Askia Training

Course 410

Editing and Verifying Data

Participant’s Coursebook

With Tutor’s Notes

askia

Contents

Introduction

Format

This course provide supplementary training for those familiar with both askiascript (Course 150) and askiatools (course 400).

The course comprises three sessions, each intended to last approximately one hour, or possibly a little longer, if additional time is required to complete the practical work.

Each session follows the same format:

1.	Introduction (by tutor)	2-3 minutes
2.	Tutorial and demonstration	15-20 minutes
3.	Summary (by tutor)	2 minutes
4.	Practical exercises	variable
5.	Recap, feedback and questions	

Module topics

Session 410.1	Concepts of Edit Scrpts in Askia
Session 410.2	Applications for Edit Scripts
Session 410.3	Verification scripts

Course prerequisites

This course assumes that participants have knowledge of askiascript and, ideally, askiatools too. If this is not the case, they should take courses 150 and 400 first.

Course 150	Askiascript
Course 400	Askiatools

Recommended learning pathways

This course deals with two related aspects of scripting in Askia

1. Scripts and commands to edit data (i.e. change its values)
2. Scripts to detect errors or “verification scripts” where errors are reported but not changed.

While both aspects are useful to anyone working directly with Askia data in a data processing role, or as someone preparing data for analysis, the course has been written in such a way that both topics can be covered independently.

This means there are three possible pathways through this material:

All topics	Edit scripts only	Verification only
Session 410.1 Session 410.2 Session 410.3	Session 410.1 Session 410.2	Session 410.3

Preparation

When preparing for this course, you will need to agree the content you will be presenting with your client contact.

If possible, provide the training materials in advance so your client can prepare copies for each participant.

Course agenda and timetable

Once you have agreed the timetable or programme with your client contact, prepare a course agenda showing the start and end times of each session, as well as any breaks.

You should add these timings to the introductory PowerPoint, as well as providing it by email or on paper.

While updating the PowerPoint, modify the title page of the slides to show your name and, if possible, the client company's logo.

Prepare a one-sheet hand-out for the participants that give them the course timings – or email this to them. It is a good idea to have some printed copies. You may also have other specific items that you wish to add to this sheet or provide as separate sheets – e.g. guidance on their company-specific example project (see below) to be used in the exercises, info on how to access the system, and so on.

Example project

This course uses two different projects. It use the Askia International Credit Card survey, which your tutor will be using to demonstrate Askia Design to you.

In addition, you will have a second project which you will use to create a survey in Askia from the beginning. We have created a survey on the topic of Mobile Internet Usage, which covers, step by step, all of the material introduced in this course.

However, it may also be possible for you to use one of your company's own existing surveys as your first example of a survey written by you. Your tutor will need to check that the survey does cover all of the material that is required, and also that it is not too advanced for the topics that will be introduced in this course. In general, we recommend using our example survey, as it has been specifically created with your needs, as a new learner of Askia, in mind.

Our example project is designed to be capable of being administered as a Web survey, a CATI survey, a CAPI survey or as a mixed-mode survey. You will be able to set up those modes that are relevant to your own work and the kinds of surveys your organisation undertakes.

Example questionnaires

This course uses two different questionnaires and several shorter examples, used to illustrate specific features covered in the course.

Leisure Travel survey	Example questionnaire for use by the tutor; examples from this questionnaire appear in the Tutor's Guide and the Participants' Coursebook. This survey can be used for the tutor for demonstrating and answering any questions from course participants during the course.
Mobile Internet survey	A survey created for course participants to use in each end-of-module exercise. It is referred to in the Participants' Coursebook, and relevant sections are addressed in each exercise. The Mobile Internet survey should not be used by the tutor, unless directly answering questions that relate to the practical exercises.

Documents and files provided

These are organised into three folders:

Folder 1. Tutor example files

leisure travel.qes	Leisure travel survey, which contains around 20 questions including several in up to 3 nested levels of loops.
leisure travel.docx	A simplified readable version of the questionnaire showing question numbers and texts, and descriptions of the routing.

Five example qes or qex files used to illustrate specific features

coding open question example.qes

deleting records.qes

grid cleaning.qes

optimisation example.qex

coding open question example.qes

Three text files containing verification script syntax examples ready to copy and paste:

verify step by step.txt

verify open response length.txt

Folder 2. Exercise files for participants

A .QES file for each exercise populated with example data and example data in LDB format for exercise 3.

Folder 3. Model answers to exercises

Four files containing worked solutions to each of the exercises

Reset required?

Before teaching the course, obtain fresh copies of these files. If you have taught the course previously, or are accessing files that have previously been used to teach the course, the files will no longer contain the examples required when demonstrating the various data edits, as they will already have been edited. Ensure you start with the original, unedited version of these files.

Working with a client-provided questionnaire

We do not recommend using a client questionnaire for this course because all of the examples have created to demonstrate each topic covered in the course, and it is very unlikely that any client-provided survey will cover all of the topics

Where the client requests that their own survey is used during the training, our recommendation is that this survey is then set up as a second exercise *after* the core training has been completed.

As a preliminary step to this activity, you (as tutor) will need to review the questionnaire and identify any topics that will require additional training or explanation. Again, these should take place *after* this training course has been delivered and the exercises completed, in order to ensure that training participants receive a consistent, controlled training experience.

At the start of the training

Setting up the room

Arrive early. Ensure you arrive to give the training in plenty of time, so that you have time to set up the room as you would like it. Make it as tidy as possible – it is distracting to be learning in a room full of clutter.

Arrange the seating. Arrange all of the seats so that your group have a clear view of the screen and also that you have a clear view of them too – so you have facial contact with each of them. Set out enough seats for the group, and put away, or move to the back or against the wall, seats you don't want to be used, so it is obvious where you want everyone to sit.

Flipchart or whiteboard. Ideally, make sure you have a flipchart or whiteboard that you can also write up important messages or notes, and the right kind of pens (indelible pens for paper; dry-wipe pens for the whiteboard).

Heat. Make sure the room is **warm**, but not too warm, and that your group will be comfortable.

Noise. Ensure the door can be closed, so that the group will not be distracted by external **noise**, and also that you won't be distracting others in the office with your training session.

Light. Adjust the **lighting** – ensure the screen is bright and visible but not dazzled from overhead lights or daylight. Don't over-dim the room: you also need to be visible to the group and you need to be able to see them.

Course materials. Set out the coursebooks for everyone – and if possible, provide them with an Askia pen to use too (or at least, pens from the stationery cupboard). Also lay out the course timetable/agenda you have created, or any other supplementary sheets you have prepared (e.g. on the company-specific example project).

Demo screen and system access. Ensure you know how to work the data projector or large screen, that you have access to the system, to the Askia software and to the directories containing the training files. After any testing, close any apps so you are starting from a 'cold start' in your training session.

PowerPoint. Finally, set the display screen to show the first slide of the course PowerPoint set – in full screen mode – so that this is on the screen when participants enter for their training.

Introductory presentation

Before you start the main training you should:

Do introductions of everyone in the group. If time permits, also:

- Ask each to say what they do
- Ask each what experience they have in (a) Askia and (b) analysis and reporting of market research surveys
- Ask each to say what they hope to get out of the course

Explain the structure of the course – use the first three slides in the PowerPoint presentation *Course 410.ppt.*

Explain the timings: start times and end times – and when break times will occur

Explain the materials and resources they will be using:

- A participant's coursebook each
- Access to AskiaDesign
- Access to the training files

If necessary, explain where they will be able to find the files

Check that everyone has the materials and resources they need

Ask if there are any questions

Continue on to Session 410.1

Checklists

This check-list summarises the points described above. Please refer to the more detailed descriptions above, for more information.

