NIPO TRAINING AND CERTIFICATION

SCRIPTING I

nipo

19 February 2019

Table of contents

1. Introduction

Hello,

Welcome to our NIPO Scripter Basic training program. During this course you will learn more about scripting, the ODIN Developer, market-research in general and managing fieldwork.

This training will cover scripting “basic” surveys. This means simple, more generic surveys and questionnaires.

The training is spread out over several chapters. You have to complete each chapter before you can move on to the next one. Once you’ve completed all the chapters, you get an Assignment that allows you to set up your very own “live” project, complete with sample and results.

The final exam serves as a benchmark, so you can test your skills and know-how.

Once you have completed the entire course, you will be considered a Basic Scripter and you will receive a formal certificate.

Who is this training course for?

This training course is for scripters who would like to get certified training. These are people who set up, specify and manage samples and market-research projects. This course is for rookie scripters, people who have done little, to no scripting before.

We will concentrate in this course on ODIN for Nfield Online and CAPI. Although ODIN NFS is in large part the same as Nfield ODIN there are differences.

This course is also for people who use Nfield Composer to create questionnaires, but don’t script themselves. As an Nfield Composer user you should know all of the Nfield software, not just the Nfield Composer. You should also learn a little bit about ODIN.

This course is not for Nfield CAPI Fieldwork Managers, Nfield Online Fieldwork Managers, or more advanced, experienced scripters.

What is the purpose of this training?

The purpose of this course is to train you in how to use the ODIN Developer to set up and manage market research projects and samples.

You can do your training at your own pace, from the comfort of your own office, or home. The course is split up into different chapters; each chapter covers a different element of scripting, using the ODIN Developer, and what it’s used for.

As you complete each chapter, you gain insight in the software. You also learn the most efficient workflow: how to set up a project from start to launching fieldwork, monitoring and checking your progress, quality and results.

This course will take you through the entire process of scripting a research project.

How is this training structured?

The training is organized in a linear, chronological fashion: from start project, to fieldwork, to checking data. By completing the chapters in order this course takes you from the creation of a new project, to set up, to loading sample, to setting targets, to launching fieldwork, to checking results.

Completing this course in the order of the chapters guarantees you will get an understanding of the ideal workflow when setting up and monitoring market research projects, saving you time, effort and cost.

You have to finish each chapter, complete with Assignments, before you can move on to the next chapter.

Once you have completed all chapters, you get a final exercise: you get to set up, run and monitor your very own “live” survey with real sample and real results.

Once we have reviewed your “live” project you graduate and receive a formal certificate for completing this course.

Is this training mandatory?

No, it is not.

You can just grab a manual from our Customer Support Center, acquire a login and learn the software yourself.

However, if you do not complete the training course, you cannot get free technical support from our Global Support Teams in Asia, America or Europe.

Certified customers who have completed this course get free technical support.

We urge you to take the time and complete the entire course. It will help your skillset and make you a real champion of the software. It’ll help you set-up and manage projects with confidence and ease.

It’s a fun and interactive way to get to know the software and to learn more about online market research.

How should I proceed?

We advise that you open two browser windows, or two tabs. You should also open the folder where you have downloaded and saved the complimentary training materials.

In the first tab or window you can open this training course and work through each chapter. In the second window open the ODIN Developer.

You can then go back and forth and explore the examples from each chapter and practice these inside the ODIN Developer as you move forward.

The course is pretty evenly split between theory and experimentation; each chapter ends with Assignments, so you can immediately put what you’ve learned into practice and test your understanding and skills.

We strongly advise you to complete this training course before running any live projects.

We furthermore advise you to use the ODIN Developer scripting guide. All ODIN commands in this training course will be printed in Courier Font, the ODIN scripting guide lists all commands alphabetically, with many more examples and features that we can do during this course. You get a copy of the reference guide when you install the ODIN Developer, see further below this course.

If you feel anything is missing from this training course, or if anything should be made clearer, please let us know. We welcome your feedback. Send us a mail at helpdesk@nipo.com

Duration

We estimate that this course will take you some 16 hours to complete.

Most chapters can be studied and completed rather swiftly. The chapters on quotas, sampling and invitations take a bit longer.

We estimate that should be able to work through a short chapter in 1-2 hours. The quota chapter will probably take 4-8 hours.

You can spread these hours out over several workdays, even do the course from the comfort of your home.

Welcome to the wonderful world of scripting!

2. Market Research

The research process is a linear one. A client will have a research question. That research question will be turned into a questionnaire with specific questions meant for a specific target audience: your respondents. A scripter or programmer will turn that questionnaire into an electronic script. This “program”, the digital questionnaire, is then processed by Nfield and delivered to people, either respondents, or interviewers who conduct the actual interviewing either by telephone, via an app or online. This is called fieldwork: delivering your questionnaire to people and inviting them to fill it in. When people begin to fill in your questionnaire you are conducting data collection. The collected data, the results, are processed and analyzed and eventually delivered to the client, either as a report, data or presentation. The results are your end product.

3. ODIN Developer

The ODIN Developer is offline software that allows you to “script” questionnaires. Scripting questionnaires is often called application programming. You use the ODIN Developer, the application, to create your questionnaire, your script. The ODIN Developer runs offline. It will require a PC/laptop, with Windows, so you can install ODIN.

Using the ODIN Developer one can create questionnaires for face2face interviewing, self-completion (online), or telephone interviewing. It is imperative that you consider your audience as you script questionnaires. Who is actually going to be working with this questionnaire? Who is going to fill it in? Is it an interviewer, or a respondent? And if it's an interviewer, how will the interview be conducted: via telephone, via an app on some device, or online?

As we mention specific ODIN commands during this training course, we strongly advise you to look up these commands in the ODIN reference guide. Each command is listed alphabetically in the scripting guide with many examples as well as suggestions for related commands.

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

- Create, edit and check questionnaires for NFS CATI, CAPI, CAWI and Nfield CAPI, Online
- Run a preview of the questionnaire in NFS CATI, CAPI, CAWI and Nfield Online
- Create and edit picture libraries for use with NIPO ODIN questionnaires
- 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.

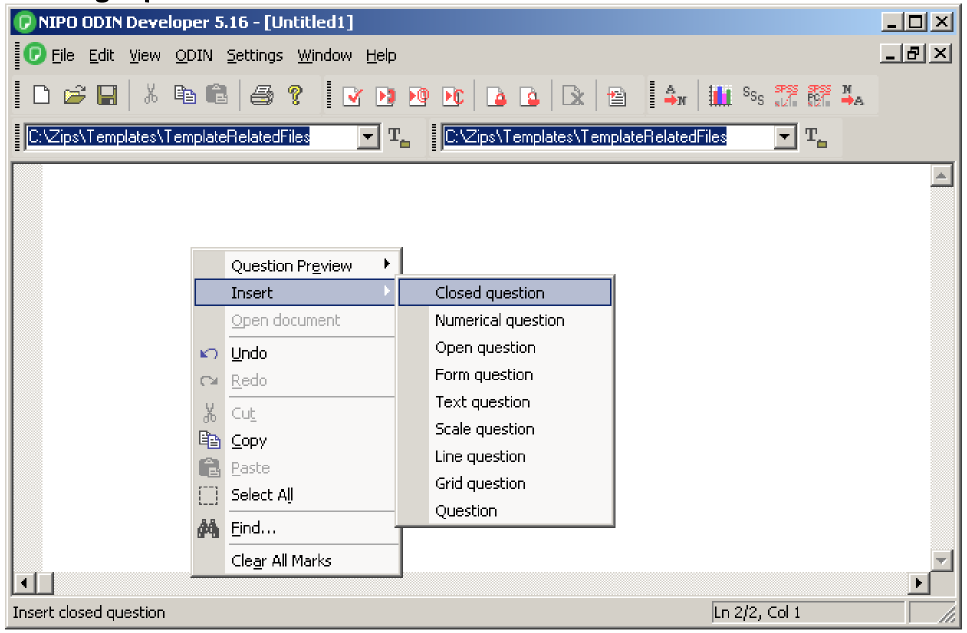
Please note: We urge you to study the entire ODIN Scripting manual. This training is not meant as a replacement.

There are two ways to script using the ODIN Developer. The first method is by using the menus to select questionnaire element and add them.

Inserting Question Definitions

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

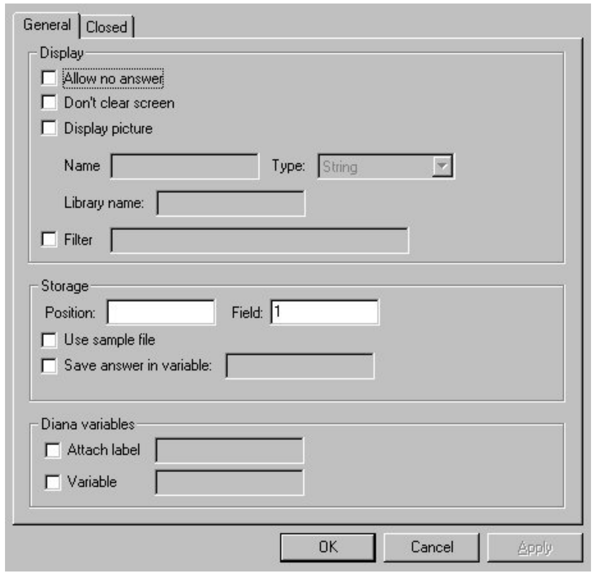
Inserting a question



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


General | Closed

Display
- ☐ Allow no answer
- ☐ Don't clear screen
- ☐ Display picture

Name: Type: String

Library name:

☐ Filter

Storage

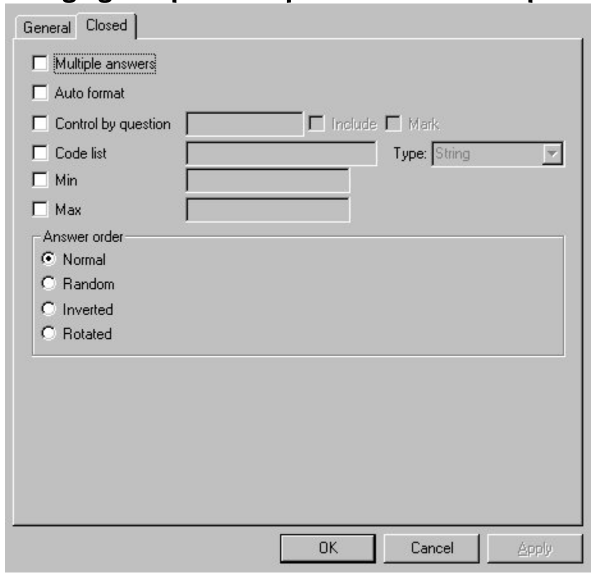
Position: Field: 1

- ☐ Use sample file
- ☐ Save answer in variable:

Diana variables
- ☐ Attach label
- ☐ Variable

OK | Cancel | Apply

Changing the question options for a *CODES question

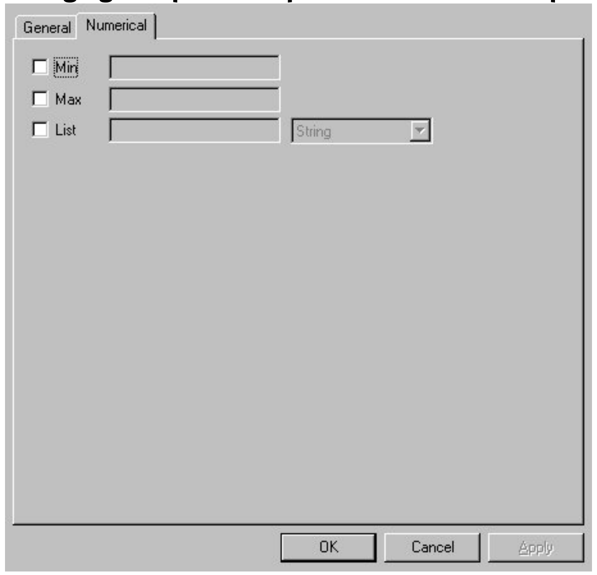

General | Closed

- ☐ Multiple answers
- ☐ Auto format
- ☐ Control by question ☐ Include ☐ Mark
- ☐ Code list Type: String
- ☐ Min
- ☐ Max

Answer order
- (●) Normal
- ( ) Random
- ( ) Inverted
- ( ) Rotated

OK | Cancel | Apply

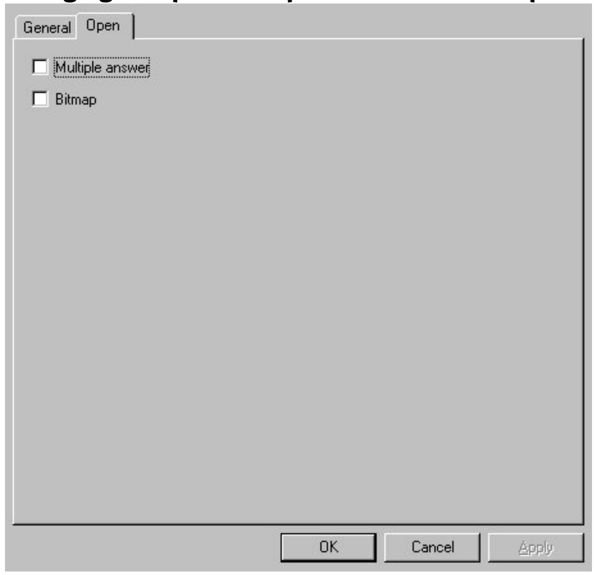
Changing the question options for a *NUMBER question


General | Numerical

- ☐ Min
- ☐ Max
- ☐ List String

OK | Cancel | Apply

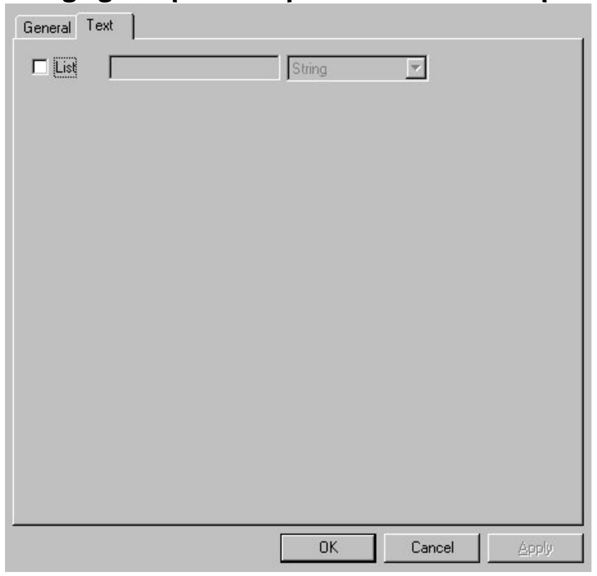
Changing the question options for an *OPEN question


