Notify interviewer to enable GPS location tracking

☐ Ask interviewer to validate location after each interview

☒ Enable continuous tracking during interview

☐ Synchronize paradata and closed answers automatically after interview

☐ Specify instruction text

Enabling continuous tracking mode will store location info in the audit log for each question and page shown during the interview. This will allow users to verify *during quality control* where an interviewer was for each part of the interview.

If the intention is to verify location info at runtime *during the interview*, the GPS fix must be triggered by inserting a dedicated location capture question (question type: \*ALPHA with \*UIOPTIONS "track-gps-location=true") at the relevant point(s) in the ODIN script (i.e. *before* the actual interview question that needs to be associated with the location fix, with both questions placed

inside the same `*BLOCK`). The `*UIOPTIONS` parameter will ensure the location capture question itself will be hidden during the interview, while the captured location info will be stored as its answer in the following json-format:

```
{{"latitude":"<value>","longitude":"<value>","accuracy":<value>,"IsGeolocationEnabled":"<true>"}}
```

### Notes

- Please make sure the allocated field length for the hidden question is at least 90 to accommodate the maximum size of the location info.
- The hidden question *does* require some question text in the script in order to work properly. Depending on the connection quality, the question may under certain conditions be visible briefly before being hidden, so please take this into consideration when choosing this question text.
- To make use of the new GPS location fix from script functionality, the continuous tracking mode **must** be enabled. If Nfield detects that the script contains a location tracking question, while continuous tracking mode is not enabled, it will write only a warning to the answer, instead of the location info.
- If location tracking is disabled on the interviewer's device, Nfield will not throw any warnings, but simply write blank or erroneous location data.

### Example

```
*BLOCK
*QUESTION 20 *ALPHA 64L100 *UIOPTIONS "track-gps-location=true"
Location capture.
*QUESTION 21 *CODES 165L1
Have you ever used this product?
1:Yes
2:No
*ENDBLOCK
```

### For more information

Please watch the following NIPO Academy: <https://nipo.com/webinar/academy-57-gps-location-fix>.

## 11.3 Adding custom fields to Sampling Points

You can attach specific attributes (using custom fields) to sampling points. These attributes can then be accessed from script during a CAPI interview. You can then base question content or routing on sampling point-specific information. This is useful in scenarios in which interviewers are generating addresses themselves during fieldwork.

To make this functionality work, there are two steps:

**Step 1.** Define one or more custom fields in the Sampling Point. This can be done in Nfield Manager by uploading a csv, where any custom fields are assigned the prefix **SampleData::**

Below is an example:

	A	B	C	D	E	F	G	H	I	J
1	SamplingPointId	Name	Description	FieldworkOffice	GroupId	QuotaTarget	Target	Stratum	Kind	SampleData::NearestLocation1
2	SP1	Katwijk aan Zee	excluding Katwijk a/d Rijn	Headquarters	1			Stratum1		Bosplein 1, Katwijk aan Zee
3	SP2	Breda Belcrum	between Terheijdenseweg Breda		2			Stratum2		Groenstraat 25, Prinsenbeek
4	SP3	Amsterdam Rivierenbuurt	between Amstelkanaal, An	Headquarters	3			Stratum3		Hemonylaan 25a, Amsterdam
										Nobelweg 27, Amsterdam

It can also be done through Nfield Public API, using the

**POST /v1/surveys/{surveyId}/samplingPoints** endpoint. In the body, you can specify custom fields under “*customDataItems*” (please see our [API help page](#) for more details). Below is an example:

```

1  {
2    "SamplingPointId": "SP4",
3    "Name": "Oegstgeest Haaswijk-Oost",
4    "Description": "between Haaswijklaan, Oegstgeester Kanaal and Morsebellaan",
5    "Instruction": "",
6    "FieldworkOfficeId": "2ed2e28b-590b-45d7-892e-f789421c925f",
7    "GroupId": "4",
8    "Stratum": "Stratum5",
9    "Kind": 0,
10   "CustomDataItems": [
11     {
12       "Name": "NearestLocation1",
13       "Value": "Boerhaaveplein 3, Oegstgeest"
14     },
15     {
16       "Name": "NearestLocation2",
17       "Value": "Standerdmolen 1, Leiden"
18     }
19   ]
20 }
```