During preparation

- ☐ Discuss course timings and participants with client and agree schedule
- ☐ Update agenda and timings in training course PowerPoint
- ☐ Discuss the example projects with client (you can forward the Word files) and review whether additional time should be allowed after the course to work on a "real" project.
- ☐ Obtain the example project and work through which questions and examples will be used in each exercise (see content checklist, above)
- ☐ If necessary, where no QEX file exists, import the survey and/or program those parts needed for the exercises to work.
- ☐ Agree who will print and bind the coursebooks for participants (double-sided and bound or in a folder is the preferred presentation)
- ☐ Ensure that there are printed copies of the Mobile Internet survey, for use by participants during the exercises.
- ☐ Prepare and print any supplementary notes – agenda, example project info, etc.
- ☐ Sort out access to the system, server etc at the client company – will you be using their PC or yours?
- ☐ Establish who will be able to update the IE settings on participant's PCs to access askiavista.
- ☐ Check the training room will have a large screen or projector you can use; Internet access for you to use; whiteboard or flipchart and pens
- ☐ Check there will be one PC for each participant or each pair of participants
- ☐ If you are not familiar with the system configuration and how Askia is deployed, check with the Askia team member responsible for the set-up, or with your client.
- ☐ Check any other arrangements with your client contact immediately before your visit – explain that you would like to have access to the room at least 30 minutes before your first session begins, in order to get it ready.

Using an client project for participant exercises

We do not recommend using a client project for the participant's exercises for this course, as it is very unlikely any project will demonstrate all of the capabilities covered in this course. A client project could possibly be used to supplement the exercises, if extended time is allowed for exercise work.

On the day

- ☐ Arrive early
- ☐ Arrange the seating and tidy away unwanted seats, clutter etc – make the room as tidy as possible
- ☐ Adjust the lighting and temperature as appropriate
- ☐ Check for noise or other distractions; check you will be able to keep the door closed
- ☐ Set out coursebooks, agendas and pens for all participants

- ☐ Check you have access to the software, the Internet, the example projects you need and any passwords required
- ☐ Check you have the latest version of the relevant askia software installed on your PC and on the participants' PCs (see Quick installation, below)
- ☐ Check whiteboard or flipchart – and that you have the right pens for each
- ☐ Ensure you have a glass or bottle of water to hand
- ☐ Ensure you know how to reach your client contact from the training room, in case of any problem
- ☐ Check PowerPoint slides load and have been edited for this client (and not for another client!)
- ☐ Finally, have PowerPoint open in presentation mode. This should be open, showing the welcome screen, when participants arrive

Software update, if required

The best way to ensure you have the latest Askia version of software, including the latest templates is to download and install the latest version. This can be found at https://dev.askia.com/projects/support/files. When installing the new version, make sure you uninstall the previous version of AskiaSuite (through control panel) before performing the installation.

Session 410.1 Concepts of Edit Scripts in Askia

Outline

Topics

In this session, you will learn about:

- The main reasons for editing or modifying data
- Edit scripts and the edit flag in Design
- Running edits in Askia Tools

Learning outcomes

At the end of this session, you will be able to:

- Define variables to perform simple edits
- Use Askia tools to perform an edit run
- Writing a simple edit command in Script

Tutorial

The main reasons for editing data

Definition: what is editing data?

→ Show first slide "*What do we mean by editing data"*

→ Explain: Editing data is about making changes to existing data to improve its accuracy or its usefulness. This could be either to correct something by changing its value to a different value, or by removing it entirely from the dataset.

Where the main purpose is to improve accuracy or reliability, the process may also be called "data cleaning".

When the purpose of the change is to reorganise it to make it more convenient for analysis and reporting, that process is sometimes called "recoding" or "reformatting.

All of these situations can dealt with in Askia by writing a script containing commands that will change the data, which, for convenience, we will call an "Edit script".

Cleaning data inconsistencies

→ Explain: if data have been scanned or keyed in from paper, there may be many logical inconsistencies in the responses to correct, such as questions answered that should not have been, or grids that are completed inconsistently.

Repairing scripting errors after data have been collected

→ Explain: For example, if a logic error meant that a question was sometimes asked when it should not have been, an edit script could blank out the responses that do not apply, to clean the data

Updating the data during fieldwork

→ Explain: For example, to add a quota part way through fieldwork, and populate the quota variable for all existing interviews, as if it had been there from the beginning

Recoding data to make it easier to analyse or present

→ Explain: Recoding is useful if presenting variables in askiavista in a different way to the way the data were collected, or to reformat data for export

Checking the data

→ Explain: We can run special scripts, called Verification Scripts, to test for many of the above problems in the data. We will look at these in the final session of the course. At the moment, we will assume that we are already aware of problems with the data, and that we are going to use data editing to rectify them.

■ **Notes for participants**

There are several reasons you might want to edit data:

- recoding data to make it easier to analyse or present in your tables and charts, or to reformat the data before exporting it;
- cleaning data inconsistencies, for example fixing logical inconsistencies in the responses where data have been scanned or keyed in from paper questionnaires;
- repairing bad data caused by scripting errors, after data have been collected, for example, blanking out responses where a logic error meant that a question was sometimes asked when it should not have been;
- updating the data during fieldwork, for example to add a quota variable, populating it based on data already collected.
- checking the data, using verification scripts; we will look at these in detail in the final session of the course.

Important advice about editing live data

→ Explain: you should *never* perform data editing on a running survey task (in other words, on live data), as there is no way to undo your changes.

→ If you make a mistake, then interview data is at risk of being corrupted. The only way back may be to carry out more interviews, which will almost certainly incur additional cost. For some time-critical surveys, it may not even be possible to go back and do more interviews at a later time as these may no longer be valid.

→ To work safely, make a copy of the QES, or download the data from the CCA, before you edit any data, and perform your edits on the copy of the data.

■ **Notes for participants**

Note You should never perform data editing on a running survey task (in other words, on live data), as there is no way to undo your changes. Instead, before editing any data, you should copy the QES, or download the data from the CCA, and perform your edits on the copied data.

A note on data cleaning in askiaanalyse

→ Explain: An alternative to editing is to redefine or logically clean the data in askiaanalyse; this is known as a “cleaning script”. This does not change the stored data, but it may be sufficient if the purpose of the edit is to make corrections for reporting done in askiaanalyse or askiavista.

→ Data edits in tools are carried out at the respondent level; this cannot be done in askiaanalyse, where edits are done on the aggregated results.

- → This course is about those situations where cleaning the data logically with a cleaning script is not enough, and want to change the actual values in the data, at a respondent level, using the askiatools software.

Notes for participants

An alternative to editing is to redefine or logically clean the data in askiaanalyse via a "cleaning script". This does not change the stored data, but does change the aggregated results, and it may be sufficient if the purpose of the edit is to make corrections for reporting done in askiaanalyse or askiavista. This course, however, will show you how to change the actual values in the data at the respondent level, using askiatools.