General | Open

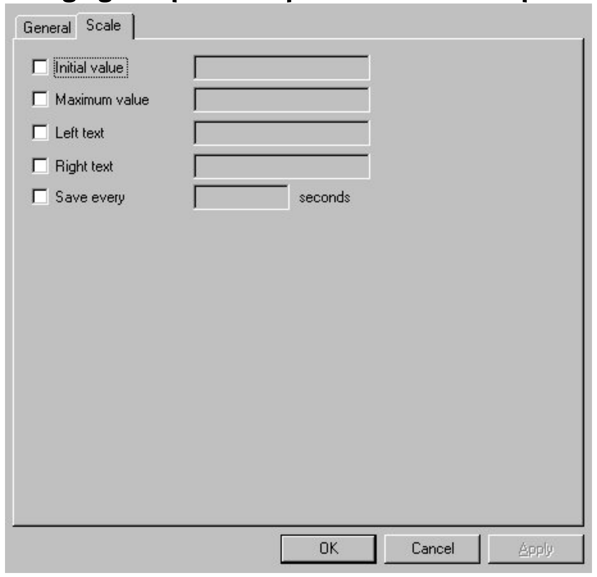
- ☐ Multiple answer
- ☐ Bitmap

OK | Cancel | Apply

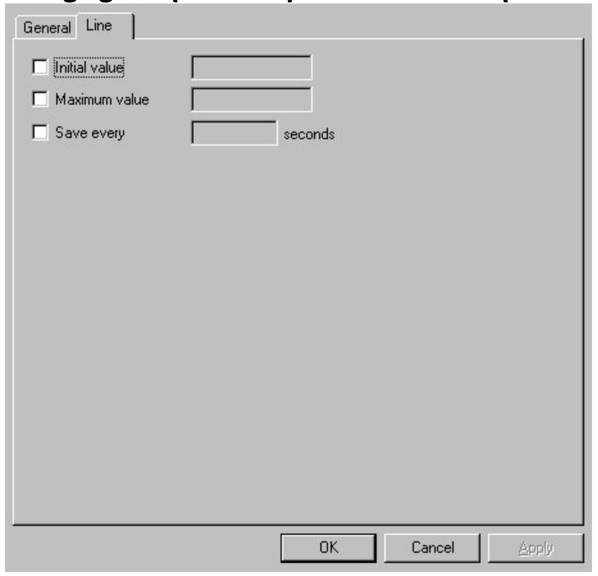
Changing the question options for an *ALPHA question



Changing the question options for a *SCALE question



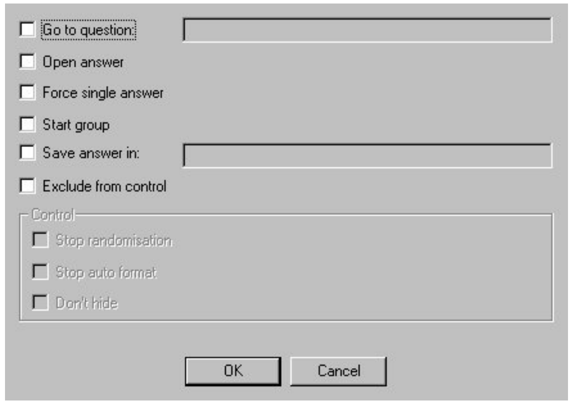
Changing the question options for a *LINE question



Changing Code Options

Right-click on a category code line to change the code options. The following dialog appears.

Changing the code options in a *CODES question



☐ Go to question:
☐ Open answer
☐ Force single answer
☐ Start group
☐ Save answer in:
☐ Exclude from control

Control
☐ Stop randomisation
☐ Stop auto format
☐ Don't hide

OK Cancel

So you can create an entire, basic questionnaire using the composer inside Nfield, or you can create a questionnaire by selecting and inserting the right elements using the ODIN Developer.

The second way to script is to actually type ODIN syntax, to actually learn the ODIN “language” so you can create questionnaires. In this training, we will focus on the actual commands, the actual ODIN language. If you can master that language, you will become an efficient scripter.

4. Let’s begin with the basics

Let’s begin with some basic definitions we will be using throughout this course.

4.1 Definitions

Survey - your total project in its entirety. A survey will consist of: a questionnaire, media files such as sound or video, a list of respondents you want to question and your collected data.

Questionnaire - your questionnaire will consist of a script (a file with the actual questions and answer options) and your quota variables and quota cells.

Script - the actual programmed questionnaire, containing the questions, the answer options and the routing

Question - an element that collects data. Questions can either be asked directly of respondents (where they input an answer) of questions can be filled with information in the background.

Answer - The answers are what we put into the question. There are various kinds of answers: open, (alphanumeric) or closed. Open answers are answers where the respondents type in an alphanumeric response, for instance their opinion about something. Alphanumeric answers are for questions such as age, or zipcode. The response - the answer - is usually a short alphanumeric string. Closed answers are answers that need to be chosen from a predefined list.

Quota - A quota is a subset of people you wish to investigate. A quota can be anything, such as Gender, or Age, or a zipcode, or a city, or an income-class. A quota consists of a quota variable - the actual name of the quota - such as "Gender" and quota-cells, the actual elements that make up the variable such as "Male" and "Female".

Targets - the numbers of people you wish to investigate overall and per quota-cell.

Sample - a detailed list of the people you wish to query/investigate

Invitations - the actual invites you send out to the people you want to participate in your online surveys.

Respondents - the people who will be answering your questions, the people you are investigating

Data - the information you are collecting

We will begin with the basics, allowing you to get an understanding of the various online research concepts as well as the actual interface, the Nfield Manager. Once you're more comfortable with those, we will begin to set up actual projects.

4.2 Main Survey Components

Let's take a closer look into the main components of Market Research. Your main components will be: your sample (the people you want to participate) and your questionnaire (the actual questions you want them to answer). The end result is the collected data itself.

Sample (Who to ask)

Approaching respondents is expensive. You want to limit the amount of people you approach. If you have to send out thousands and thousands of invites to reach the right people, fieldwork will become very costly. Making sure your sample is just right is the first important step. Sample will usually be delivered by the client, by a sample provider, or by your own sampling department. We advise you to always check a sample file. Use your own eyes to check and make sure that sample is alright. We have suggestions further down this course on exactly what to check and watch for.

Questionnaire (What to ask)

The questionnaire is the actual data collection tool. It will typically consist of four sections.

1. **Sample Variables / Background Questions** - Your sample, a file containing the actual people you want to invite, will often contain personal information about your potential respondents, such as email, age, gender, name etc. These variables are "read" into your questionnaire, so that they become part of the data you collect. The respondent specific information from the sample is read into your questionnaire in the background.
2. **Screener** - You want to make sure you query the right people. A screener section is meant to identify and classify the right people and to weed out the wrong people. If you are conducting research on car drivers? One of your screener questions will be: Do you have a driver's license? The people who say "no", cannot continue. The screener section will usually take place at the beginning of the questionnaire. Conducting fieldwork is

expensive. You want to disregard and dismiss people who do not meet your research criteria as soon as possible.

3. **Main Questionnaire** - This will be the actual research, containing all the questions you want to ask to your respondents.
4. **Exit** - Respondents will exit your survey in a variety of ways. People who do not meet your research criteria will exit in a different way as the people who complete your survey. The “outro” or exit-section of your survey is where respondents are: a.) thanked for their time and participation and b.) classified with an exit-status. This can be ‘successful’, if the respondent completes the interview in its entirety. Or it can be “screened-out” if the respondent doesn’t meet your research criteria. There are numerous ways to exit from a survey and you can customize those exits.

Look and Feel

The way your survey looks, feels and behaves, it’s colors, logos and buttons, is controlled outside of the actual questionnaire. The look and feel of your survey is determined by cascading style sheets and javascript. We call this a template and a theme. A template is a “skin” that sits around your survey. A template interacts with the browser to render the survey in a specific way. A template can have themes. Themes contain custom elements that overrule the master template, altering standard behavior for specific projects, clients or even questions.

1. **Template** - Because respondents are filling in the questionnaires themselves, you have to make absolutely sure your survey looks and performs the same on every device, be it a laptop, a PC, a phone or a tablet. A template is a “skin” that sits around your digital questionnaire that determines the look and feel of your survey. It determines the colors of buttons, logo’s and the “style” of the overall questionnaire. Nfield comes equipped with a number of pre-designed templates. Templates can only be uploaded and activated by NIPO. They determine the way things look inside the browser or the CAPI app. We call that “client side”, or the “front end”. They can also interact with the “back-end”, the actual Nfield servers, which is why only NIPO Software can upload, activate and manage templates.
2. **Theme** - By using a custom Theme you can overwrite certain elements and controls in the template. You can overwrite and change fonts, colors, logos, even the way specific questions look and behave. Themes are written using JavaScript. If you are considering creating your own custom themes you should get in touch with our helpdesk. They have extensive documentation and example themes available. Themes cannot do anything server side. They only regulate things client-side, at the respondent’s end, in their browser. Themes are therefore created and managed by you.

Data

The questionnaire is filled in by typing answers to all the questions, those answers are stored as data. There are various kinds of data. There is open ended data, when respondents or interviewers have to state an opinion and actually type words or sentences as their answer. And there is closed data, when respondents or interviewers merely select an item from a predefined list which is stored as an (alpha)numeric code. Each respondent becomes a “record” in the data. All the records, all the respondents combined are called your datafile.

4.3 Typical Workflow

The typical workload of a scripter is to set up these components so that data can be collected, while monitoring the overall quality of the sample, fieldwork and data.

You will want to know what your sample consists of and how many people you need to interview.

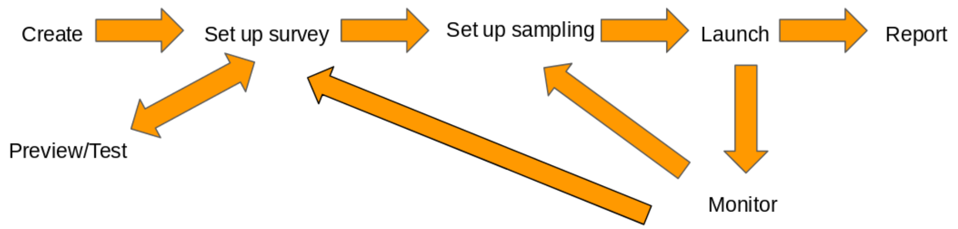
You will require a questionnaire that you will need to thoroughly test.

After testing you will deliver your script with all its settings so that data collection and fieldwork can begin.

Once that is done we will share a link to the actual survey with respondents. Sharing this link can be done in a variety of ways: via an sms, embedding the link on some website, or sending a formal invitation with the link to specific respondents. Alternatively, if interviewers are conducting the actual interviews, you will either provide them with sample (telephone numbers to dial) or a walking route or location, where to find people to interview.

Once fieldwork begins and data is being collected, we will monitor fieldwork, check progress and monitor the quality of the results.

The diagram below depicts the actual workflow. *Please have a close look.*



The standard workflow consists of the following steps:

Always

- Create a new script or copy an old one
- Test/Preview

Note: This is a circular process. As you create and style your questionnaire and add media files to become an actual script, you thoroughly test, going back and forth between creating and editing or (re)uploading and preview, to make sure all is functioning properly.

If needed

- Upload a custom theme (talk to our helpdesk)
- Set up quotas/targets
- Upload respondent data/specific respondents to interview (sample)
- Set up Invitations

Always

- Check all your settings
- Test/Preview your survey
- Share the survey link (or if needed: send invitations)
- Monitor fieldwork (progress, targets, quality control)
- Run toplines to check results

- Download data for processing and reporting

Note: The 2nd part of this process is also circular. After you've prepared your sample and begun inviting people, you check and monitor fieldwork and make adjustments as well. You can upload more sample, send more invites etc. You furthermore use the reporting features to check your actual data and make adjustments if anything is wrong and/or missing. These adjustments can mean changes to the questionnaire, its settings, or the sample.

The first chapters will deal with some basic understanding of the ODIN Developer and what the different elements and settings mean and can be used for. In following chapters we begin to set up actual surveys.

This training will explain the above workflow, allowing you to create market research questionnaires using the ODIN Developer.

5. Installing the ODIN Developer

Installing the ODIN Developer is very straightforward. Simply download the ODIN Developer from the customer section of our website. Just contact our helpdesk and ask them for a login to our Customer Support Center. Please note: You will need to have ADMIN permissions, which allow you to install software on your PC or laptop. If you do not have such permissions, speak to your local IT Manager.

The ODIN Developer comes with various manuals, example scripts and other tools. This training course is **not meant as a replacement of those**. As you begin scripting you will use these manuals either in print or digital form regularly, to look things up, or find examples of code.

The ODIN Developer allows you to either select and insert questionnaire objects, or type the syntax, the actual code yourself. In this training we will be dealing with basic syntax, teaching you how to script.

6. Components

Let's start with the main components that make up a question. A question will always have an ID, this is a number, a unique identifier. It's always best to begin with Question 1 and then Question 2 and then so on and so forth. A question will also have a type. The type determines what kind of question it is, open or closed for instance. Each question will have a starting data position and a specified length. This is the actual physical space the question will take up in our data file. And finally, each question will have options that determine the behavior inside a specific question type, for instance rendering an answer list in a randomized order. The Question Text is how the question is actually phrased.

The syntax will always look like this, with each argument preceded by an asterix.

***Question ID *Questiontype Lnumber *Questionoptions Question Text**

7. L (Length) In Questions

As stated before, each interview, each respondent becomes a record in our actual datafile. A record is a single line in a datafile. That line will consist of positions. We begin on the left-hand side, with position 1 and we end wherever we end up. The ODIN Developer allows the user to allocate data positions for your questions, making sure answers are logically and comprehensively stored. To do that, the user needs to estimate the number of positions needed to store the answer to the question and add it to the L argument. For example, in the script in section 8, *Alpha (below) L15 means 15 places are available to store the answer.

8. Question Types

A questionnaire is a data collection tool, nothing more and nothing less. The purpose of all questionnaires is to collect data. Each question is therefore a 'net' that captures the actual answer(s), the data. The ODIN Developer allows you to create different types of questions. The main question types are listed below.

*ALPHA

These are **open-ended** questions. Meaning there are **no answer options.** These questions are used when we want an interviewer or respondent to type in a name, or a zip-code, something short.

For instance:

```
*QUESTION 1 *ALPHA L15
What is your first name?
```

Our data, the actual answer(s) is open ended as well. These open-ended answers are coded to regular codes, by a coding department, after fieldwork has completed. Please do not confuse these two question types. *ALPHA is used when we want some short input from the respondent, such as a name, or a brand. We use *OPEN when we want the respondent or interviewer to really be able to state an opinion, a lengthier response.

Please note: Open ended questions do not end up in your datafile. They end up in what we call an O-file, a file containing just the responses, the answers to open-ended questions. The O-file is usually coded by a coding department after fieldwork. (We will deal with this in later chapters).