**Please note** that custom field names for Sampling Points are case-insensitive and cannot be the same as system field names (however, there are no changes in the constraints for custom field names in Addresses)

**Step 2.** Declare the custom field(s) as *\*SAMPLEDATA* in your script (without the **SampleData::** prefix):

```
*TEMPLATE "NfieldChicago"

*SAMPLEDATA NearestLocation1, NearestLocation2

*QUESTION 10 *CODES 61L1
Have you ever visited the supermarket located at *?NearestLocation1 ?
1: Yes *GOTO 300
2: No

*QUESTION 20 *CODES 62L1
Have you ever visited the supermarket located at *?NearestLocation2 ?
1: Yes
2: No *GOTO 500
```

### Notes:

1. If values for a custom field are different in the Sampling Point and Address, the value in Address takes precedence, except if the latter is empty.
2. Sampling Point information from a custom field is **read-only** during the interview (for Addresses, it is read-write).
3. Currently, this option is only supported for CAPI surveys with sampling methods Sampling Points with Addresses (with or without quota). In a future update, we will also enable this for Sampling Points without Addresses.

## 12. Appendix

### 12.1 Nfield Acceptable Use Policy

Please review the [Nfield Acceptable Use Policy](#) on the NIPO website.

The Nfield platform creates and renders large and complex questionnaires that are coated with engaging, attractive and mobile-friendly designs, while revising quota targets and sample managements. All users only need internet, a username and a password to work with Nfield. NIPO takes care of all the IT management for all the users. All of this comes with lower costs and better scalability than ever before.

To keep the performance and availability bar so high, all the users of Nfield need to be aware of the rules that:

- maintain and even bolster Nfield's powers for each user.
- support the legal framework of the services provided by Nfield. We encourage you to read "[Nfield Terms and Conditions](#)")

These rules are what this Acceptable Use Policy is about.

### 12.2 Maximum Number of Respondents To Upload

You can upload up to 100,000 respondents to your survey in Nfield. Technically you can upload more respondents, but this will negatively impact the performance of the Nfield Manager. If you need to upload more than 100,000 respondents to a survey, you should split them in separate surveys with no more than 100,000 respondents per survey.

### 12.3 Maximum Length for Sample Fields

The maximum length for sample fields is 1,500 characters. Any character added after this limit will now automatically be removed by the system. The only exception is the field which is indicated to hold the email address for sending invitations, this is limited to 700 characters. Any row that goes over this limit will prevent the batch from being sent.

### 12.4 Response Codes

Response codes are survey exit codes, which are used to track how respondents exit the questionnaire. Did they complete? Got screened-out? Or simply not finished it? It's important to make sure the questionnaire has proper exit commands so that correct exit codes get recorded.

There is a number of standard, system-wide response codes, which cannot be altered.

**Default response codes**

Response code	Response code list	Text shown on the screen	Specification
18	<i>System code</i>	Successful	Saved data
19	<i>System code</i>	*ENDNGB (in questionnaire)	Saved data
20	<i>Online and CAPI: System error</i>		
21	<i>Stratification reached</i>		Saved data
22	<i>*ABORT</i>		Data is not saved
23	<i>-END</i>		
24	<i>Refusals to be retried</i>		
26	<i>Only CAPI: duplicate interview</i>		
27	<i>Used by survey</i>		
29	<i>Online and CAPI: Interrupted</i>		
30	<i>Only Online: fieldwork is not allowed</i>		
31	<i>Only Online: the survey is stopped</i>		
32	<i>Only Online: interview was already completed</i>		
33	<i>Only Online: respondent key is required for the survey</i>		
34	<i>Only Online: respondent key does not exist, but is required for the survey</i>		
35	<i>Only Online: the survey cannot be found</i>		
36	<i>Only Online: An error occurred while processing the request</i>		
101	<i>Only Online: Timed out</i>		Saved data
102	<i>Only Online: Active respondent when fieldwork stops</i>		
104	<i>Only CAPI: stopped by interviewer</i>		Saved data
105	<i>Only CAPI: reset by interviewer</i>		Data is not saved
106	<i>Only Online: Active</i>		
107	<i>Only Online: handed over to external application</i>		Saved data
108	<i>Only Online: stopped; exceeded metric thresholds</i>	Stopped; exceeded metric thresholds	

- **Successful (18):** respondents completed the entire survey, passed the screener and finished the questionnaire all the way to the last question. The command to use in the questionnaire, as previously discussed, is \*END.