Writing an edit script

Explain:

- → Before you create an edit script, always make a copy of your QES or data file. Then, if something goes wrong (the script doesn't have the result you expected), you can just go back to the original file and make another copy to work from. That way, no data is lost.
- → It is possible to run an edit script on live data (a running task in askiaCCA). This is very dangerous because you might affect the raw data. You should always download your data, or copy the QES file, rather than running edits on the live data.
- → Edits are performed by inserting a script into the Askia QES file, and the running that script.
- → You define your **edit script** in askiadesign, as a special type of routing instruction.
- → You run your edit script in askiatools, which updates the data accordingly.
- → An edit run will only execute scripts that have
  - o the *set value* action type, and;
  - o the *edit* flag set.
  - o No other routing instructions will be executed during an edit run.
- → Scripts with the *edit* flag are treated like any other routing and are executed during interviewing.
- → Note that set value scripts, will always be performed during data collection (e.g. in the Web or CATI interview), if their *edit* flag is not set. If their *edit* flag is set, they will only execute in edit mode.

Notes for participants

.

Always backup your data first! Before you create an edit script, always make a copy of your QES or data file. Then, if something goes wrong and you accidentally change more than you intended to, you can just go back to the original file and make another copy to work from. That way, no data are ever lost.

In order to write an edit script, you simply define a routing instruction in askiadesign. However, bear in mind the following points:

- Your edit script must use the **set value** action type.
- It should have the **edit flag** set (select the option **edits** on the script in askiadesign's routing mode).

Askiafield will treat an edit script as any other routing if it is present during a Web or CATI interview, for example, and execute it at the point that it finds it.

The purpose of the edit flag is to identify to askiatools which scripts should be executed. In other words, scripts without the edit flag will be ignored by askiatools but scripts with the edit flag will be executed by askiafield during an interview.

Creating an example edit script

Recoding a numeric variable into a series of intervals

→ Explain:

- In this example, we will recode a numeric variable into different categories.

→ Demonstrate:

- In **askiadesign**, open the file *leisure travel.qes.*
- Show the question Q3.
- Show the existing data to the participants in **askiaanalyse**:
  - Start askiaanalyse, and open the same QES.
  - In the **view** menu, select **data**.
- Now switch to **askiadesign** and create a new single-coded question called Q3_S, with the following responses:
  - None
  - 1-3
  - 4-6
  - 7-10
  - 11 or more

- Ensure your question is not shown during interviewing (de-select **visible during data entry**).
- Now create the edit routing in **askiadesign**, as follows:
    - In the toolbar, click **routing mode**.
    - Click **ins**.
    - Select the **edits** flag.
    - In **action**, select **set value**.
    - In **target question**, select **Q3_S**
    - Beneath **execute**, select **always**
    - Next to **value if true**, click **…** and enter the following script:

```
If(Q3=0) then
  Return 1
ElseIf(Q3<=3) then
  Return 2
ElseIf(Q3<=6) then
  Return 3
ElseIf(Q3<=10) Then
  Return 4
Else
  Return 5
EndIf
```

Running an edit script

→ Explain: we will now run the example edit script we just created.

→ Make a back up of the QES (explain that you should always do this before running the edit script, as it may change your data in ways that you didn't intend if you made a mistake when creating it).

- To do so, make a copy of the file in Windows explorer.

→ Now, run the script in askiatools, as follows:

- Start askiatools, and open the QES.
- In the **tools** menu, select **run the edits**, then **on all interviews**.
- Show the result of the edit: a new question, *Age_S*, is now present in the questionnaire structure (you can show this in askiatools or askiadesign).

→ Show the new data in askiaanalyse.

→ **Notes for participants**

All edit scripts are executed in askiatools. After creating your script as a routing in askiadesign (see above), and then taking a back-up of the QES, you then need to open the QES in askiatools in order to run the script.

Make a copy! Remember what we said earlier about always working on a copy of the data! Before running your script, **always make a back-up of your QES** in Windows Explorer. The script will possibly change your data, and you may need to roll back to the original data if something goes wrong (i.e. if the script does not have the intended effect).

To run your edits script:

1. If you have not already done so, make a back-up of your QES file before you continue.
2. Open your QES in askiatools.
3. In the **tools** menu, select **run the edits**, then the appropriate sub-command, e.g. **on all interviews**.

Notes:

- Askiatools runs all routings with the **edit** flag set (this flag is set in askiadesign's routing mode), and changes the data accordingly.
- No other routings are run during the edit run (in other words, routings without the **edit** flag set are *not* run).

Recap

→ Show slide "***Summary*** *Concepts of the Edit Script*"

In this first session you have learned how to:

- Identify situations where you may want to edit the data
- Write an edit script
  - Applying the Set value action
  - Selecting the edits flag
- Use askiatools to run your edit script
- Check your results in askia analyse
- The importance of backing up or making a copy before you edit any survey data

Exercise

→ Ask the participants to complete this exercise. Provide them with a clean copy of the file *Mobile Internet exercise 1.qes*, which is based on the mobile internet survey used by the tutor in course 200 Analysis. There is a model answer provided in the file *exercise 1 model answer.qes* which will be useful if the participants need help with the script. However, do not provide this to them in the first instance.

Please do the following exercise. Your tutor will provide you with the relevant file to use.

Converting numeric data into single-coded ranges

In this exercise, you will be coding the answers to Q8, which is a numeric question recording the number of apps installed by the respondent on their mobile device.

Follow these steps:

1. Open *Mobile Internet Survey - exercise 1.qes* in askiadesign.
2. Create a new, single-coded question called Q8_S, and add appropriate responses (invent appropriate numeric categories, e.g. 0-4, 5-10, etc.).
3. Remember to set your new question as *not* visible during data entry, to ensure it is not presented to respondents.
4. Create a new edit script (routing) to recode *Q8* into your new variable. The routing **action** will be **set value**, and the **target question** will be your new question.
5. Set the script to execute **always**.

6 Remember to select the **edits** option for the script, to ensure it runs as an edits script (and doesn't run during interviewing).

7. Next to **value if true**, click **…** and enter an appropriate script. If you need a hint, your tutor can provide a model answer.
8. Now open askiatools, open the QES, and run your edit script**.** In the **tools** menu, select **run the edits**, then **on all interviews**
9. If you have askiaanalyse installed, use it to view the new data, by producing a table that includes *Q8_S*.

Session 410.2 Applications for Edits Scripts

Outline

Topics

In this session, you will learn about:

- Edit commands for open questions
- Edit commands for grid questions
- Populating data in a live project
- Optimising the performance of your script

Learning outcomes

At the end of this session, you will be able to:

- Write an edit script to code open questions
- Correct completion errors in grids from scanned questionnaires
- Use a script to apply a quota check properly that has been introduced after a survey has gone live
- Identify ways to make a slow-running script run faster

Tutorial

Coding an open question

→ Explain:

- This method is best used when the list of possible responses is very predictable, e.g. a list of brands or names of cities.
- This is because you need to write a logical rule for each item to be coded.
- We will use a script to re-code an open-ended question, where respondents may have used more than one term for a particular brand. In this case, we will show a very simple example of re-coding mentions of "Coke" as "Coca-cola".

→ Demonstrate:

- Open the example file *coding open question example.qes* in askiadesign, so we can look at the scripts.
- Show the question `SOFT_DRINK_OTHER`, and explain we want to re-code any mentions here of "coke" into `SOFT_DRINK` as response 2 (Coca-cola).
- Show the routing for `SOFT_DRINK_OTHER`. Point out:
  - We have used the method , so that the script applies to any mentions of "coke", no matter what combination of lower and upper case characters has been entered.
  - We return the value `SOFT_DRINK+02`. This retains any other responses gives to `SOFT_DRINK`, and also selects response 2 (Coca-cola). If we had just set the value to `02`, then any other responses to this question would be over-written!