All other questions types are closed questions. This means they come with answer options or a limit to what can be submitted as an answer. Closed questions end up in your datafile, where all the responses are stored as numbers.

Assignment 1

Create a script with a question asking for someone's name

*NUMBER

These are numeric questions, where we expect the interviewer, or respondent to type in a number.

For instance:

```
*QUESTION 1 *NUMBER L3 *MIN 16 *MAX 99
What is your age?
```

In this case our data will be a number between 16 and 99, the actual age of the person we are interviewing.

Assignment 2

Building on the previous created script add a question asking for somebody's exact age

*CODES

The most common questions will be categorized questions. These are questions that come with a predefined list of answer options. There are two types of categorized, coded questions, namely: single response questions and multi response questions.

Single

```
*QUESTION 1 *CODES 61L1
Do you own a car?

1: yes
2: no
```

Multi

```
*QUESTION 1 *CODES 62L20 *MULTI
Which car do you own?

1: Audi
2: BMW
3: Citroen
4: Fiat
5: Ford
6: Hyundai
7: Other *OPEN
8: None of these *NMUL
```

L (Length) In Questions

As stated before, each interview, each respondent becomes a record in our actual datafile. A record is a single line in a datafile. That line will consist of positions. We begin on the left-hand side, with position 1 and we end wherever we end up. The ODIN Developer allows you to allocate data positions for your questions, making sure answers are logically and comprehensively stored.
Note the first 60 positions of each data record are reserved for system variables.

Assignment 3

Building on the previous created script add a question asking for somebody's gender

The numbers, 61 and 62, mean the actual positions in our datafile, the 61st and 62nd positions, counting from left to right.

The "L" stands for LENGTH.

Our single question will only take up one position in our datafile, containing either a '1' or a '2'.

Our multi question can take up a maximum of 8 positions, since we have 8 answer options. We therefore use 62L8, which means *begin at position 62 and allocate 8 positions from there onwards*. A multi question is 'punched' (coded) into the data as a string of zeros ('0') and ones ('1'). The one ('1') means **mentioned** and the zero ('0') means **not mentioned**.

So
`10000000`
Means they own an Audi

And
`00101000`
Means they own a Citroen AND a Ford.

As you can see we added two "special" options, namely "Other" where we added `*OPEN`. This means that whenever this option is chosen, the interviewer or the respondent will be prompted to type in the actual car brand.

The second "special" option `*NMUL` means "NOT MULTI". If our respondent doesn't own a car, they need to be able to say, "None of these". Options like "Don't know", "Refuse to answer" and "None of these" are **always** the only response possible.

You cannot own a brand of car and at the same time say, "None of these". We therefore **always** use *NMUL on these options to maintain our data consistency and data integrity.

Please note: You do not have to add positions and length to your questions. When you are done scripting your questionnaire, you compile your questionnaire. The ODIN Developer will then fix positions and length automatically, making sure all your codes fit based on the L(engths) you have provided in your ODIN script.

Let's recap: So there are various **question types**, some of which come with predefined codes. Then there are various **question options**, such as conditions and randomization. And then there are **codes options**, such as `*NMUL`. And finally, there are **commands**, instructions which allow for flow-control and questionnaire logic.

Assignment 4

Building on the previous created script add a question asking for all the car brands somebody might know

*OPEN

These are **open ended questions**. This means that there is no predefined answer list. An *OPEN question is used for a different purpose than an *ALPHA question. An *OPEN question is used when we want a respondent or interviewer to type in their opinion about something. For instance, "What do you think of our Prime Minister"?

```
*Question 1 *OPEN 61L100
What do you think of your Prime Minister?
```

Please note: *OPEN questions do not end up in your data file (U-file), but in your open-ended answer file (O-file). For more on these files, see further down this training.

Assignment 5

Building on the previous created script add an unaided awareness question and an aided awareness question.

9. Controlling Answers

For respondents and interviewers alike, filling in questionnaires can be quite tedious and even boring. It is therefore important to make sure that we control what appears on somebody's screen and how. With simple single punched or open-ended questions, we don't have to do much. When dealing with longer answer lists, or answer options which require us to include or exclude previous answers, we need to make sure we control how they are displayed.

*CONTROL

```
*QUESTION 1 *CODES 61L1
Please state the first car brand that comes to your mind.

1: Audi
2: BMW
3: Citroen
4: Fiat
5: Ford
6: Hyundai
7: Other *OPEN
8: None of these *NMUL

*QUESTION 2 *CODES 62L8 *MULTI *CONTROL Q1 N
I am going to read out a number of car brands.
Which of these have you heard of?

1: Audi
2: BMW
3: Citroen
4: Fiat
5: Ford
6: Hyundai
8: None of these *NMUL *NOCON
```

Let's begin with the `*CONTROL` command. This command allows you to INCLUDE or EXCLUDE answer options from previous questions. The "N" means EXCLUDE. And the "W" means INCLUDE. So `*CONTROL Q1 N`, means show all the answers, except the answer already given at Q1. Using this construction means that the respondent cannot duplicate their answer from Q1 to Q2. The answer they have already given at Q1 is excluded from the answer options in Q2.

The `*NOCON` command makes sure that this answer option - “None of the above” doesn’t fall under the `*CONTROL` command and is therefore always displayed.

Now let’s try that the other way around.

```
*QUESTION 1 *CODES 61L8 *MULTI
Which of the following car brands have you heard of?

1: Audi
2: BMW
3: Citroen
4: Fiat
5: Ford
6: Hyundai
7: Other *OPEN
8: None of these *NMUL

*QUESTION 2 *CODES 71L1 *CONTROL Q1 W
And which of these is your favorite? (Please pick one).

1: Audi
2: BMW
3: Citroen
4: Fiat
5: Ford
6: Hyundai
8: None of these *NMUL *NOCON
```

In this example we will ask Q2 only if the respondent mentions brands 1-6. We will include the answers given in Q1, so that the respondent can only pick a favorite, out of their total brands.

Using *CONTROL allows to safeguard your data. We don’t want a respondent who knows brands A and B to suddenly drive brand C, since *he would have to know that brand as well*. We don’t want people buying things or driving things they are unaware of.

*RANDOM

When people have to pick answers from long lists, they tend to pick answers that are at the top of the list. To offset this, you have to *randomize* the answers of multi response questions. If the answer options are always randomized, you offset any bias.

```
*QUESTION 1 *CODES 61L29 *MULTI *RANDOM
Which of the following car brands have you heard of?

1: Audi
2: BMW
3: Citroen
4: Fiat
5: Ford
6: Hyundai
7: Mercedez
8: Ferrari
```

```
9: Porsche
10: Volkswagen
11: Opel
12: Seat
13: Peugeot
14: Renault
15: Rolls Royce
21: Other *OPEN *NOCON
29: None of these *NMUL *NOCON
```

The `*RANDOM` command will randomize the order in which the list is displayed. This is useful for respondents as well as interviewers and makes for higher quality data collection. Please note: We do not want our “Other” or “None of These” options to be randomized, so we use `*NOCON`, to make sure they are not randomized.

*BUT

You can also create additional buttons, adding answer options that do not appear in your answer list. In the example below the button with the text “Don’t know” will appear, and if the user selects it, code 5 will be saved in the data. More information on this command can be found in NIPO ODIN 5.18 Scripter's Guide.pdf.

```
*Question 1 *CODES 61L1 *BUT 5 “Don’t know”
Which is your favorite Star Wars character?

1: Luke Skywalker
2: Han Solo
3: Darth Vader
4: Yoda
```

10. Flow Control

As stated before, a questionnaire is always linear, both for the scripter, as well as for the people who will later fill in the questionnaire. It should therefore be as easy as possible to get from the beginning to the end of the questionnaire. Not all questions will be asked of all people. Some questions will only be asked if people meet certain conditions. Making sure people only get the questions they are supposed to get, is called flow control. You, the scripter, will control the flow of the respondents throughout the questionnaire. It is also called “routing”; you dictate the route that respondents take through the questionnaire.

*GOTO

We use the `*GOTO` command when we want somebody to move forward to another question in the questionnaire. If we want them to miss a section for instance.

```
*QUESTION 1 *CODES 61L1
Do you own a car?

1: Yes
2: No *GOTO 999

*QUESTION 2 *ALPHA 62L20
```

```
What brand of car do you own?

*QUESTION 999
```

Please note: You can move **forward** and **backwards** using the `*GOTO` command. Make sure you direct people to questions they've actually answered, or it will throw a warning.

Please note: Question 999 is a blank question. It doesn't have any text or any answers. `Q999` is nothing more than a label to direct people to, like a specific address or location, inside your questionnaire.

Please note: We are directing people to '999' and NOT 'Q999'

*BACK

There are situations thinkable where it makes sense to skip somebody back into the questionnaire. You do this with the `*BACK` command. Let's say we're looking for the person who does the main daily shopping for the household. Our questionnaire would look like this.

```
*VARS NumCars

*QUESTION 1 *CODES 61L29 *MULTI *RANDOM
Which of the following car brands have you heard of?
Mention at least 3 brands

1: Audi
2: BMW
3: Citroen
4: Fiat
5: Ford
6: Hyundai
7: Mercedez
8: Ferrari
9: Porsche
10: Volkswagen
11: Opel
12: Seat
13: Peugeot
14: Renault
15: Rolls Royce
21: Other *OPEN *NOCON
29: None of these *NMUL *NOCON

*COUNT NumCars Q1

*IF [ NumCars < 4 ] *PAGE Less than 4 cars *BACK 1
```

Please note: When a respondent doesn't meet our criteria (they've selected less than 4 cars), we show them the text "Less than 4 cars", send them back to the previous question and re-open with our introduction.

Please note: We are directing people to '**1**' and not '**Q1**'

*IF

The most commonly used command for flow control is `*IF`. It works very logically. If the respondent is younger than 18 you do not ask questions about beer. You can place `*IF` on questions, to make sure they are only show to the right people. Our previous example about car ownership, will then look like this.

```
*QUESTION 1 *CODES 61L1
Do you own a car?

1: Yes
2: No

*QUESTION 2 *ALPHA 62L20 *IF [ Q1 , 1 ]
What brand of car do you own?
```

In this example, we are asking Q2, only if people answer '1' at Q1.

`*IF` is much more powerful and flexible than the other commands, since you can make combinations of codes, like this.

```
*QUESTION 1 *CODES 62L6 *MULTI
Which of the following movies have you seen in the last 6
months?

1: Star Wars: The Last Jedi
2: The Guardians Of The Galaxy vol. 2
3: Arrival
4: Charlie Chaplin: Gold Rush
5: Orson Welles: Citizen Kane
6: None of the above *NMUL

*QUESTION 2 *CODES 92L1 *IF [ Q1 , 4-5 ]
So you like old, classical movies?

1: Yes
2: No
```

In this example we are using a combination of codes for our `*IF` clause, namely codes 4 and 5, both old, classical movies. If either of those answers is given? We ask `Q2`: So you like old, classical movies?

You can use any combination of codes. If the codes are sequential, you can separate them using the hyphen ("-"). If they are not sequential, you should use a comma (",") to separate the codes, like so:

```
*QUESTION 3 *CODES 92L1 *IF [ Q1 , 1,2,3 ]
So you're a science fiction fan?

1: Yes
2: No

*QUESTION 3 *CODES 92L1 *IF [ Q1 , 1-3 ]
So you're a science fiction fan?

1: Yes
2: No
```

Both `*IF` statements mean the same thing. If somebody answers *any* of the codes 1, 2 or 3 in `Q1`, we will ask `Q3`: whether they are a science fiction fan?

11. Intro

Each questionnaire begins with an introduction. This introduction is either read by the respondents themselves or read aloud by the interviewers as they conduct the actual interviews. The introduction will explain:

- Who “we” are. What company we’re from.
- What we are researching. What topic we wish to investigate.
- Who we’re looking for. Which people we want to interview.

It’s important to keep your introduction simple, concise and to the point.

There are two main ways to create an introduction using ODIN. You can create an “empty” question; this is a question without any answer-options. Or you can create a nice-looking page with the introduction.

An empty question will look like this:

```
*QUESTION 1
Hello, my name is <x>. I am with <name agency>.

We are conducting a survey about cars.
We are therefore looking for people who drive a car, to answer
some questions.

We are doing this research for <name car brand>.

The entire survey will take a maximum of 10 minutes.
```

The above text will appear on the screen with a <NEXT> button when the research begins. Once the text has been read, the <NEXT> button continues the actual questionnaire.

We urge you to look up all ODIN commands used in this training in the ODIN reference guide, where you will find more features, options and many more examples on how to use these commands.

Always make sure your questionnaire begins with a clear and comprehensive introduction so that interviewers and/or respondents immediately know what your research is about.

12. Ending A Questionnaire

There are various reasons to end a questionnaire, so there are multiple commands to do so.

A respondent can for instance not fit our criteria. We may be conducting a study about cars, for which we only need car drivers. Anybody under 18 would therefore not qualify, neither would anybody older who doesn’t have a driver’s license. We would therefore end the questionnaire if the respondent is not useful for us.

If a respondent completes our entire questionnaire, we will want to thank them for their participation and end the questionnaire there.

*ENDNGB

Sometimes we will only deal with the screening section of our script at a later stage in the questionnaire. We will ask people their age, gender, income, shopping habits etc. We then find out they do not drive a car, so we end the questionnaire. The command *ENDNGB means that **data will be written**, but the interview will be marked as “not successful”.

```
*QUESTION 1 *CODES 61L1
Do you have a driver’s licence?

1: Yes
2: No *GOTO 998

*QUESTION 2
* Entire questionnaire here

*END

*QUESTION 998
Thank you for your participation thus far. Unfortunately, we
are only looking for people who drive cars. I therefore have
no questions for you.

*ENDNGB
```

*END

If a respondent completes our entire questionnaire, we consider them a “complete”, a “successful” interview, which must be stored in the data and marked as such. *END is a command that is therefore usually placed at the end of the entire questionnaire, at the very bottom of your script.

```
*QUESTION 999
This concludes our interview. Thank you for your time and
consideration. Have a nice day.

*END
```

Assignment 6

Building on the previous created script end the questionnaire if the respondent is younger than 18 years

Result Codes

Result codes are survey exit codes which you can use to track how respondents exit the questionnaire. A result-code determines how the respondent left the survey. Did they complete? Did they screen-out? Or did they simply not finish?

We use a number of standard, system wide result codes. These codes cannot be altered.

The most common codes are:

- Successful (18): these are respondents that completed the entire survey, passed the screener and finished the questionnaire all the way to the last question. The command to use is `*END`.
- Dropped out (101): these are respondents that didn't finish the survey. They either dropped out during the survey (101) or they closed their browser (106).
- Screened out (19): these are respondents that didn't pass our screening. They either didn't fit your research criteria, or they were no longer necessary because we reached our quota targets already. The command to use is `*ENDNGB`.