- **Screened out (19):** respondents didn't pass our screening. They either didn't fit the research criteria, or they were no longer necessary because we reached our quota targets already. The command to use in the questionnaire is \*ENDNGB.
- **Dropped out (29):** respondents didn't finish the survey, but *are* allowed to restart it. In CAPI, if you see a code 29, it means that there is probably an appointment for another date/time was made. In Online, code 29 means the respondent has clicked the *Pause* button (this button is only present if there is a respondent key and you allowed the interview to be paused in the survey settings on the *Setup Survey/Fieldwork* page). They can restart the survey.
- **Respondent key does not exist, but is required for the survey (34):** respondent key is required for this survey, but only known respondents are allowed for this survey, but the key used by respondent does not match any known respondents (in the sample table for the survey).
- **Dropped out (101):** respondents didn't finish the survey and are *NOT* allowed to restart it (unlike for code 29). For the Online survey, it means that they closed their browser during a survey *without* a respondent key (101). For CAPI, it probably means they've opted out of completing survey (during the interview), and do not wish to make a new appointment.
- **Active respondent (102):** active respondent when fieldwork stops.
- **Active live (106):** this interview is running right now.
- **Stopped, exceeded metric thresholds (108):** the interview has exceeded the metrics values and stopped by Nfield to protect the other interviews in the survey. This should be checked and corrected by the scripter IMMEDIATELY to prevent repeats.

You can also define custom result codes for every survey. These need to be numbered from 201 and upwards. The result codes with numbers 1-200 are reserved as system wide codes. Custom codes can be used for survey specific reasons. For example, you're doing research and you want to only interview people who drive a red motorcycle. You would create 2 custom codes: "Doesn't drive a motorcycle" (201) and another code "Color is not red" (202).

When the fieldwork is done, and the data goes to the data processing department, who will use these exit codes to filter out the "successful" interviews for processing and delivery of results. They will also perform checks on the "not successful" and "aborted" interviews to see where and why they failed.

The results are viewed by fieldwork managers to monitor progress and quality of the fieldwork. Each completed "successful" interview is added to various targets for the survey.

## 12.5 TTStartLink

In section 7.4.2 *Rules for the Sample Table Headers*, there is a list of system variables (that all start with TT). None of them are available for scripting, with the exception of TTStartLink.

TTStartLink holds the URL of the online interview. Using it you can pass the URL to a panel or a third-party so they know which interview to restart.

The TTStartLink needs to be added to the script as \*TEXTVARS variable. You don't need to add any data for that variable – it's a read-only variable. In the interviewing backend we will fill this variable with the proper interview start link.

You can then use that variable in the script as they want. They can use it for the external API calls that you want to do or to build the exit link from within the script. For that purpose, you would also need an ExitLink sample data variable (you can call it as you want) in the script and in the response code exit link.

For example, you can have this relocation URL in the Nfield Manager *https://www.{ExitLink}* and this script:

```
*SAMPLEDATA ExitLink, Age, Gender
*TEXTVARS TTStartLink
*QUESTION 1 *ALPHA 62L10
What is your Gender?
*PUT Gender Q1
*PUT ExitLink 'www.google.com/?startlink=*&TTStartLink&gender=*&Gender'

* PAGE
ExitLink = *&ExitLink
StartLink = *&TTStartLink

*ENDST 107
107
```

The variables will be replaced inside the script and we don't need to wait for the replacement at the end of the interview.

**Note:** The problem with this, is that the data will end up in the database, so if it is Personal Information (PI) data, you would also need to take care of clearing the ExitLink column if it is requested by the respondent.





**NIPO**

Amsteldijk 166  
1079 LH Amsterdam  
The Netherlands  
[helpdesk@nipo.com](mailto:helpdesk@nipo.com)

[www.nipo.com](http://www.nipo.com)