→ Remind:

- Remind participants of the importance of working on a copy of the data. If the pervious error had been made to live data or the only copy of a completed survey, the original values would have been irrevocably lost.

→ Close askiadesign.

→ Make a back up of the QES.

→ Open the QES in askiatools and run the script.

→ Show the results in askiaanalyse.

■ Notes for participants

It is best to use scripts to code open questions when the list of possible responses is very predictable, such a list of brands or city names. This is because you need to write a logical rule for each item to be coded.

When writing a script to deal with open-ended data, bear in mind the following points:

- As with all edit scripts, remember to set the **edit** flag on your routing, so that it is not executed during fieldwork, and can be run in askiatools as an edit script.
- Edit scripts must use the **set value** action type.
- Use the *lcase* function when you compare two text strings, so that differences in case are ignored (so, for example, *iphone* and *iPhone* are treated as the same in your script).
- If your script adds a coded response to an existing multi-coded question, remember to preserve any existing answers by using the `AND` operator (e.g. `mobile_brand+01`) . If you only specified `01` without referring `mobile_brand`, then all the existing answers will be lost and the variable will only contain the answer `01`.

Cleaning a grid

Using a script to clean inconsistencies in grid data.

→ Explain:

- We may sometimes want to clean up grid data. This is most likely to happen in a paper-based survey, where inconsistent choices have been recorded.
- In askiadesign, open the example *grid cleaning.qes*. Explain:
  - In this example, we have a brands purchased question, then a loop which asks how many times the respondent bought each brand.
  - However, the loop is being asked for every brand.
  - We can use an edit script to code "dk" into the question for loop iterations that should not have been asked.
  - Normally, this would have been prevented by good routing in the survey.
- In askiadesign, show the questions in the QES, then show the edit script.
- In askiatools, open the QES and then run the edit script.

■ Notes for participants

We can use a script to clean up inconsistent grid data. This might happen if there has been a mistake made when setting up the routing in the survey, or if keyed-in or scanned data has led to invalid data.

Your script can check whether data is present in a grid cell, and set the value accordingly. For example, if brands asks about brands ever purchased, and Q1 asks about the number of purchases in the last month, you would not want a value in Q1 to be present for a brand that the respondent has never bought. You might set up a routing instruction (setting the **edit** flag to make it an edit script) as follows:

Condition:
```
brands.answers.index hasnone brandloop.CurrentIteration
```

Value if true: `dk`

When the above script is run in askiatools, any iterations at Q1 for which the corresponding brand was not selected at *brands*, will be set to *dk*.

Populating data in a live askiafield project

Changing the data in a live askiafield project.

→ *Note to tutor: There is no practical demonstration for this example, as it would require access in real time to an instance of CCA, and that is undesirable and probably unachievable if you are giving this course at an operational site. Instead, this example is one for you to describe.*

→ Show the set of three slides starting with the slide "*Live editing scenario*"

→ Explain:

- Previously we said you should avoid editing live data. In this situation, we will show you where sometimes, being able to run an edit script on a live project while it is still in the field can be useful.
- I will describe to you how do this safely, without overwriting any original data.
- In this example situation, the researcher or client responsible for the project has decided to apply quotas *after* fieldwork has started. Because the survey went live without the quotas or the variables needed to apply the quotas, this is one occasion where a script on a live project will be useful.
- Although the variables and quota check can be added in after the fieldwork has started, the problem is that it will only start counting how many interviews fall into each quota cell from the moment the variable is added. We need a way to go back and populate all the existing interviews with their quota information – and this is where a script can be used.
- With a script we can add the new variable, and use a script to populate it in the existing records, based on the data found there.
- Explain that you should never perform an edit on any existing data, but create new variables and populate these with data from existing variables. That way, we are not changing the data that has already been stored, and they will always be there to refer to, in their original state.
- Always make a back-up copy before you run any edit scripts.

Notes for participants

You can also run edit scripts on live data. This process allows you to change the data in any question while fieldwork is still in progress..

One situation where this is useful is when you need to introduce a quota during fieldwork, but it is not already present in the survey. You therefore need to add the variable and populate it for existing records, based on the data in them.

Simply adding the variables and quota check without updating the data will not fix the problem, because any interviews done before the change was made will not get counted. A script, however, can be used to update any existing data in the SQL database and update the routing so it is correct for all new data collected from this point onwards.

Run a backup first! Our advice is always to work on a copy – even if you are working with live data. Make a back-up copy first. If possible, ensure the survey is offline temporarily, then run your edit and check that it has worked correctly, so you can restore the data from the backup in case you notice that any data have been lost.

Work on new variables

In this example, we will introduce a new variable to the data file, and work on that, so leaving the data in the other variables intact and without risk of being changed.

Working on new variables will also ensure original data is not overwritten or corrupted. As discussed before, the principle is to work on a copy of the original – this time by creating one or more new variables and computing the values you need by referring to existing variables but storing the calculated results only in the newly created variables, so that all original data are left untouched.

Process to follow

The procedure for running an edit script on live data is as follows:

1. Open the QEX in askiadesign by using the **edit working copy** or **get working copy** command in askiafield Supervisor.
2. Add your edit script as a new routing.
3. Ensure this new routing has the **edits** option selected.
4. In askiafield Supervisor, use the command **update task** to upload your changes back onto the server.
5. In askiatools, open the **file** menu and select **open askiafield task…**, then enter the connection details for your survey. For help on using this feature, please see the following topic in the Analysis Assistant:

   http://analysishelp.askia.com/opening_a_qes_file_or_sql_server_database$askiafield

   Note that if you are not running askiatools directly on the CCA server, you will also need a local copy of the QES file in order to open the task.
6. In askiatools, in the tools menu, select **run the edits**, and select **on all interviews**.
7. In askiafield Supervisor, right-click the task and select **reload survey/script only** to ensure the task is updated to reflect your changes.

Use a live edit only exceptional circumstances. Check carefully beforehand to establish there is no other feasible way to achieve the desired outcome. Test your script fully on a copy of the survey database and check that no other data have been affected by your changes before applying your script to live data.

Optimising your scripts

→ Explain:

- Edits can sometimes take a long time to run.

    - We can reduce the running time by ensuring that our code is efficient. As our script runs once for every record, the larger the data set, the more time we can save by making our script efficient.

→ Demonstrate:

- Open the QES *optimisation example.qes.*
- Show the first routing for `SOFT_DRINK_OTHER`. Point out the following:
    - We continually refer to the value in `SOFT_DRINK_OTHER`. This is more time-consuming than using a local variable.
    - There are several separate `IF` blocks, which all need to be evaluated, instead of one block with `ELSEIF` statements. With `ELSEIF`, if a condition is satisfied, the subsequent `ELSEIF`s do not need to be evaluated for that record.
- Now show the second routing for `SOFT_DRINK_OTHER`. Point out the following:
    - We now use a local variable, *other_drink* in our IF block, instead of continually extracting the information from the data set.
        - This is faster.
        - It also makes the `IF` block easier to read.
    - We now use `ELSEIF`, which, over many records, could save considerable execution time.
- Explain:
    - You should also check any loops in your script, to ensure that loops do not iterate unnecessarily.
    - It is better to use a WHILE loop rather than a FOR..NEXT loop, because as soon as the condition in the loop is satisfied the loop will exit. A FOR..NEXT loop will iterate a fixed number of times, and this may result in unnecessary additional iterations.

■ Notes for participants

Edits can sometimes take a long time to run. By making sure your code is efficient, you can reduce this run time. Your script runs once for every record in your data set, so the larger our data set, the more time you can save by making the script efficient.