Result codes (exit codes) are used by fieldwork managers to monitor progress and quality. Each completed, each "successful" interview is added to various targets. When fieldwork is done, and the data goes to the data processing department, they 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.

It's important to make sure your questionnaire has proper exits.

Assignment 7

Building on the previous created script rescore the age questions into groups and handle respondents refusing to disclose their age.

13. Temporary Variables

A question is a permanent variable. The answers to the question are permanently stored in the datafile (or the O-file). ODIN also allows you to store temporary information in variables. A variable is not a question. It's a temporary 'container', which captures information *only during the interview*.

The information stored in a variable is therefore only available during the interview, inside the script and *does not become part of the data*. If you want to permanently store the information in these variables, you will need to code these variables into a dummy question, which is called a dummy question.

ODIN distinguishes two types of temporary variables. Text variables (variables which contain text) and regular variables (variables which contain a code, a number).

*TEXTVARS

A text variable will contain a piece of text. This can be the answer to an open-ended question, or it can be the actual text of an answer option. You can also add text to such a variable yourself.

```
*TEXTVARS Gender

*QUESTION 1 *CODES 61L1 *SAVE Gender
What is your gender?
1: Male
2: Female
```

In this example we are storing the actual text of the answer, “male” or “female”, inside a new variable, called “Gender”. We first define this text variable, before the question is asked. We then ask the question and add `*SAVE` Gender to the question. The text of the actual answer given is now stored in this temporary text variable.

We can reference and display that text variable, by using the command `*?`.

For instance

```
*PAGE
So you are a *? Gender?
```

*VARS

You can also do this with numbers and punches. A regular variable will then contain a number (from a number question) or the actual punch from a predefined answer list.

```
*VARS Numkids

*QUESTION 1 *NUMBER 61L2 *MAX 10 *SAVE Numkids
How many children under the age of 19 do you have living at
home?

*QUESTION 2 *CODES 63L1 *IF [NumKids >= 1 ]
Do they all have their own room?

1: Yes
2: No
```

In our example we are using the command `*SAVE` which directly saves the information to our temporary variable. We can also use the `*PUT` command to put something inside a variable.

```
*VARS Age

*QUESTION 1 *NUMBER 61L2 *MIN 16 *MAX 99
What is your age?

*PUT Age Q1
```

All your questions will have numbers. This is called the **Question Identifier**, or **Question ID**.

So:

```
*QUESTION 1
```

```
*QUESTION 2

*QUESTION 3
```

… can be referenced throughout your questionnaire by their identifiers as `Q1, Q2` and `Q3`.

Please note: Variables **must have** a logical, sensible name, which makes them easily identifiable.

Please note: You can also create iterated variables or add an array to variables.

```
*Vars Brand[10]
```

You create one variable, with an array of 10. You can now reference these variables as: `Brand[1], Brand[2], Brand[3]` etc.

Assignment 8

Building on the previous created script create a total awareness car brand question using the spontaneous and aided awareness questions.

14. Storing Answers

We have discussed that we can store information in temporary variables, either as text or as numbers. This is a useful tool during fieldwork, to display information in questions, intros or answer options. The information stored however, is **not stored in the data**. It will often occur where you **do need to store information** in data.

Let's begin with making our lives easier. Most surveys will have questions that use the same answer options. Questions where people have to like or dislike something, or questions about brands for instance. Instead of constantly typing in, or copying the exact same answer list inside ODIN Developer for each question that uses the same answer options, we can create answer.

*LISTS

The `*LIST` command allows you to create a *response list* that can be used for multiple questions, throughout your questionnaire. Our earlier example with cars would then look something like this.

```
*LIST Cars
1: Audi
2: BMW
3: Citroen
4: Fiat
5: Ford
6: Hyundai
21: Other *OPEN *NOCON
29: None of these *NMUL *NOCON

*QUESTION 1 *CODES 61L29 *MULTI *RANDOM
Which of the following car brands have you heard of?
```

```
*USELIST Cars
```

The `*USELIST` command adds the actual *response list* to the question.

When dealing with questions where answers from previous questions need to be included or excluded, the “Other” and “None of these” options can become a problem. We’ve solved that by making them *NOCON and *NMUL in the actual list. You can achieve more flexibility by proceeding as follows.

```
*LIST Cars
1: Audi
2: BMW
3: Citroen
4: Fiat
5: Ford
6: Hyundai

*QUESTION 1 *CODES 61L29 *MULTI *RANDOM
Which of the following car brands have you heard of?

*USELIST Cars
21: Other *OPEN *NOCON
29: None of these *NMUL *NOCON
```

In the above scenario, we’ve left the “Other” and “None of These” options out of the actual list and added them as separate options on the question itself. Questions where an interviewer is reading out answers to a respondent, will not include any “Other” options, so this means the list can be used for more questions, without adding more programming or control.

*DUMMY

A dummy is a fake question. It’s a question that is never asked, but filled by scripters, to *permanently store information in the data*. A dummy question can contain anything, codes, numbers, even alphanumeric information.

Continuing with our example from above.

```
*QUESTION 1 *CODES 61L29 *MULTI *DUMMY
Total Brand Awareness

*USELIST Cars
21: Other
29: None of these

*QUESTION 2 *CODES 161L2
Thinking of cars, can you name the first brand that comes to
mind?

*USELIST Cars
21: Other *OPEN
29: None of these

*QUESTION 3 *CODES 171L29 *MULTI *RANDOM *CONTROL Q2 N
I am going to read out a number of car brands.
Which of these do you know?

*USELIST Cars
29: None of these *NMUL *NOCON
```

We have asked first brand mentioned. We exclude that answer from the following question, all subsequent brands with `*CONTROL Q2 N`, meaning show everything *except the answer given at Q2*. Now we add these two questions together in the dummy question, so we have Total Brand Awareness, stored in a single question, our dummy Q1.

Please note: This makes for much easier analysis by your data processing department once fieldwork is complete.

*INCLUDE/ *COPY / *EXCLUDE

The `*INCLUDE` command allows you to transfer data from a multi question into another (dummy) multi question. The `*INCLUDE` command **adds** codes. So any original codes already in the question are **not** overwritten.

The command `*COPY` allows you to do this for **single** questions and individual codes. `*COPY` also allows you to do this for `*ALPHA` questions. The `*COPY` command **overwrites** codes. Whatever was in the question before, will be overwritten.

In our example, we will fill Q1, our dummy, with the contents of Q2 and Q3.

```
*IF [ Q2 , 1-21 ] *INCLUDE Q1 Q2
*IF [ Q3 , 1-21 ] *INCLUDE Q1 Q3

*IF [ Q2 , 29 & Q3 , 29 ] *INCLUDE Q1 [ 29 ]
```

The `*INCLUDE` command allows you to copy an entire question into another question as well as allows you to copy individual codes.

In our case we have added an `*IF` clause. We don’t want “None of these” copied, since that answer is available twice.

So we only copy Q2 into our dummy if a brand is mentioned. We then do the same for Q3. We only copy Q3 into our dummy if a brand is mentioned.

If a person selects “none of these” twice, at both Q2 and Q3? We copy the actual “None of these” code into our dummy, namely code [29].

Our dummy, Q1, will now contain total awareness, a combination of Q2 and Q3.

Our dummy can now be used for future questions. For instance we want to find out, out of all the brands of cars somebody is aware of, which one they would consider buying?

```
*QUESTION 4 *CODES 171L29 *MULTI *CONTROL Q1 W *IF [ Q1 , 1-20
]
Out of all the brands you know, which would you consider
buying?

*USELIST Cars
29: None of these *NMUL *NOCON
```

If our respondent knows at least 1 brand from the pre-coded brands, we will include *only the brands they know*, from our dummy, `*CONTROL Q1 W`, and let them select which of those they might want to buy.

Another useful command here is *COUNT. You can count the number of answers given to multi questions.

```
*VARS numb
*COUNT numb Q1
```

We create a variable, we then *COUNT our total awareness dummy and see if there is more than 1 brand coded there. We store this in a variable called numb.

```
*QUESTION 41 *CODES 171L29 *CONTROL Q1 W *IF [ numb > 1 ]
Out of all the brands you know, which one is your favorite?

*USELIST Cars
29: None of these *NMUL *NOCON
```

We only have to ask Question 41 if our respondent knows more than one car brand, otherwise there is nothing to pick as far as a favorite is concerned.

The *EXCLUDE command is used to remove one or more codes from an existing question and works the other way around. *INCLUDE can be used for all codes questions (**both multi and single**), while *COPY can **only** be used for **single** questions. *EXCLUDE can be used for **both multi and single** questions.

`*EXCLUDE Q2 Q3` wil exclude all the answers given at Q3 from Q2.
`*EXCLUDE Q2 [ 29 ]` wil exclude code '29' from Q2.

15. Repetition

It will often occur that you will have to ask a question, or a set of questions *about* a specific brand, product or service. In our example from above, now that we've stored our *Total Awareness*, we may want to know if our respondent can rate all those car brands and give them a school figure, from 1 to 10.

This is done using a loop, or a repeat block.

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, rotated or inverted loops.

In nested blocks, the operator ?R returns the value of the loop where that is currently being run. To access values from loops above or below the current, 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 cannot 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 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 68L12 *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
```

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
```

*REPEAT

```
*TEXTVARS Car

*REPEAT 6 FIELD 61L12 *RANDOM

*PUT car Q1, ?R

*QUESTION 5 *NUMBER 1L2 *MAX 10 *IF [ Q1 , ?R ]
Can you please rate the car brands you know.
Think of a school figure from 1 to 10.
How would you rate *? Car
```

```
*ENDREP
```

Please note: The FIELD is allocated when you fix the positions of your questionnaire, before you go into field. You can read more about it below.

Please note: Inside the repeat block, the data positions are relative. So when we have a FIELD of 61L2, 61-62 will be 1L2 and then 63-64 will be 1L2 etc. The positions and length of the questions inside the repeat block are relative to the overall size of the repeat block.

We first define our text variable, in which we will store, temporarily, the name of the actual car brand, `Car`, so the interviewer or respondent know what car brand they have to rate.

We then define our repeat block, we do this with the `*REPEAT` command. We have 6 car brands, so we define 12 positions, 6 x 2 columns for the `*NUMBER` question. We want our respondent to rate the different car brands in `*RANDOM` order.

We then use *PUT to copy the actual text of the car brand from our dummy Q1 into this temporary variable, `Car`. Each cycle through the repeat block is called an *iteration*. Our loop has 6 brands, so 6 iterations, from 1 to 6. The internal ODIN variable `?R` stores the iteration number. So as we cycle through the repeat block, we copy iterations 1, 2, 3 etc. into the temporary variable `Car`.

Question 5, inside the repeat block will be asked 6 times, `*IF` the brand is known and stored in the Q1 *Total Awareness* dummy. As we cycle through the repeat block, every time we check `*IF [ Q1 ]` our dummy contains codes 1, 2, 3 etc, using the same internal ODIN variable which holds the iteration number `?R`.

`*ENDREP` ends our repeat block.

In this example we are using `*IF` to control Question 5, making sure it is only asked if our respondent is indeed aware of that brand and has that brand stored in our Q1 total awareness dummy.

There is another way to achieve this control.

```
*TEXTVARS Car

*REPEAT 6 FIELD 61L12 *RANDOM *CONTROL Q1 W

*PUT car Q1, ?R

*QUESTION 5 *NUMBER 2L2 *MAX 10
Can you please rate the car brands you know.
Think of a school figure from 1 to 10.
How would you rate *? Car

*ENDREP
```

As you can see, we can control our repeat block making sure only the brands stored in our Q1 total awareness dummy are shown to the respondent, `*CONTROL Q1 W`. We therefore no longer need the `*IF [ Q1 , ?R ]` check on Question 5.

Assignment 9

Building on the previous created script add a question(s) for each car brand a respondent knows if the respondent has ever driven a model of that brand.

*MATRIX

Matrix Questions

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.

Let's explore some the syntax and some scenario's in more detail.

Syntax

```
*MATRIX n Qn [W|N] [*FIELD <pos>L<length>] [*ROT|*INV|*RANDOM|*ORDER]
Question text
<any question type including all the options that a question can have. Excluding 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	Default: only answers not mentioned in Qn will be displayed.
`*FIELD <pos>L<length>`	Defines the data positions.
`*ROT|*INV|*RANDOM|*ORDER`	Defines the order in which the statements are to be shown.

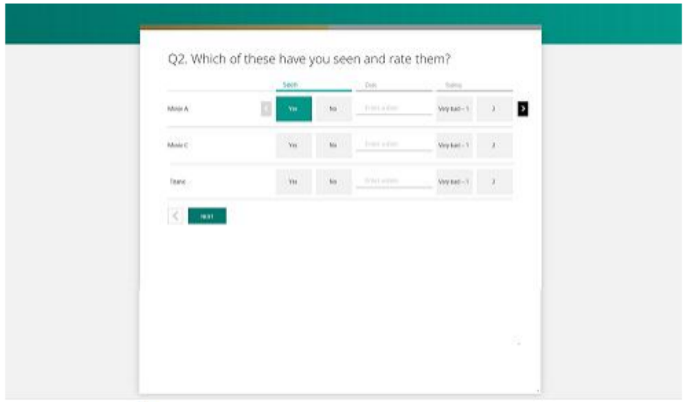
Expression Operator for Matrix

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.

Please note: *INV is no longer supported.

Let' explore some examples. 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.


Q2. Which of these have you seen and rate them?

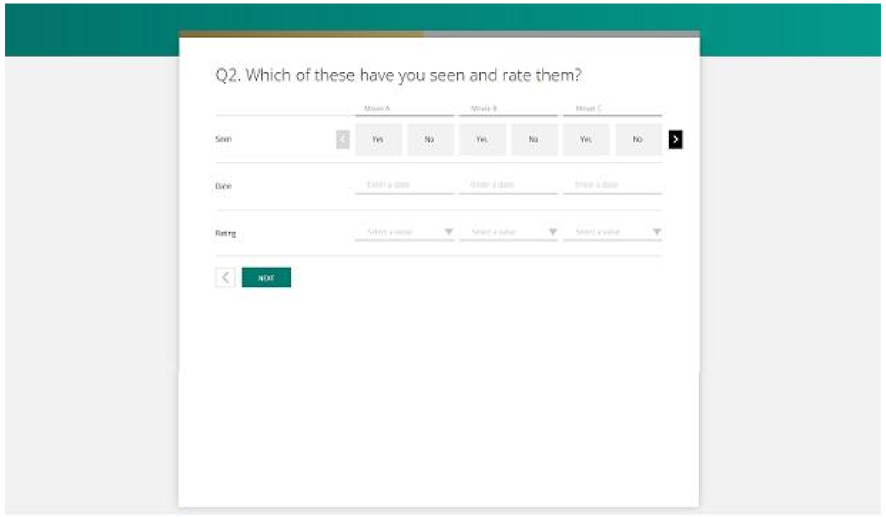
Let's take a look at how we would script this using ODIN for Nfield.