When writing your scripts, bear in mind the following:

- When comparing a value multiple times, consider storing the value in a local `DIM` variable, rather than looking it up each time. For example, when repeatedly checking the value in a variable, first store this value in a local variable as follows:

  ```
  Dim favourite_brand
  favourite_brand = Q4.value
  If favourite_brand = 01 then
  …
  Elseif favourite_brand = 02 then
  …
  ```
  Etc.

- When comparing many values, use `ElseIf` constructions where possible, rather than multiple separate `If` blocks.

- Make sure any loops in your scripts are as efficient as possible and do not loop more times than is necessary.

- A `WHILE` loop is often more efficient than a `FOR .. NEXT` loop because as soon as the condition for the loop has been satisfied (i.e. the first time it is true) the loop will exit. This can save unnecessary iterations.

Recap

→ Show slide "***Summary** Applications for Edit Scripts*"

In this second session on applications for edit scripts, you learned how to:

- Code an open question
- Clean a grid which contains completion errors, because it was imported from an external source to askia
- Add a quota variable to a live project and populate it with data
- Take proper precautions if updating live data
  - by using new variables
  - and taking a backup before you start
- Make performance improvements to slow-running scripts
  - Using `ElseIf` and `Else` not separate `If` blocks
  - Using `While` instead of `For..Next` loops

→ Point out to participants that there is some information in their notes (under the heading *Further reading*, about a more advanced way to write auto coding instructions that include an element of "fuzzy matching". Then prepare participants for the exercise.

Further reading

If you are considering writing an Edit script to code simple open data fields such as brand lists, you may be interested in one of Askia's more advanced features. This deals with the very common situation where participants have made simple mistakes in the response they have typed in, which an exact match will fail to pick up. For example, several participants have entered these as their preferred beverage:

- Cocacola
- Coca cola
- Coka-cola
- Coca-coa

The recode example we looked at earlier in this session is set up to recode "Coke into "Coca-cola". It will fail to code any of these correctly, even though we can see these are all essentially spelling mistakes and should be classified as "Coca-cola" too.

One way to ensure these kinds of mis-matches are catered for would be to write your logic to recognise these and any other possible mistakes you can imagine. Even so, it will only be a matter of time before someone enters something you have not thought of.

A better alternative is to use a "fuzzy match", which Askia script supports though the `.DLDistance()` method. Using this, you can get your script to identify all the close matches where the text entered differs from your original by one or two characters. You can specify how much difference or "distance" you will tolerate in the command you write.

There is a knowledge base article tells you more about how to use this feature, either follow the link below,

http://support.askia.com/hc/en-us/articles/115005854405-Auto-Coding

or search the knowledge base for "Auto Coding".

If you have some spare time when you finish this exercise, take a look at the article, and maybe try out writing an Edit script using `.DLDistance()`to work on a few data records you have created containing some deliberate spelling mistakes.

Exercise

→ Ask the participants to complete this exercise. Use the file *Mobile Internet Survey - exercise 2.qes*. There is a model answer provided in the file *Mobile Internet Survey - exercise 2 model answer.qes* which will be useful if the participants need help with the script

Coding an open question

In this exercise, you will be coding the answers to Q9_OTHER, which is an open-ended question recording the model of the respondent's cell phone/mobile phone, into Q9.

Follow these steps:

1. Open *Mobile Internet Survey - exercise 2.qes* in askiadesign.
2. Create a new edit script (routing), which we will use to recode Q9_OTHER into Q9.
3. The routing **action** will be **set value**, and the **target question** will be Q9.
4. Set the script to execute **always**.
5 Remember to select the **edits** option for the script, to ensure it runs as an edits script (and doesn't run during interviewing).
6. Next to **value if true**, click **…** and enter an appropriate script. For the purposes of this exercise, we will just concern ourselves with possible mention of the iPhone; in a real script, you would need to cover other phone models as well.

   Code any mentions of "apple" or "ios" as iPhone (response 03 at Q9). Remember to retain any existing answers to Q9; do not simply over-write them.

   Also, remember to handle any variations on the case of the response (e.g. "iOS", "IOS" or "ios").

   If you need a hint, your tutor can provide a model answer.
7. Now open askiatools, open the QES, and run your edit script**.** In the **tools** menu, select **run the edits**, then **on all interviews**.

Session 410.3 Verification scripts

Outline

Topics presented

In this session, we will introduce you to:

- Accessing the verification scripts tool
- The Assert object
- Useful keywords when writing verification scripts
- Generating an automated verification script
- Testing and running your script
- Viewing the output

Learning outcomes

At the end of this session you will be able to:

- Write scripts to apply quality control checks on your data
- Operate verification scripts effectively, including testing them
- Detect erroneous or suspect data values in your survey data and assess the overall quality of the data
- Identify individual cases which require further action and determine what corrective actions to take

Tutorial

What are verification scripts?

What are verification scripts?

→ **Verification scripts** allow you to check the quality of your data. They can find bad data caused by mistakes in your routing, or by interview cheats (e.g. people rushing through the survey).

→ You can run verification scripts on

- test data,
- the first few records collected during fieldwork, or
- the final data set.

→ They do not change the data, but help you detect problems with the data set. Once your script has detected one or more problems, you then need to decide what to do (e.g. exclude bad records or change bad data).

- This is often done using edit scripts, which we looked at in the first session of this course.

→ A verification script consists of one or more custom checks on the data, known as **asserts**; if any asserts fail, you can view details when the script has finished running. You will be able to discover which data records have the detected problems.

→ You define verification scripts in askiatools, and run them on the data currently in the QES.

- You can run them on all interviews, or on a subset of the interviews.

→ Emphasise that this process does not change the data at all. It merely runs the checks you set up, and flags up any warning messages that result from the checks. It is up to you to decide what to do with the problematic data, and you will need to decide based on the situation.

- You may want to exclude problematic records, or
- you may prefer to correct bad data (by using a data edits script).

→ Demonstrate running a script. Don't go into detail, and explain that we will cover everything in detail later on. Just show the overall process of running a script.

→ To do so:

- Start askiatools.

- Open the file *leisure travel.qes*.
- In the **tools** menu, select **run a verification script…**.
- In the toolbar, click **open existing script…**, and open the file *verify open response length.txt*.
- In the toolbar, click **run/continue script** button to run the script.
- Point out that the results are shown in the bottom four panes as the script runs – we will look at this later in the course.

■ Notes for participants

Verification scripts allow you to check the quality of your data. You can use them to check test data, the first few records collected during fieldwork, or the final data set.

You define your verification scripts in askiatools, and run them on the data currently in the QES, either on all the interviews, or a subset of the interviews. They consist of one or more custom checks on the data, known as **asserts**; for each assert that fails, a warning message is logged, which you can view when the script has finished running. The warning messages show details of the error, and let you determine which data records have the detected problems.

Running a verification script does not change your data in any way. It merely performs the checks you set up, and flags up any failures to pass these checks. It is then up to you to decide what to do about the problematic data. You may decide to exclude bad records entirely, or change bad data. To clean up your data, you would use the techniques we covered in the previous part of this course, in session 410.1 and 410.2.

Writing a verification script

How to create a script

→ Show the PPT slide "*Verification scripts are defined in* ***askiatools***"

→ Explain:

- Verification scripts use the **askiascript** language.
- Unlike with data editing scripts, we do not define them in askiadesign as routing items in the questionnaire; instead we create them in askiatools.
- Scripts are saved in plain text format, so if you wish, you can use any text editor to create them.
  - However, we recommend using the askiatools interface, because the syntax is checked there as you create the script. In this course, we will be using askiatools to create our scripts.