```
*LIST "movies"
1: Movie A           ** category text becomes the 1st statement
2: Movie B           ** category text becomes the 2nd statement
3: Titanic           ** category text becomes the 3rd statement
4: Other *OPEN       ** open answer becomes the 4th statement
*QUESTION 1 *CODES 61L4 *MULTI *LIST "movies"
Q1. Which movies would you like to watch?
*MATRIX 4 Q1 W *FIELD 65L48
Q2. Which of these have you seen and rate them?
*QUESTION 2 *CODES 1L1
Seen             ** this becomes the parent column header
1: yes           ** category text becomes the answer box column header
2: no            ** category text becomes the answer box column header
*QUESTION 3 *ALPHA 2L10
Date             ** this becomes a column header

*QUESTION 4 *CODES 12L1
Rating           ** this becomes the parent 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

** Using the expression operator
*QUESTION 5 *OPEN 100L1 *IF[Q1,1 & Q2S1,2] ** will only ask this question if Movie A was not
                                            watched but mentioned as movie that would like
                                            to watch
Why did you not watch Movie A?
```

Our Matrix with dropdown looks like this::


Q2. Which of these have you seen and rate them?

Multiple Choice Matrix

```
*TEXTVARS other

*LIST "movies"
1: Movie A                              **Answered
2: Movie B
3: Movie C
4: Other *OPEN                  **Answered - Titanic

*LIST "aspects"
1:exciting                              **Answered
2:boring
3:other *OPEN *SAVE other               **Answered - Funny

*QUESTION 1 *CODES 61L4 *MULTI *LIST "movies"
Did you see?

*QUESTION 2 *CODES 65L4 *MULTI *LIST "aspects"
What sort of things do you look for in a movie?

*MATRIX 4 Q1 W *FIELD 69L48

*QUESTION 3 *CODES 1L1 *CONTROL Q2 W
Movie Aspects
1:exciting
2:boring
3:*?other
*ENDMATRIX
```

Please note: This is the way things look in the ChicagoTemplate.

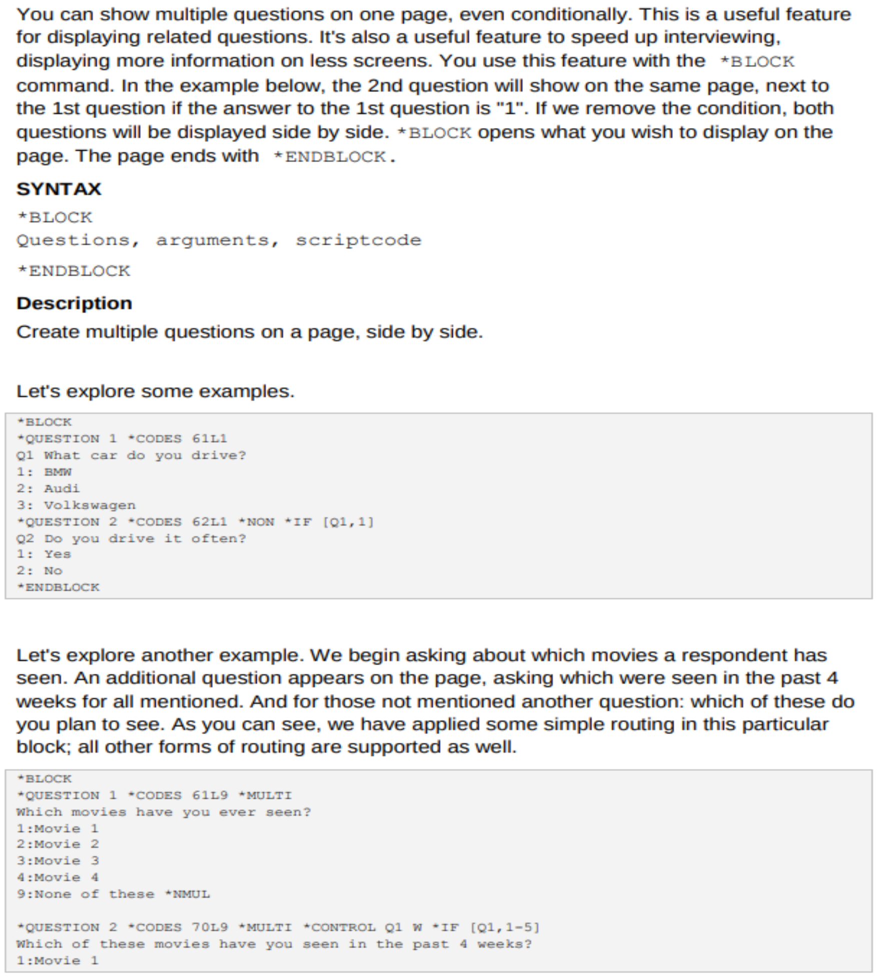
Assignment 10

Building on the previous created script add a Matrix question asking for a rating, of 1-10, for all known car brands.

Assignment 11

Building on the previous created script rework the Matrix question created in the previous assignment so all car-brands that are known AND driven by the respondent are scored on the 10-point scale.

*BLOCK



You can show multiple questions on one page, even conditionally. This is a useful feature for displaying related questions. It's also a useful feature to speed up interviewing, displaying more information on less screens. You use this feature with the `*BLOCK` command. In the example below, the 2nd question will show on the same page, next to the 1st question if the answer to the 1st question is "1". If we remove the condition, both questions will be displayed side by side. `*BLOCK` opens what you wish to display on the page. The page ends with `*ENDBLOCK`.

SYNTAX

```
*BLOCK
Questions, arguments, scriptcode
*ENDBLOCK
```

Description

Create multiple questions on a page, side by side.

Let's explore some examples.

```
*BLOCK
*QUESTION 1 *CODES 61L1
Q1 What car do you drive?
1: BMW
2: Audi
3: Volkswagen
*QUESTION 2 *CODES 62L1 *NON *IF [Q1,1]
Q2 Do you drive it often?
1: Yes
2: No
*ENDBLOCK
```

Let's explore another example. We begin asking about which movies a respondent has seen. An additional question appears on the page, asking which were seen in the past 4 weeks for all mentioned. And for those not mentioned another question: which of these do you plan to see. As you can see, we have applied some simple routing in this particular block; all other forms of routing are supported as well.

```
*BLOCK
*QUESTION 1 *CODES 61L9 *MULTI
Which movies have you ever seen?
1:Movie 1
2:Movie 2
3:Movie 3
4:Movie 4
9:None of these *NMUL

*QUESTION 2 *CODES 70L9 *MULTI *CONTROL Q1 W *IF [Q1,1-5]
Which of these movies have you seen in the past 4 weeks?
1:Movie 1
```

Assignment 12

Building on the previous created script add a block asking if the respondent owned one of the driven brand and if so which one.

16. ODIN 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.

Questions

Question numbers are made up of positive numbers (>= 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. 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
`*QUESTION 1`	`*QUESTION 0`
`*QUESTION 2`	`*QUESTION -2`
`*Q 3`	`*QUEST 3`
`*QUESTION 2501`	`*QUESTION 25a`
`*QUESTION 109`	`*QUESTION 10.9`
`*QUESTION 11001`	`*QUESTION 11.001`
`*QUESTION 102`	`*QUESTION 1-2`
`*QUESTION 2147473647`	`*QUESTION 2234567890`
`*QUESTION 1234567890`	`*QUESTION 1 234 567 890`

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. Omitting starting positions is not allowed for questions that refer to data fields in the sample table. These starting positions always have to be defined.

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 99999. Make sure your analysis program can handle the data size.

Please note: Always fix the positions of your questionnaire/script BEFORE you upload it to Nfield.

The first 60 positions of the data filed are reserved for system data

Answer Codes

Answer codes are recognized by a number followed by a colon. This number has to be the first non-blank character at the beginning of a line. Code numbers can be used indifferently. Code numbers have to be unique within a question. If the code text does not fit on a line, it is allowed to continue on the next line. All the following lines will be considered as part of the code until the first blank line.

Answer code numbers may range from 0 to 2147483647 for single-coded questions and from 1 to the remaining positions available in the U-file for multiple-coded (`*MULTI`) questions.

`*HEADING` texts may be provided to group sections of codes, but headings are not considered part of the answer labels.

Example

```
*QUESTION 1 *CODES L2
This is the question text.

Long lines in question and code texts will automatically be
truncated at complete words if the line exceeds the screen
width.

Question text and codes may contain empty lines. Empty lines
between codes are removed if *AUTO command is used AND the
text is too long to fit on the screen.

1: This is category one
2: This is category two
5: This is category
 five

7: This is category seven

 running over three

 lines

This line will be considered as text

11: This is category eleven

*END
```

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` or `*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 content can be displayed with the `*?` operator. Note that the content of the variable depends on the type of variable.

A numeric variable (`*VARS`) may contain any value. Use the `*FORMAT` command 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`.

Note: The content of variables is not stored in the data file automatically. So make sure that if you base questions or routing on a variable, you store this information.

Example 1

```
*TEXTVARS Gender

*QUESTION 1 *CODES L1 *SAVE Gender
Interviewer: Mark gender:

1: Male
2: Female
```

If code 1 is selected the text variable `Gender` will contain the text Male. If code 2 is selected, the variable `Gender` will contain the text 'Female'.

Example 2

```
*VARS Gender

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

1: Male
2: Female
```

If code 1 is selected, the numeric variable `Gender` will contain the value 1. If code 2 is selected, the variable `Gender` will contain the value 2.

Example 3

```
*TEXTVARS Gender

*IF [ Q1,1 ] *PUT Gender "man"
*IF [ Q1,2 ] *PUT Gender "woman"
```

If question 1 contains code 1, the variable `Gender` will contain the text 'man', else the variable `Gender` will contain the text 'woman'.

Example 4

```
*VARS Gender
```

```
*IF [ Q1,1 ] *PUT Gender "man"
*IF [ Q1,2 ] *PUT Gender "woman"
```

The syntax check gives a warning message, because a text is written in a numeric variable. After the filter is checked, the contents of the variable will be 1 (regardless of the outcome of the filter).

Example 5

```
*VARS Age

*QUESTION 2 *NUMBER L2 *SAVE Age
What is your age?
```

After question 2 is answered, the variable `Age` will contain the value that entered at Q2.

Example 6

```
*TEXTVARS Age

*QUESTION 2 *CODES L1 *SAVE Age
What is your age?

1: 18 - 24 years
2: 25 - 34 years
3: 35 - 44 years
4: 45 - 54 years
5: 55 - 64 years
6: 65 years or older
```

```
9: won't tell
```

After question 2 is answered, the variable `Age` will contain the code text of the code that was selected.

Example 7

```
*VARS Age

*QUESTION 2 *CODES L1 *SAVE Age
What is your age?
1: 18-24 years
2: 25-34 years
3: 35-44 years
4: 45-54 years
5: 55-64 years
6: 65 years or older

9: won't tell
```

After question 2 is answered, the variable `Age` will contain the code number of the code that was selected.

Example 8

```
*VARS SelectBrand
*TEXTVARS Brand

*QUESTION 31 *CODES L1 *DUMMY
```

```
Here 3 brands are specified. The system will select 1 brand
randomly.

1: Brand A
2: Brand B
3: Brand C

** Select a number below 3 and then add 1 to the result
** (this will randomly select 1 through 3)
*PUT SelectBrand [ (RAN 3) + 1 ]

** Now put the category text of the dummy question in
** the text variable Brand
*IF [ SelectBrand = 1 ] *PUT Brand Q31,1
*IF [ SelectBrand = 2 ] *PUT Brand Q31,2
*IF [ SelectBrand = 3 ] *PUT Brand Q31,3

** Now save the contents of SelectBrand in the datafile.
** If you forget to do this, you will not be able to know,
** which brand was selected during the survey.

*COPY Q31 [ SelectBrand ]
```

17. Look and Feel

We briefly mentioned that - as a scripter - you should consider *your audience*.

You should also *consider your platform*.

You need to consider **WHO** will be filling in your questionnaire.

Are they interviewers or are they respondents? And if they are respondents? What kind of respondents? The people who will fill in your questionnaire are *your audience*.

You also need to think about **HOW** they will be filling in your questionnaire.

Are respondents filling it in themselves on their PC, on their laptop, on their phone or with pen and paper? Are interviewers conducting interviews and filling in your questionnaire via telephone, an app, or via pen and paper? The mode of filling in, telephone interview, personal interview with an app, pen and paper, self-completion via laptop, PC or pen and paper. That is called your platform.

Finally: if interviewers are filling in your questionnaire *on location*, you need to consider WHERE they are filling in your questionnaire.

We have so far focused our attention on understanding basic ODIN commands to build a questionnaire consisting of various types of questions and some logic to make the questionnaire run smoother.

Now that you understand the basics we will introduce a new element, namely Look and Feel. ODIN was originally developed as a straightforward programming language for telephone interviewing. Using regular ODIN code will create a very basic looking questionnaire, which is ideal for telephone interviewing and allows for fast input, while talking to people on the phone. If you are dealing with respondents directly however, or interviewers who deal with respondents face to face... the way your survey **looks and feels** becomes much more important.

To give you a simple example: a brand list with 100 brands will never work when fieldwork is conducted using a mobile phone. Never. It will simply not fit and make for a horrible experience for the respondent. We can name numerous other examples.

A questionnaire with a functional, yet nice look and feel will mean a better experience for the person who fills in the questionnaire. When dealing directly with respondents it makes for a more engaging experience for those respondents and makes for higher quality data.

The look and feel of a questionnaire in Nfield CAPI for Android is decided by two factors:

- The **template** in use.
- The **theme** applied to the selected template.

Templates and themes must be selected in NIPO ODIN script. This section provides an introduction to templates and themes and examples for use.

What are templates for?

In ODIN you use templates to define the structure of your questionnaire pages. Templates allow full control over the placement of text and components on questions.

Templates can also change the rendering of questions. For example, a numerical question is by default rendered as an input box where a number may be entered. Using a different rendering, a numerical question may be changed into anything conceivable within the template, such as sliders, turn-knobs, spin wheels and drag and drop items, which all can help boosting the overall response to a survey.

Each template comes with its own template-specific rendering controls, integration with social network services (if any), and requirements for creating or changing themes.

What are themes for?

Using themes, you can apply changes to the look and feel of existing templates and customize them to the requirements of your research customer.

Themes set styling, colorization and other fine-grained customizations to the template in use. Themes differ from templates in the sense that where templates set the overall look and feel and introduce rendering controls for the various question types, themes are used to tune the template to the requirements of the survey, its client and the users.

Themes are based on a template. This means that a theme is specifically designed to work with one template and cannot work with another. Themes must be uploaded in the Nfield CAPI for Android Manager prior to use, but the NIPO ODIN script sets which theme is used when and where during an interview.

We offer extensive and interactive online documentation on how to render questions, making them look and behave differently so as to style your questionnaire.

Interactive Template Documentation

Please note: Our online and interactive documentation shows you the various types, options, what they look like on a pc, laptop, tablet or phone. It also allows you to copy out the ODIN script.

Please note: We urge you to take the time, study the interactive documentation carefully and familiarize yourself with the various options and the code.

Please note: As a starting scripter, we advise you to focus on scripting your questionnaire **first**. Properly test it. Make sure everything is working as it is supposed to. Check that data is being correctly stored. Only **after that**, add the look and feel.

Please note: styling a questionnaire is never necessary when dealing with telephone interviewing or data entry of pen and paper questionnaires. In those situations, you will want a basic, efficient ODIN script where an interviewer can input data.

Please note: When dealing directly with respondents - self completion - also called *web interviewing* (CAWI) you will want to style your questionnaire so as to engage the respondent.

Please note: When dealing with interviewers who are dealing directly with respondents, either during house visits, or at some location, you will want to style your questionnaire so as to make sure it all fits on a mobile device and is easy to use for interviewers to quickly input data.

*TEMPLATE

```
*TEMPLATE "[templatename[;themename]]"
```

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, or device used.

At present the most common used template is called NfieldChicago.

So:

```
*TEMPLATE “NfieldChicago”
```

… will have to appear at the very top of every script.

You can switch, during your script from templates and themes, allowing you to use multiple templates and themes in the same script.

There are 2 ODIN commands that you can place on every individual question, to determine their look and feel.

*UIRENDER

Sets the rendering control (visual representation and input interface) for all questions of a specific question **type**. `*UIRENDER` determines the overall look at feel of a **type** of question. So it determines how `*OPEN, *NUMBER, *ALPHA` and `*CODES` questions look and behave on any screen.