→ Show the next PPT slide "*The **Assert.Check** script instruction*"

→ Explain:

- Verification scripts use the **`Assert.Check`** instruction to perform checks. For each Assert.Check in the script, a check will be performed on the data.
- If a check fails on any record, a message is logged, identifying the record, and including the message text defined in the `Assert.Check` instruction.
- When you find a problem, you would either repair the data in question, or exclude the problematic records.

→ Show the next PPT slide "*Useful syntax to use with **Assert.Check***"

→ Explain:

- In a script, we can use **`%q`** (variable name) and **`%s`** (caption) to reference the current question.
- We recommend using %q, because if it refers to the variable name, and this has the advantage that it is unique, which the caption may not be.

→ Recap some common askiascript keywords that the participants will be familiar with, and which are useful in verification scripts:

- **`If/Then/EndIf`**
- **`Goto`**
- **`Dim`**
- **`For/Next`**

→ Explain:

- We are going to create a verification script to check the responses to an open-ended question in our Leisure Travel survey, to detect records where the question hasn't been answered properly.

→ Demonstrate:

- In askiatools, open *leisure travel.qes*.
- In the tools menu, select **run a verification script…**.
- In the script window, enter the following (instead of typing it, you can also load this script in, by loading the file *verify open response length.txt*):

```
Assert.Check(len(^Q15^)>=10,"The response at Q15 is less
than 10 characters in length.")
```

- Explain briefly what we can expect this script to do.
- Explain that we will now test the script and run it (note: this is covered in the next sections of the course).

Notes for participants

Verification scripts use the askiascript language. Unlike with conventional routing instructions and data editing scripts, however, we define them in askiatools.

Note Scripts are saved in plain text format, so you can use any text editor to create them. However, we recommend using the askiatools interface, because it checks the script syntax as you work.

In verification scripts, we use one or more `Assert.Check` instructions to perform tests on the data. Each time one of these tests fails, a message will be added to the log.

The syntax to use is as follows:

`Assert.Check(`*`expression, message`*`)`, where *`expression`* is a variant and `message` is a string.

Each `Assert.check` evaluates the specified expression as a boolean value. If the result is true, nothing happens; if it is false, the specified message is logged.

For example:

```
Assert.Check(Q5A+Q5B+Q5C+Q5D+Q5E=100,
"Q5A to Q5E should add up to exactly 100")
```

In a script, you can use **`%q`** to reference the variable name of the current question and **`%s`** to reference its short caption (or if the short caption is blank, it will use the long caption instead).

We recommend using `%q`, because if is shorter, and it will always be unique, compared to the question caption.

The following askiascript keywords are not specific to verification scripts, and you may already be familiar with them from writing routing scripts, but they are very useful when writing them:

- **`If/Then/EndIf`** for creating branching in the script, depending on specific data states
- **`GoTo`** for jumping to a later point in the script, allowing you to skip sections of the script, depending on branching logic (`If/Then`) you define
- **`For/Next`** to create loops in the script, in order to perform the same or similar actions multiple times
- **`Dim`** for creating temporary variables in the script, to allow you to perform comparisons and calculations

If you need a reminder about how to use these keywords, please refer to the askiascript documentation in the askiadesign Assistant (found at http://designhelp.askia.com/home), or the course-book you were given during the askiascript course.

To create a script:

1. Open askiatools.
2. Open your QES file.
3. In the **tools** menu, select **run a verification script…**.
4. In the large text box, type your script.
5. You can then test and run your script (this is covered in the next section of this course).

Testing your script

Validating your script

Validate your script's syntax and how it matches with the data.

→ Explain:

- Before we run our script, we can have askiatools run some checks on it.
- Askiatools cannot guarantee that the script will have the effect we desire, but it can detect some errors.
- Askiatools can check:
  - the syntax;
  - that the script corresponds to the data set (the variable names in your script are valid).

→ Demonstrate:

- Open, or go back to the window already open displaying *leisure travel.qes*
- In the toolbar, click **compile script**.
  - Explain that this does not run the script; it simply checks the script for errors.
  - Point out that in this case, askiatools displays the message "no error".
- Now we will look at what happens if there are errors in our script.
  - Load the script *verify open response length with errors.txt*.

- Explain that this shows what might happen if we have errors in the script we just compiled.
- Point out that there is a mistake in the variable name (it refers to Q115, which does not exist). Point out that askiatools has detected this, and changed the colour-coding of the variable name to indicate it is not recognised.
- Point out that we also have a mistake in our script syntax. There is an extra parameter at the end of the `assert` statement.
- Compile the script again. Askiatools detects the first error in the script, and displays an error message. It also highlights the part of the script where this error was found.
- Correct the variable name to Q15 (point out that the variable name changes back to green, to show it is recognised). Explain that askiatools cannot know whether we have specified the correct question; just that we have specified a valid questions.
- Now compile the script again. Askiatools detects the other error in the script, and displays an error message ("too many parameters"). It highlights the script keyword to which this error applies; it is up to us to check our script to find the superfluous parameter.
- Remove the extra parameter ",1" from the end of the assert statement.
- Compile the script again. Askiatools reports "no error".
- Our script is ready to run (note that this is covered in the following section).

Notes for participants

Your verification script must conform to the askiascript rules of syntax. Askiatools automatically detects some errors in your script, and will highlight these in red within the script window.

You can also compile the script to have Askiatools test your script. Askiatools can test

- the *syntax* of your script, and
- confirm that the script corresponds to the data set (e.g. variable names match those in the survey).

For more details on askiascript syntax, refer to the askiadesign Assistant at http://designhelp.askia.com.

Note While this check validates the syntax of your script, it cannot guarantee that the script logic is correct, and therefore you will need to carefully check that the script is performing the checks you intend it to.

To validate your script's syntax:

1. In the toolbar, click **compile script**.
2. If there are any errors, askiatools will display an error message. Check your script for potential errors, correct any you find, and then repeat this process until no more errors are found.

Performing a test run

Testing the script.

→ Explain:

- The syntax check can find errors in our script, but it cannot guarantee the script has the effect we intend, because it cannot know what we want the script to do.
- However, we can perform a further check before we run our script.
- Askiatools allows us to perform a test run, where we step through the script and examine what the script is doing

→ Show the slide "*Step-by-step mode*"

→ Demonstrate:

- Open the script *verify script step by step.txt*.
- In the toolbar, click **run script step by step**.
- Explain that the script execution has stopped at the first element of the script. The box at the bottom left shows the keyword we are on, and the result, including the contents of any variables during this step.
- Click **run script step by step** repeatedly, and show that the script continues executing in steps, and that we can examine what is happening.
- Explain that as you continue clicking the button, the script carries on running until all of the rows have been executed.

Script testing options

→ Explain that several options are available when we are testing our script:

- Resetting the run
  - If we want to do another test run, or if we want to restart it mid-run, we need to reset the current run.
  - Clicking **reset run** takes us back to the start, clears all variables and prepares the script for a fresh run..

- Restarting the current interview
  - In the middle of a test, you can restart the current interview if you wish, by using the toolbar command **restart current interview**.
- Starting on a different data record
  - By default, the step by step run begins on the first data record. We can, if we wish, begin on a different record, by using the navigation buttons in the toolbar to specify a different record.
  - If you change to a different record, the script will restart at the first line.