```
*UIRENDER "[questiontype]=[renderingcontrol]"
```

*UIOPTIONS

Sets the options for the current question with the currently active rendering control of a Question. So these regulate your options within a given question type.

```
*UIOPTIONS "[name1]=[value1];[name2]=[value2];{...}"
```

Look and feel depends greatly on a client's wishes, corporate branding and your audience.

Whatever styles and behaviour you choose for your questions: **make sure to be consistent**. If you're going to do a brand list with buttons? Do that brand list with buttons all the time. If you're doing a scale from 1 to 5 with a slider? Then do it with a slider *all the time*.

Consistency makes for faster response and input as well as higher quality data.

We're doing data collection, we're not doing web design, so be consistent, let styling be functional, consider your audience and consider the platform.

```
*TEMPLATE “NfieldChicago”

*UIRENDER “*NUMBERr=Slider”

*QUESTION 110 *CODES 61L3 *MIN[1] *MAX[10] *UIOPTIONS
“instruction=NumericalSlider”
What percentage of total budget did you spend on your
accomodation?

*END
```

Please note: `*UIRENDER` options must be reset to return to the default setting. `*UIOPTIONS` do not have to be reset.

Assignment 13

Building on the previous created and using the NfieldChicago template create an introduction in the Chicago look and feel (Welcome page).

Redesign the block question created in Assignment 12 into a NfieldChicago “Mixed matrix” asking for each driven car-brand if the respondent owned that car-brand and if so what model of that brand.

18. Background Variables

Market research is rarely done on just some random group of people. There is always a target audience, a specific group of people we wish to interview to investigate <topic X>. Usually we will be provided with a sample. A sample is a file that contains potential respondents. People whom we'd like to participate in our survey. The sample will usually be proprietary, meaning somebody else owns the sample and they gave it to us (perhaps our client) or they sold it to us (a sample provider).

Sample is quite expensive. We will have to invite or approach a lot of people, a lot of sample, to achieve only a few successful interviews. We want to approach as little people as necessary. The longer fieldwork runs, the more expensive our study becomes.
The sample file itself will contain respondent specific information, personalized information about the respondent.

A sample file will often be some type of excel file, which looks a bit like this.

Id	name	email	telno	gender
00001	John Smith	j.smitth@hotmail.com	+4499999999	M
00002	Jane Stone	j.stone@gmail.com	+4499999991	F
Etc.				

The various fields in this file, NAME, EMAIL etc, are called *sample variables*. The people in the file and the file itself are called *sample*, and the information contained therein, are called *sample variables*. Sample variables often provide valuable socio demographic or more personal information about our respondents.

Often enough this information is later used for analysis, by the data processing department, the researchers, even the client.

We therefore read these variables into our ODIN script, into our data, in the background, as our interview begins. That's why these sample variables are also sometimes called *background variables*.

You import the sample variables with one simple command.

```
*SAMPLEDATA name, email, telno, gender
```

By simply defining them, these variables are now imported into the ODIN script and can be used in the survey as *variables*.

```
*PAGE
According to our information, your gender is *? gender.
```

As we have learned, variables only store information *during the interview*, they **don't store anything in the data**. Only questions and their answers are stored in the data. If we wish to import these sample variables into our data file, we have to create dummies.

```
*QUESTION 101 *ALPHA 61L100 *DUMMY
Name
*COPY Q101 *? name

*QUESTION 102 *ALPHA 161L100 *DUMMY
Email
```

```
*COPY Q102 *? email

*QUESTION 103 *ALPHA 261L100 *DUMMY
Telno

*COPY Q103 *? telno

*QUESTION 104 *CODES 361L1 *DUMMY
Gender
1: Male
2: Female

*IF [ *? gender = "Male" ] *INCLUDE Q104 [1] ; *ELSE *INCLUDE
Q104 [2]
```

We have created 4 dummies, each with a question type (`*ALPHA/*CODES`) that fits their specific fields. We then use `*COPY` for `*ALPHA` questions (don't use `*INCLUDE`). Our `Gender` background variable we fill with an actual code, '1' or '2', with an `*IF` clause.

You can also update your sample, using the sample `*SAMPLEDATA` command. Let's say we received an old sample, we don't know exactly if their telephone numbers are still correct so we ask them.

```
*SAMPLEDATA telno

*QUESTION 103 *ALPHA 261L10 *SAVE telno
What is your telephone number?
```

Please note: This will update the actual sample file inside our interviewing systems. It will not change the file we were initially given by the client or the sample provider. The changes, the update occurs only in our systems.

As a beginning scripter, you will want to discuss what to import with your researcher, the client, the fieldwork department, and the sample provider.

19. Interviewing the "right" people

Quota

It will rarely happen that we're just interviewing random people from anywhere, how many we may find.

Typically every survey has a target. This target is the total number of people you want to interview for the *entire* survey. In market research, this target is called the *total target*.

In most surveys you do not just want to ask for example a random set of people their opinion.

You want to make sure these respondents represent the group you are seeking to answer questions for.

This is achieved by using quotas.

A quota is a specific subset of respondents with specific characteristics. This can be anything: a zip code, a city, a gender, an age group… anything. Quotas allow you to specify the subgroups for your survey and to set targets for each of them.

By referencing the quotas in your questionnaire script, you can screen out respondents during the interviews.

A quota can be this:

Total target:	100
Age: Young	40
Age: Old	40

In this case 100 respondents will be interviewed. At least 40 young respondents, 40 old respondents and 20 respondents that could either be young or old.

We offer more extensive documentation and tutorials on how to work with quotas.

Let's first consider what to use quotas for and when to use them. Quotas are a means to guarantee a distribution of sample you wish to interview. So when a client wants a distribution of 50% Male and 50% Female, you'd use quotas.

If you are merely trying to collect as much data, as many respondents as you can, then you would **not** use a quota.

If you are trying to find a very specific, hard to reach group, for instance all the people who bought a red car in Amsterdam between 2010-2011? Then you're probably going to be working with a complete customer/address list. You would then **not** use a quota.

Ask yourself, based on your target audience, your sample and your distribution: do I need to use a quota? If you do, then you begin by setting up the quotas, their cells and their targets.

In our example, we will be interviewing 100 people. Our survey will have an even distribution of men and women and age groups. To achieve this, we will be creating a quota for gender and give male and female a target of 40 each. And we will be defining different age groups.

Quota Variables, Quota Cells and Quota Check

We will start by defining our quota variables. A variable is a definition of a quota, in this example they will be "Gender" and "Age". Each quota variable consists of quota *cells*, in this example the cells for Gender are: Male and Female and for Age: Young and Old.

The quota variables and cells that you define in Nfield Online need to be called *exactly* the same as in the ODIN script. When you define a questionnaire variable in the Nfield Online Manager that is called 'gender", you **must** also have a variable in ODIN called 'gender'. When you enter a quota variable name, the Nfield Online Manager automatically suggests a questionnaire variable name for you.

In the below example you can see a simple quota set up in ODIN script. It defines the questionnaire variable "Gender" which is filed by data from question 1. The actual contents of the quota-variable, "Male" and "Female" are called quota-cells.

```
*SAMPLEDATA Gender, Age

*QUESTION 1 *CODES L1 *SAVE Gender
Are you male or female?
1:Male
2:Female

*QUESTION 2 *CODES L1
What's your age?
1: 16-24
2: 25-49
3: 50-64
4: 65+

*IF [Q2,1-2] *PUT Age "Young"
*IF [Q2,3-4] *PUT Age "Old"
```

Working with quotas always means collaborating with your fieldwork department. They will be using the exact same quota variables and cells, spelled *exactly* the same, to keep track of progress and performance. Our ODIN variable `Gender` will be used to count the number of people who fit this cell, it will be used to count the number of Males and Females during fieldwork. The same goes for our `Age` variable.

If you're doing an online survey or a face to face survey where interviewing is conducted on an app, using NIPO Nfield? Then you will want to take a look at the way quotas are set up by the fieldwork department using Nfield.

Please note: Take into consideration that once your survey is published you **cannot** change quotas or add new quota variables. You cannot change the **quota frame**, you can only change the **targets**.

Please note: All quota targets are *minimum* targets. If you are using quotas to achieve a certain distribution, this means you will most likely be weighing your data afterwards to resemble a specific universe. Weighing is done by the data processing department. This means that you will want to achieve your minimum targets with some oversampling to be safe.

Please note: The actual quota check takes place inside the ODIN script, using the `*STRAT` command. The respondent is usually checked **early** in the script to see if they fit our particular profile. This check is conducted as soon as the `*STRAT` command executes. The respondent is checked against their own quota and then checked to see if they may fit into another cell. If a respondent is accepted and still needed to reach our targets the respondent continues with the interview. It therefore makes sense to use the `*STRAT` command *right after the quota has been determined*. Once a person falls within a specific quota, you will want to check how many people you still need to interview in that quota. So you first identify a person and stick them into a quota cell, you then check that person against your overall target for that cell. So a person will become male and old first, you then check how many more *male* and *old* respondents you still need to interview.

The quota is counted *only when the respondent completes the entire questionnaire*. So your total counts and numbers go up, once a respondent finishes the interview and returns with a "successful" exit code (or a "screened-out" or "quota full" code).

If a respondent pauses the interview by closing their browser, the respondent can continue where they last left the interview. Before they return to their interview however, the quota check is conducted again, to check if we still need this person. If the quota cell this person was accepted for is already full, the person can be rejected anyway, even though they were accepted before.

From our example above.

```
*SAMPLEDATA Gender

*QUESTION 1 *CODES 101L1 Are you male or female?
1:Male
2:Female

*IF [Q1,1] *PUT gender "Male"
*IF [Q1,2] *PUT gender "Female"

*STRAT 999

*QUESTION 2
We are conducting a study about cars.
Our study will take a maximum of 10 minutes.

** Questionnaire

*QUESTION 998
Thank you for your participation, enjoy your day.

*END

*QUESTION 999
I am sorry you do not fall into the group of people we wish to
question.
We thank you for your participation.

*ENDST 21
```

Please note: `*ENDST` means end stratification. It's ending the questionnaire for the people who do not meet our quota-requirements. The code "21" means "quota full". The code is our result code.

We want 40 males. We first check into which quota cell the respondent falls. If it's a male, the `*STRAT` command checks the servers and fieldwork totals to see how many males are still needed for our study. If we already have 40, then the `*STRAT` command will return with "sorry, quota is full". The 999 redirects the respondent to a fake question, exiting the questionnaire as 'not successful'.
When the `*STRAT` executes, we are performing a *quota check*, on our *quota variable*, checking its *quota cells*, against what has been already collected during fieldwork.

Assignment 14

Create this interlocked online quota frame for Gender and Age:

Total: 10
- Male: 5
  - Male <18: 2
  - Male 18-35: 2
  - Male 35+: 1
- Female: 5
  - Female <18: 2
  - Female 18-35: 2
  - Female 35+: 1

Create a new script filling this quota frame and ending the questionnaire and quota full when the quota cell has been reached.

20. Future Proof (Future Waves)

It will happen quite often that a study is conducted more than once. Perhaps once a month, perhaps once a year. This means that your script must be futureproof.

Making a script futureproof is nearly impossible, but there are ways you make sure your script can be used for future research, by other scripters, researchers and even agencies.

Naming Conventions

You can give (text) variables any name you want. Please give them sensible names. So the variable that counts the number of cars a respondent knows should not be called `X1122`, but `TotalNumCars` for instance.

If there are several scripters working in your company, you may wish to maintain so called *naming conventions*, making sure all your scripters name their variables the same way. That makes it easier to pick up scripts from colleagues.

Allocate Enough Space

As we have seen, brand lists can be quite lengthy. And as research continues, new brands may be added, other brands may be removed. We have seen that we can manage this by maintaining master brand lists in the questionnaire. Using the `*LIST` command means you can quickly add new brands in a single location, just in the list, and all your questions using that list, will be fine. If you do not use lists, you will have to make changes to numerous individual questions.

The trick to apply is to make sure your last codes: Other, Don't know and None of the above, always receive high numbers (punches), like so:

```
*LIST Brands
1: brand 1
2: brand 2
3: brand 3
4: brand 4
997: Other *OPEN
998: Don’t know *NMUL
999: None of these *NMUL
```

Please note: This also means I need to allocate a length of 999 for the questions that use this list. Now, 999 is a crazy high number. It will be very rare that a list goes up to 999 brands, because it then becomes difficult to manage. But use 197 to 199 then. This way you will allocate enough space in the data file for future research. When research is conducted several times a year, using a similar questionnaire, this is called a wave. Each time the research is conducted, you're conducting a new wave. By allocating enough space, you can maintain data integrity for future waves.

```
*QUESTION 1 *CODES 10L999 *MULTI *RANDOM
Which of the following brands do you know?

*USELIST Brands
```

In our first wave, we may begin with just 21 brands. But we can simply add brands as the research continues. As brands are removed, you can simply remove the entire line.

```
*LIST Brands
1: brand 1
2: brand 2
** 3: brand 3 ** Removed in wave 2, feb 2018
4: brand 4
997: Other *OPEN
998: Don't know *NMUL
999: None of these *NMUL
```

Comments

As a beginning scripter it is important to remember that your logic, might not be clear to all other scripters. It is therefore important to add comments to your code. This will help your colleagues understand what it is you are doing, but will also help you remember what it is you did, if one of your scripts ever returns a year later.

In ODIN you can add comments by using the ** command.

Whatever you type after that is considered a comment.

```
*SAMPLEDATA Gender

*QUESTION 1 *CODES 101L1 Are you male or female?
1:Male
2:Female

** Filling gender sample variable for quotacheck
*IF [Q1,1] *PUT gender "Male"
*IF [Q1,2] *PUT gender "Female"

*QUESTION 1 *NUMBER 61L2 *BUT 99 "Does not want to say"
What is your age?

*QUESTION 2 *CODES 63L1 *DUMMY
Recoded age to categories.
1:<18
2:18-25
```

```
3:26-35
4:35+

** filling categorized age dummy
*IF [Q1 < 18] *INCLUDE Q2 [1]
*IF [Q1 >= 18 & Q1 <= 25] *INCLUDE Q2 [2]
*IF [Q1 > 18 & Q1 <= 35] *INCLUDE Q2 [3]
*IF [Q1 > 35] *INCLUDE Q2 [4]
```

Using comments makes life easier for **everybody**: yourself, your colleagues, as well as our Global Support staff, should you ever require assistance.

21. Logical Expressions/Operators

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.

Expression Operators

Operator	Description	Use	Note
`Q`	Question reference	`Qn`	Reference to question n or its contents
`L`	Length of field	`nLm`	Position n, length m
`F`	Field test	`QxFn`	Contents of field (cell) number n within form-question x
`,`	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
`&`	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
<>	not equal to	n<>m	Results in 1 if true or 0 if false
?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, *INV or *ROT	QxR	
QxMy	order of mentioning number y in a *MULTI question	QxMy	
QxRy	display order Used in combination with *ORDERnumber y of a question, when using *RAN, *INV or *ROT	QxRy	

Question reference

The reference to a question by means of Qn where n represents the question number. For example, Q10 references question 10. If it concerns a code test then more code values may occur; these will be treated as a logical or.

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

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.

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 of 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 ]`
`*IF [Q20,3\4\5]`	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.	`*IF [ Q20,3,4,5 ]`
`*IF [ Q21=1,2,3 ]`	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 U-file contains a 2 or a 3.	`*IF [ Q21,1,2,3 ]`
`*IF [ Q21 = 1-3 ]`	Check if Q21 has code 1, 2 or 3 marked.	This checks if the value of Q21 equals -2 (1-3).	`*IF [ Q21,1,2,3 ]`
`*IF [ #Q21 = 1 ]`	Check if Q21 is not equal to 21	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.	`*IF [ #(Q21 = 1) ]`

22. Testing

Once your script is finished, you must thoroughly check and test it.

Check

Your first test is a syntax check. Simply select “Check” inside the ODIN Developer. Your questionnaire will be compiled into a real executable program. During that compilation process your script is checked for syntax errors. A syntax error is a piece of ODIN code that was “written” the wrong way, just like regular spelling errors. When you press “Check”, the ODIN parser will flag all the errors, including the line numbers on which those errors occur. The idea is to walk through all your errors, make the necessary corrections and run the “Check” again. This is a circular process that you repeat until you end up with 0 (zero) errors.

Please note: The ODIN Developer *only checks for syntax errors*, ODIN code that was “spelled” the wrong way. **It does not check for logical errors**. Any errors in your questionnaire logic you will have to detect and solve by physically testing your script.

Run

In the same menu, inside the ODIN Developer, select “Run”. Please note that this is for basic checking and testing only! We recommend you test using Nfield, *not ODIN*.

The next step is to check all the spelling of the actual question- and answer texts. You must check the interviewer instructions, or respondent’s instructions. And finally you check your script logic. By script logic, we mean the code that you have built around the various questions and answers. If you are filling a dummy with 2 questions? You will need to check that the dummy indeed gets filled. If you want to randomize an answer list? You will need to check that it’s indeed randomized. You do this by running through your own script various times. Try the different responses and see if your routing all works out. If you are storing information in variables, it makes sense to display some of these variables during testing to make sure they get properly filled. As shown above, you show the various variables by using the `? *Variable` command.

Generate Dummy Data

As we explained above, there are two main ways to program using the ODIN Developer. You can insert questionnaire objects, by right click and then select “Insert” followed by what object you wish to insert. Especially for beginning scripters, this saves a lot of time in actual scripting. As you do this you will notice that the ODIN Developer *doesn’t allocate any data positions*. If you insert a “Coded” question, it will show up as this:

```
*QUESTION ? *CODES
```

The idea is to build your questionnaire *first* and to worry about data positions later. Once you’re done you can select “Fix” inside the ODIN Developer. The “Fix” fixes the data positions and automatically allocates enough space based on your answer lists and types of questions. Once ODIN compiles and fixes the data positions you can generate dummy data. These are fake interviews conducted by ODIN. The idea is to run some 100 fake interviews so you can test if the data is being punched correctly. If you are storing the name of some brand in an `*ALPHA` question with 25 positions of length? Then you want to check the actual data file and make sure that there is indeed 25 positions of text in that file.

After you’ve used “Fix”, you can generate dummy data.

The ODIN Developer then generates a fake data file. This is called an U-file. Your ODIN questionnaire will be called `NameQ`. Your data file will be named `NameU` and your open answers file will be called `NameO`. The U file and the O file are your data. The Q file is your questionnaire.

You can then select “File” and “Open Data File” to physically look at the datafile. Each line in the datafile is called a record. Each record is an interview where we have written data for. It’s a simple matter of running your cursor along a single line of data. If you have allocated 61L1 for gender? Then at position 61 you should only find 1s and 2s for Male and Female. If you have allocated 62L2 for age from 16-99? Then you should only find 16-99 in positions 61-62 etc.

Once you have checked the syntax, physically tested the questionnaire, and made sure all is being punched correctly in the data, you can begin a “real” test.

What does it look like?

As stated before. You need to consider your *audience* and your *platform*. Who will be filling in this questionnaire: interviewers or respondents? And how will they be filling it in? Via an app, via telephone or on a PC/laptop or handheld device?

It's now time to test your questionnaire as if you're a respondent, or an interviewer, using the channel the interviews will be conducted on.

For personal interviewing, where an interviewer speaks to respondents in person and then inputs the numbers via an app, using data-entry? You will want to test your questionnaire, using the actual app, to make sure all looks okay. You will also want to test using the devices the interviewers will be using. Interviewers, fieldwork managers and account managers must also test.

For web interviewing, where respondents are filling in the questionnaire themselves, via their PC/laptop, or tablet/phone? You will want to pretend to be a respondent and fill in a questionnaire to see if all is clear and appears on screen as it should. You will want to test using different browsers and devices. Fieldwork managers and account executives must also test.

Finally, if you are using any quotas, you will want to check if your quotas are working properly: are people being added to the right cells? And are these cells updating as people complete interviews? This is something you will have to check with your fieldwork department.

Usually the client also tests themselves and then gives the final sign-off.

Once everybody has tested and we are sure that all is functioning as it should, we begin fieldwork.

23. Fieldwork

When fieldwork begins, we start our actual data collection; our harvesting of information. Fieldwork is the most expensive part of market research. A sample consisting of potential respondents often costs money and engaging respondents, either online, via telephone or personally is a costly endeavor as well.

The last thing you do before you deliver your script to the fieldwork department is **fix your positions**. If this is a new survey, not some follow-up wave to an old/existing survey, but a new survey, **you must fix your data positions before fieldwork begins**.

This is imperative.

The data we will be collecting during fieldwork will be **stored in exactly those positions**.

Once fieldwork begins, you must never ever change those positions.

Making Changes During Fieldwork

The life of a scripter is by no means perfect.

Changes *do occur during fieldwork*. Clients and researchers will often take a look at the initial results and then add or remove things to an existing questionnaire. If this occurs, you should

1. Mark the occasion. You can do this by adding a dummy question and fixing a number in it, like '1'. This is useful for data processing later. Every record which has that fixed '1' in the data, is from *after the changes*.
2. You should force any new questions that are added *after your existing data positions*.

Like so.

We have a questionnaire with 20 questions.

```
*QUESTION 1 *CODES 61L1
What is your gender?
1: Male
2: Female

*QUESTION 2 *CODES 71L99 *MULTI *RANDOM
Which car do you drive?
1: Alpha Romeo
2: BMW
3: Citroen
**etc
99: None of these *NMUL

** Questionnaire

*QUESTION 20 *ALPHA 1121L6
Please enter your zipcode

*END
```

If we decide to add a car brand during fieldwork? Then we do not have a problem. We simply add a code. This *does not affect the positions since we allocated enough space, namely 99 codes, length 99.*

If we decide to remove Q2 and add a Q11 however, *this will affect the positions*. We are removing a question and adding an entirely new one.

```
*QUESTION 1 *CODES 61L1
What is your gender?
1: Male
2: Female

*QUESTION 2 *CODES 71L99 *MULTI *RANDOM *DUMMY
Which car do you drive?
1: Alpha Romeo
2: BMW
3: Citroen
**etc
99: None of these *NMUL
** Questionnaire

*Q101 *CODES 1201L1 *DUMMY
From here changed fieldwork, added Q11 on February 2018
1: Fix feb 2018 change fieldwork, adding Q11
```

```
** Fixing '1' for changed fieldwork
*COPY Q101 [1]

*QUESTION 11 *CODES 1211L2
Do you own a dog?
1: Yes
2: No

** Questionnaire

*QUESTION 20 *ALPHA 1121L6
Please enter your zip code

*END
```

As you can see. We don't physically remove Q2. We simply turn it into a `*DUMMY` so it doesn't get asked and we keep the positions Q2, **71L99**, still intact.

We then add Q11. As we can see our highest position is **1121L6**, from Q20. We therefore allocate plenty of room and place Q101 on **1201L1, adding it to the data file, making sure it doesn't overwrite anything**. We mark the change by adding a '1' to our data file and we then add Q11 on **1211L1**.

It is **imperative** that you keep track of which positions are used and which are not when adding questions and codes or removing questions and codes. **You must make sure the positions stay fixed and that your data map stays the same. Otherwise data processing will be impossible**.

Please note: Once changes have been made to a questionnaire that is already in field it is important that you run through your entire testing process again. Changes may seem simple or even trivial, but they can unintended effects. Once a change has been made, all must be tested again to avoid any errors or problems during fieldwork.

24. Data Formats

The NIPO ODIN Developer allows you to export your data to a variety of statistical software packages. The following formats are supported:

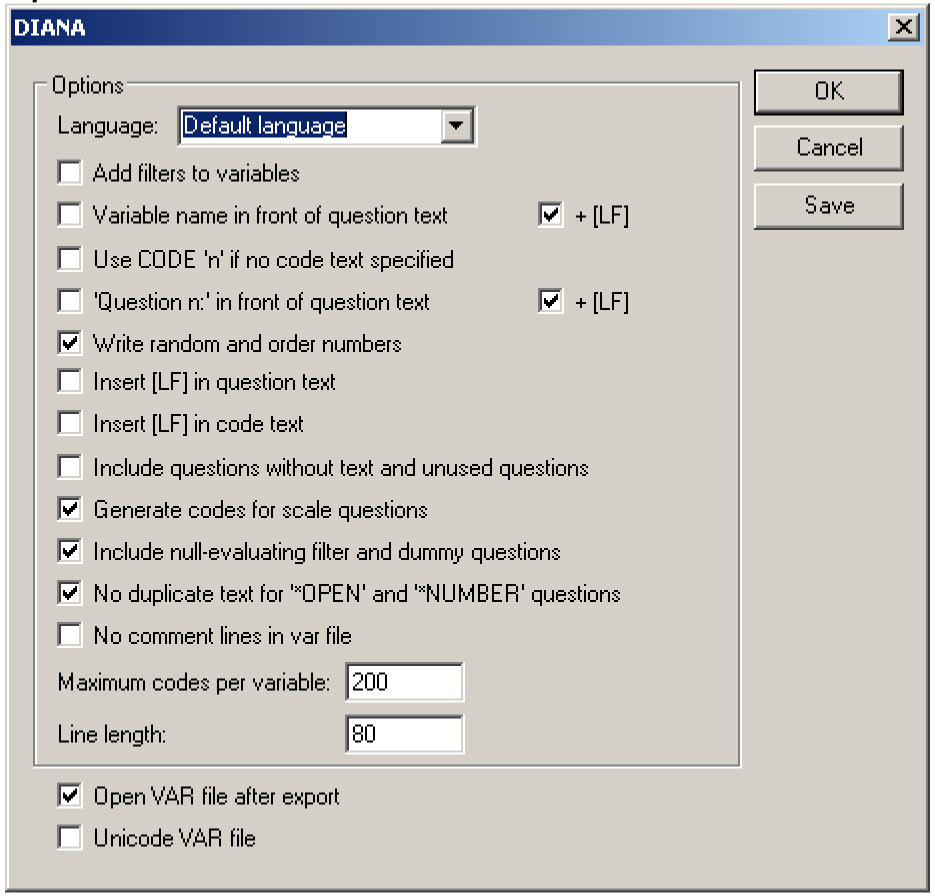
- NIPO Diana / Nvision Script
- Triple-S XML
- SPSS Portable
- SPSS-PC (script format)
- Ascribe
- Quantime
- Image / Sound

Please note: NIPO maintains a DCS which allows you to convert your data so you can directly process it in SPSS/Dimensions. Ask our helpdesk for more information.

NIPO Diana

The export to NIPO Diana may be used to create files that may be used by both NIPO Diana and Nvision Script.

Export to NIPO Diana variables



DIANA

Options

Language: Default language

- [ ] Add filters to variables
- [ ] Variable name in front of question text — [x] + [LF]
- [ ] Use CODE 'n' if no code text specified
- [ ] 'Question n:' in front of question text — [x] + [LF]
- [x] Write random and order numbers
- [ ] Insert [LF] in question text
- [ ] Insert [LF] in code text
- [ ] Include questions without text and unused questions
- [x] Generate codes for scale questions
- [x] Include null-evaluating filter and dummy questions
- [x] No duplicate text for '*OPEN' and '*NUMBER' questions
- [ ] No comment lines in var file

Maximum codes per variable: 200

Line length: 80

- [x] Open VAR file after export
- [ ] Unicode VAR file

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

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.

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.

Generate codes for scale questions
Generates code numbers for `*SCALE` questions. By default, these are exported as numerical variables.

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 only).

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
```

Random Code Number Export in NIPO Diana

Exporting random order

```
*QUESTION 2 *CODES 81L9 *RANDOM 90L3 *CONTROL Q1 N
Which of the following brands of beers do you know?
(Int: read out)

1: Heineken
2: Amstel
3: Grolsch
8: Other *OPEN *NOCON
9: None *NMUL *NOCON
```