Notes for participants

Before you perform the actual run, you should perform a test run to confirm that your script is checking the data in the way you intend. You can do this by stepping through the script, making corrections as needed, resetting the run, and testing again, until you are satisfied with the script.

To test your script:

1. This procedure assumes that you already have your verification script open in askiatools. If not, please see *Writing a verification script* on page 33.
2. In the toolbar, click **compile script**. As we have seen, this runs a syntax test on the script, and it also ensures that the script information panes are displayed below the script.
3. In the toolbar, click **run script step by step**. The script pauses when it reaches the first element of your script (e.g. an askiascript keyword). At this point, you can view details of the script in the panes below, such as the content of any variables during this step of the script.
4. To continue, click **run script step by step** repeatedly until the script has completed, or you are satisfied with the operation of your script.
5. To end your testing, simply stop clicking **run script step by step**.

If you want to do another test run, or simply re-start the test mid-run, you need to reset the current run. To do so, click **reset run** in the toolbar. This clears all asserts and prepares the script for a fresh run.

When testing your script, the following options are useful:

Restart current interview	In the middle of a test, you can begin the current interview again if you wish, by using the toolbar command **restart current interview**.
Interview navigation buttons	By default, your test will begin at the first record. However, you can use the navigation buttons to begin at any record you wish You can also double-click the position indicator (e.g. 10/520) to go to a specific record. Note that if you change to a different record, your script will restart at the first line.

Running the script

→ Explain:

- When we have finished testing, we can run the script.

→ Demonstrate:

- Return to the script *verify open response length.txt* (the same file as you last were last using)
- Run the script (click **run/continue script** in the toolbar).

→ Discuss the following options, and run the script again as necessary to illustrate them:

- **Run on reduced data**: if this option is selected, the script will take into consideration only your asserts, ignoring any other routing logic you have in your QES; this helps your script to run faster. However, before using this option, you should carefully consider whether your routing logic might affect the result of your asserts; if so, this option may lead to misleading results from your asserts.
- **Completes only**: useful if you are not planning to include incomplete records in your analyses, as incomplete records will be skipped (and hence the script will run more quickly).
- Running on a sub-set of the data (use the **select** option). To set up a filter, click **select…**, and then enter the appropriate askiascript code. For example:

  e.g. `^Country list^.value has {180}`

- **Speed test**: diagnoses slow scripts; explain that speed stats will appear in the results, which we will look at shortly. If this option is selected, *duration* and *percentage* columns are populated, indicating which parts of your script are taking the most time to run.

■ Notes for participants

To run your script:

1. In the toolbar, click **run/continue the script**.
2. By default, the script will run on all interviews in the QES, but you can use the options described below to refine how your script runs.

The following options are useful when running your script:

Break on assert	If selected, the script will pause when the first assert message is displayed (i.e. the first failed check). You can then continue the script, check the script or reset the run, as necessary.
Run on reduced data	If selected, the script will consider only your asserts and ignore any other routing logic in your survey. While this will help your script to run faster, it may cause your asserts to return misleading results. For this reason, you should carefully consider whether your asserts are dependent on the outcome of any routing logic before using this option.
Completes only	Select this option if you want to run your script on completed records only. This can be a good choice if you

	are not going to include incomplete interviews in your data analyses.
Select	Use this command to run your script on a selected group of respondents, instead of the entire data set. For example, you may want to specify respondents in a certain geographical region, so might enter the following script (assuming that North West is coded as 1 in the variable `Region`) `Region/value has {1}`
Speed test	This option is useful if your script is taking a long time to run, as it helps you determine which parts are taking the longest; you may then be able to amend these areas of the script to make it more efficient. If you select this option, the script results will include information on how long the script took to run (in the duration and percentage fields; see the results section on page 41 below).

Viewing the output

Viewing the results.

→ Demonstrate:

- Run the script *verify open response length.txt*, if you have not done so, and show the results.

→ Explain:

- Remember that the script has not changed any data; it simply flags up any failures to pass the tests (asserts) defined in it.
- Talk the participants through the four panes of the results view at the bottom of the screen, as follows:
  - **First pane**: displays, for each part of the script:
    - **Type:** indicates the type of item that is on this row (script keyword, question, syntax, variable, function or method).
    - **Value:** indicates more detail about the item on this row.
    - **Result:** indicates the actual value of the item after it was executed.
    - If you are using the **speed test** option, this view also displays **duration** and **percentage** columns, which are useful if your script is running slowly; they can help you determine where the slowest parts are, and remedy this by making your script more efficient.

- **Second pane**: displays details of any records that have failed your asserts.
- **Third pane**: For each line in the script, this view displays the line number, number of interviews checked, number of interviews that failed a check (assert) and the associated message for that check (assert) when relevant.
- **Fourth pane**: A console window that lets you type code and immediately see the result, for example `q1.Value + { 1 to 3}.`

Notes for participants

Once your script has finished running, you can view the results section at the bottom of the screen.

Note When executing a validation script, remember that your script has not changed any data; it has simply flagged up any failures to pass the tests that you defined.

The results section consists of four panes. They are described below from left to right:

- The **first pane** displays, for each part of the script, the following information:
  - **Type**: indicates the type of keyword, question, syntax, variable, function, method.
  - **Value**: Indicates more information within the type.
  - **Result**: indicates the actual value of the token after it was run.
  - If you selected the **speed test** option before you ran your script, you will also see information in the **duration** and **percentage** columns (you will probably need to scroll right to see them). This information is useful if your script is taking a long time to complete, as you can narrow down which lines or sections of the script are taking the longest, and amend the script to improve its efficiency.
- The **second pane** displays details of any records that have failed your asserts.
- The **third pane** displays a summary. For each line in the script, you will see the line number, number of interviews checked, number of interviews that failed an assert, and the associated message for that assert where relevant.
- The **fourth pane** is a console window, which lets you type some code and immediately see the result. For example:

```
q1.Value + { 1 to 3}.
```

Generating scripts automatically

Generating scripts automatically for questions not already referenced in the current script.

→ Explain:

- Askiatools can automatically create asserts for every question you haven't explicitly referenced in your script.
- This is a useful way to quickly create checks for each question of a specific type in your survey, for example, if you want to check that it contains the correct type of data.
- You define the form that this script takes in the program options.

→ Demonstrate:

- Setting the global options for script generation (in **tools – options… – generation of verification script**). Enter an example assert for two or three question types, as follows (if these asserts are already set up, then show these asserts to the participants):

```
Assert.Check(%q.HasValidData, "%s does not
appear to have valid data.")
```

- Now generate script automatically from the **Run a verification script** dialog, by clicking **generate script for questions not in current script** in the toolbar. Show that script lines have been generated for the specified question types.
  - Explain that if you have any questions with the same name as Askiascript keywords, this will cause an error in the automatically generated code (e.g. for the question `Month` on our survey). These can be fixed by putting the ^ character around the variable name.
  - Run the script and show the results.
  - Do not close this window as you will be using it again shortly to demonstrate how to save a script.

Notes for participants

If you want to verify a large number of questions in the same way (e.g. to perform simple checks to make sure they have data of an appropriate type), you do not have to write a script for each individual question. Instead, you can define a single assert statement for each question type (single-coded, multi-coded, open-ended etc.), and then automatically generate asserts for every question of that type.

This is accomplished by going into the program options, and defining an appropriate `assert` statement for each question type. For example, if you set up the following assert for single-type questions, then it will be automatically generated for every single-coded question in your QES:

```
Assert.Check(%q.HasValidData, "%s does not appear to have
valid data. ")
```

Note These generation settings are global, and apply to all QES files. If you need different settings on another survey, you will need to change the global settings.