Is exported as:

```
*V2 *MV 81L9:V2[LF]Question 2[LF]Which of the following brands
of brands of beers do you know? (Int: read out)
1: Heineken
2: Amstel
3: Grolsch
8: Other
9: None

*V2_R1 *SNG 90L1:V2_R1[LF]Question 2 - Random nr. 1[LF]Which
of the following brands of brands of beers do you know? (Int:
read out)
1: Heineken
2: Amstel
3: Grolsch
8: Other
9: None

*V2_R2 *SNG 91L1:V2_R2[LF]Question 2 - Random nr. 2[LF]Which
of the following brands of brands of beers do you know? (Int:
read out)
1: Heineken
2: Amstel
3: Grolsch
8: Other
9: None

*V2_R3 *SNG 92L1:V2_R3[LF]Question 2 - Random nr. 3[LF]Which
of the following brands of brands of beers do you know? (Int:
read out)

1: Heineken
2: Amstel
3: Grolsch
8: Other
9: None
```

Order Code Number Export in NIPO Diana

Exporting order of mentions

```
*QUESTION 1 *CODES 61L9 *MULTI 70L3
Which brands of beers do you know?

(Int: type the codes in the same order as mentioned by
respondent)

1: Heineken
2: Amstel
3: Grolsch
8: Other *OPEN
9: None *NMUL
```

Is exported as:

```
*V1 *MV 61L9:V1[LF]Question 1[LF]Which brands of beers do you
know? (Int: type the codes in the same order as mentioned by
respondent)
1: Heineken
2: Amstel
3: Grolsch
8: Other
9: None
*V1_O1 *SNG 70L1:V1_O1[LF]Question 1 - Order nr. 1[LF]Which
brands of beers do you know? (Int: type the codes in the same
order as mentioned by respondent)
1: Heineken
2: Amstel
3: Grolsch
8: Other
9: None
*V1_O2 *SNG 71L1:V1_O2[LF]Question 1 - Order nr. 2[LF]Which
brands of beers do you know? (Int: type the codes in the same
order as mentioned by respondent)
1: Heineken
2: Amstel
3: Grolsch
8: Other
9: None
*V1_O3 *SNG 72L1:V1_O3[LF]Question 1 - Order nr. 3[LF]Which
brands of beers do you know? (Int: type the codes in the same
order as mentioned by respondent
1: Heineken
2: Amstel
3: Grolsch
8: Other
9: None
```

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. Note that in a `*SCALE` or `*GRID` question the appropriate column headers are exported if tabs have been used correctly to separate these.

Example script 1

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

1: Street  *ALPHA 61L35
2: House number  *NUMBER 96L5
3: Postal code  *ALPHA 101L10
4: City  *ALPHA 111L30
```

Result export to NIPO Diana script 1

```
*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
```

Example script 2

```
*QUESTION 2 *FORM
What did you think of the service at the following gas
stations?

  Esso  Mobil Oil  Shell  Texaco
1:Very good  *GRID 141L5 6.1 4.10
Good
Average
Poor
Very poor
No opinion
```

In the script above, tabs are used to space column headers. Note that there is also a tab between the label and the `*GRID` on the first line, and that both the column header and the first line are ended with a space and tab.

Result export to NIPO Diana script 2

```
*V2_1 141L1: What did you think of the service at the
following gas stations?  Esso
1:Very good
2:Good
3:Average
4:Poor
5:Very poor
6:No opinion
*V2_2 142L1: What did you think of the service at the
following gas stations?  Mobil Oil
1:Very good
2:Good
3:Average
4:Poor
5:Very poor
6:No opinion
*V2_3 143L1: What did you think of the service at the
following gas stations?  Shell
1:Very good
2:Good
3:Average
4:Poor
5:Very poor
```

```
6:No opinion
*V2_4 144L1: What did you think of the service at the
following gas stations?  Texaco
1:Very good
2:Good
3:Average
4:Poor
5:Very poor
6:No opinion
```

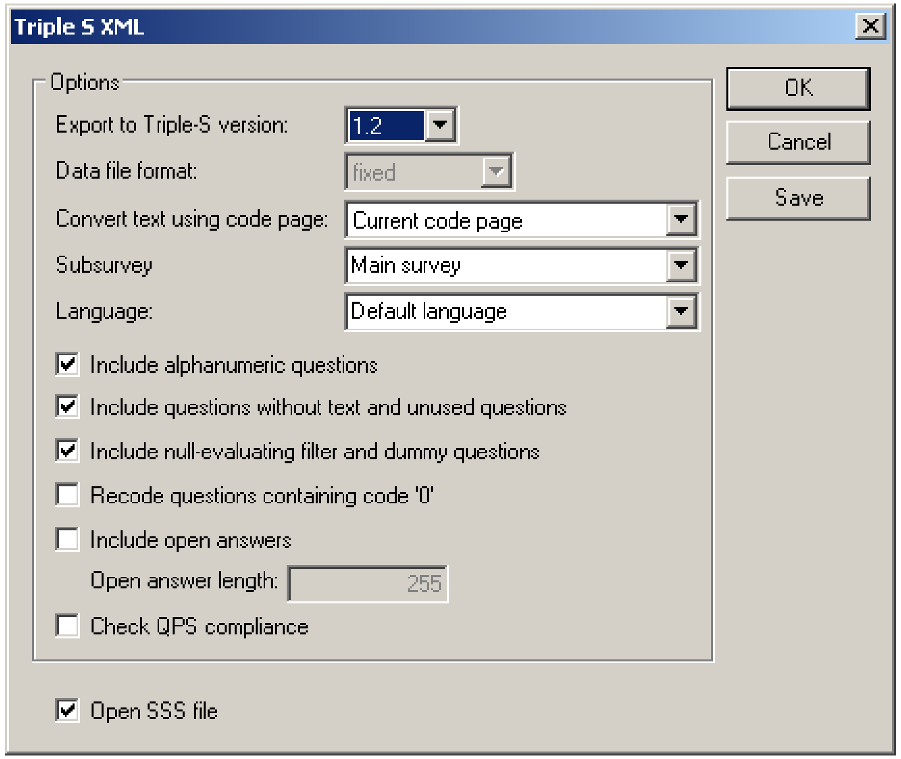
Triple-S XML

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

- Confirmit/Bellview
- CfMC Survent
- Merlin
- Miriad (TNS)
- 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



Triple S XML

Options

Export to Triple-S version: 1.2

Data file format: fixed

Convert text using code page: Current code page

Subsurvey: Main survey

Language: Default language

[✓] Include alphanumeric questions

[✓] Include questions without text and unused questions

[✓] Include null-evaluating filter and dummy questions

[ ] Recode questions containing code '0'

[ ] Include open answers

Open answer length: 255

[ ] Check QPS compliance

[✓] Open SSS file

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:

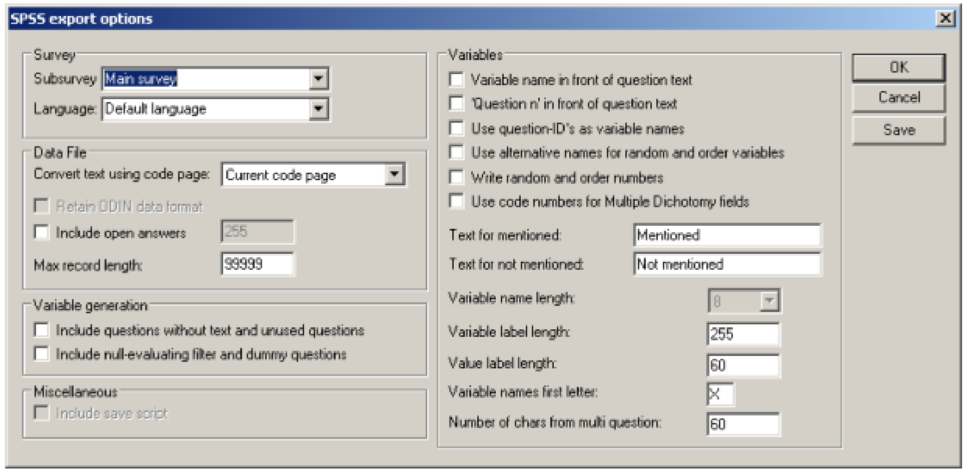
- < is replaced by &lt;

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

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


SPSS export options

Survey
Subsurvey: Main survey
Language: Default language

Data File
Convert text using code page: Current code page
☐ Retain ODIN data format
☐ Include open answers 255
Max record length: 99999

Variable generation
☐ Include questions without text and unused questions
☐ Include null-evaluating filter and dummy questions

Miscellaneous
☐ Include save script

Variables
☐ Variable name in front of question text
☐ 'Question n' in front of question text
☐ Use question-ID's as variable names
☐ Use alternative names for random and order variables
☐ Write random and order numbers
☐ Use code numbers for Multiple Dichotomy fields
Text for mentioned: Mentioned
Text for not mentioned: Not mentioned
Variable name length: 8
Variable label length: 255
Value label length: 60
Variable names first letter: X
Number of chars from multi question: 60

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

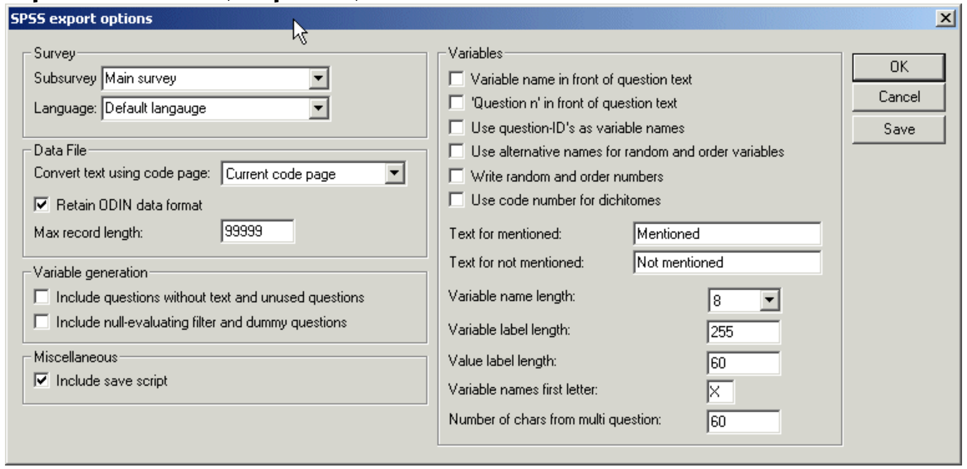
By default, multiple dichotomy questions receive the full original question text in addition to the code label. This option configures the maximum number of characters from the question label to include.

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.

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)


SPSS export options

Survey
Subsurvey: Main survey
Language: Default langauge

Data File
Convert text using code page: Current code page
☑ Retain ODIN data format
Max record length: 99999

Variable generation
☐ Include questions without text and unused questions
☐ Include null-evaluating filter and dummy questions

Miscellaneous
☑ Include save script

Variables
☐ Variable name in front of question text
☐ 'Question n' in front of question text
☐ Use question-ID's as variable names
☐ Use alternative names for random and order variables
☐ Write random and order numbers
☐ Use code number for dichitomes

Text for mentioned: Mentioned
Text for not mentioned: Not mentioned
Variable name length: 8
Variable label length: 255
Value label length: 60
Variable names first letter: X
Number of chars from multi question: 60

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

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

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. 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 '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.

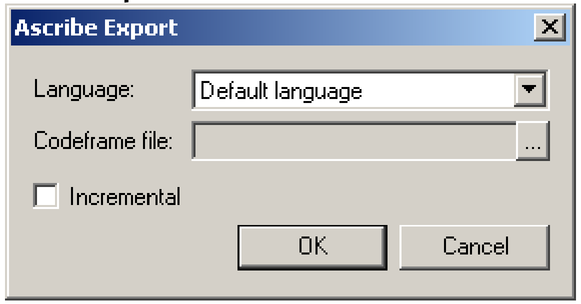
Note: The selected settings for variable names may cause the export to generate variable names that exceed the maximum number of characters for the SPSS version you are using. A warning is issued for the minimum limit for older versions - adjust the settings if required.

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

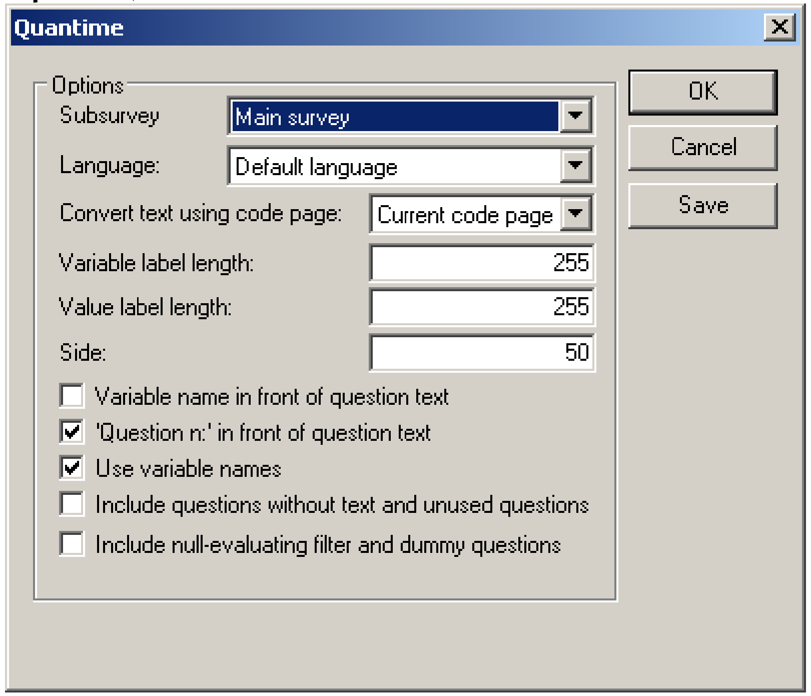
Check this option if you plan to add the current export to an existing Ascribe project. Use this feature to add new data to an existing project.

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.

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 `NAMEQ`, the data file should be named `NAME.DAT`.

Export to Quantime



Subsurvey: Main survey
Language: Default language
Convert text using code page: Current code page
Variable label length: 255
Value label length: 255
Side: 50
☐ 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

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.

Appendix 1 Terms of Use and Acceptable Use policy

Since this document is subject to (legal change), please use this link and carefully study this document. https://www.nipo.com/nfield-terms-of-use.
This document also has a link to the Acceptable Use Policy (AUP)

Resources

www.nipo.com
On our Customer Support Site, under Documentation, ODIN Developer, you can find more information, a full reference guide, the manuals and the template documentation. Just ask our helpdesk for a login.
Customer Support Site - Documentation

Congratulations!

You have successfully completed your ODIN Scripter training. We will check your final Assignment through the invites that you've sent us. Once that's done, we will send you an email and your certificate.

We hoped you enjoyed your training.

Welcome to the wonderful world of scripting.

nipo

NIPO
Amsteldijk 166
1079 LH Amsterdam
The Netherlands
hello@nipo.com

www.nipo.com