To view or edit the global generation options:

1. In the **tools** menu, select **options…**.
2. Set the options in the **generation of verification script** section (for example, in **single question**, enter a script that will be generated automatically for single type questions).

Once you have specified assert statements for one or more question types, you can automatically generate the scripts at any time.

Note Asserts are not generated automatically for any questions which are already referenced in your manually defined asserts. For example, if you have referenced Q1 in an assert, then an assert will not be automatically generated for it.

To automatically generate verification scripts:

1. When you have a QES open, ensure you are in the *verification script* dialog. To open it, in the **tools** menu, select **run a verification script…**.
2. Ensure you have already defined automatic scripts in the global settings (see above).
3. In the toolbar, click **generate script for questions not in current script**.

Note if you have any questions in your QES that have the same names as Askiascript keywords (e.g. `Month`), you will need to place caret (^) characters around the question names after generating the script. For example: `^Month^`

Saving and loading scripts

Saving scripts for future use

→ Explain:

- You can save verification scripts for future use.
- Script files are saved in plain text format, with a `.txt` filename extension.
- They can be re-loaded at a future date, perhaps into a later project, modified if necessary, and then run on that project.

→ Demonstrate:

- Go back to the window you previously used when generating a verification script and with that script in view, choose the option to save it (in the toolbar, click **save script**).
- Close the verification script dialog.
- Re-open the verification script dialog (**tools – run a verification script…**) and then open the script you just saved (in the toolbar, click

open existing script…).

Notes for participants

It is possible to save scripts for future use: you can then re-use them on multiple projects, either as-is, or with changes appropriate to the new project. Script files are saved in plain text format, with a `.txt` filename extension.

To save a verification script:

1. When editing your script, in the toolbar, click **save script**.
2. The save dialog opens, allowing you to specify a location and file name for your script file.

To open a previously-saved verification script:

1. Ensure you have the verification script editor open (if not, in the **tools** menu, select **run a verification script…**).
2. In the toolbar, click **open existing script…**.
3. In the file dialog, select the script you want to open.

Some more useful keywords and objects

Using other script elements

→ Show PowerPoint slide "*Using an IF block to avoid an unwanted fail*"

→ Explain:

- As we saw earlier, we can use any Askiascript elements in our verification scripts. If we want to perform more complex checks, IF blocks are very useful. For example:

```
Dim OpenQ15 = Q15.value
If Q15.HasValidData Then
      Assert.Check(OpenQ15.length>=5,"The response
      at Q15 is less than 5 characters in length.")
EndIf
```

This script ensures that Q15 has data in it before we check the number of characters in it. In other words, if it is empty, we don't perform the check within the If block. If we didn't use this IF block, our Assert would fail when Q15 is empty.

More useful keywords

→ Explain:

- In addition to what we have already covered, there are several other keywords that are useful in verification scripts.

→ Show slide "*Some more useful keywords and options*"

→ Briefly explain each of the following keywords:

- **`Assert.ToString()`** Returns the string representation of the object/variable.
- **`GoTo`** Allows you to skip sections of your script by moving forwards and ignoring intervening code.
- **`Interview.`*`property`*** can be used to obtain information about the current interview. For example:
  - **`Interview.PanelID`** returns the Askia Panel ID when available.
  - **`Interview.Progress`** returns the percentage value of the respondent's progress through the questionnaire.
- **`LoopQuestion.AllIterations`** Returns a value that indicates the number of times a looped question exists in the questionnaire. This depends on the number of loop iterations, and whether nested loops are involved. Examples:
  - For example, if Q1 is a question inside a loop with 3 iterations then Q1.AllIterations.Count will return a value of 3.
  - If Q2 is a question inside a loop within another loop, with 3 and 2 iterations, then Q2.AllIterations.Count will return a value of 6.
- **`Question.HasNoData`** Returns *false* if there is any data in the question or *true* if the question contains no data.
- **`Question.HasValidData`** Returns *true* or *false*. It checks that there is data in the current question and if so, whether that data is valid, according to specific checks:
  - if the data is compatible with DK settings (for all question types)
  - if the data complies with the min/max number of answers and any exclusivity settings (for multi-coded questions)
  - if the data complies with the size (for open questions)
  - if the data falls within the expected range (for numeric and date questions).

→ Show slide "*Verifying loop variables*"

→ Explain the following example that combines two of the above keywords:

- Q13a is a multi-coded question inside a loop, and you want to check that it has valid data on all iterations. You might create the following script:

```
Dim i
For i = 1 to ^Q13a^.AIterations.Count
   Assert.Check(^Q13a^.Iteration[i].HasValidData ,
   "Q13a doesn't have valid data")
Next i
```

Notes for participants

In addition to the script keywords we have already covered, there are several other keywords that are useful in verification scripts:

`Assert.ToString()`	Returns the string representation of the object/variable.

`GoTo` *`label`*	Allows you to skip sections of your script.
	Example:
	`If Country has {11} Then`
	`     Assert.Check(Language has {1},"When country is 'UK', 'English' should be selected, but was "+ ^Language^.Answers[1].Caption)`
	`     GoTo Section2`
	`EndIf`
	
	`'The following section of code will be skipped over if the country code is 11`
	
	`'More code here etc`
	
	`Section2:`
	`'This code will not be skipped over and will be the first code after the goto for those with the country code 11`
	
	In this example, if the country code is 11, then the script execution skips ahead to the label `Section2`.
	Note: `GoTo` can only go forward. It is not possible to use `GoTo` to go backwards/upwards in the script.

`Interview.`*`property`*	This object can be used to obtain information about the current interview. You can use it in the following ways: • **`Interview.Broker`** returns the Broker ID. • **`Interview.BrokerPanelID`** returns the broker panel id as received in askiaweb (when available). • **`Interview.GUID`** returns the global unique identifier for each respondent. • **`Interview.IPAddress`** returns the IP address of the interview (if available). • **`Interview.Latitude`** returns the latitude for the respondent location. If this is not available, 0 is returned.

	• **`Interview.Longitude`** returns the longitude for the respondent location. If this is not available, 0 is returned. • **`Interview.PanelID`** returns the Askia Panel ID when available. • **`Interview.Progress`** returns the percentage value of the respondent's progress through the questionnaire. • **`Interview.Seed`** returns a pseudo-unique number for the interview, which is used by askiaweb to generate random numbers in the survey.
`Question``.`**`AllIterations`**	This returns a value that indicates the number of times a looped question exists in the questionnaire. This depends on the number of loop iterations, and whether nested loops are involved. For example, if Q1 is a question inside a loop with three iterations then Q1.AllIterations. Count will return a value of 3. If Q2 is a question inside a loop with three iteration within another loop with two iterations, then Q2.AllIterations.Count will return a value of 6.
`Question``.`**`HasNoData`**	This boolean (true/ false) keyword indicates if there is any data in the question or not.
`Question``.`**`HasValidData`**	Indicates if the answer of the current question is valid, for open, numeric and closed questions (depending on its type). This keyword checks if there is a value and if that value is compatible with these 4 checks: 1. the DK settings, for all question types, 2. the min/max number of answers and any exclusivity settings, for multi-coded questions, 3. size, for open questions 4. and range, for numeric and date questions.

Keyword examples

Suppose Q1 is a multi-coded question inside a loop, and you want to check that it has valid data on all iterations, you might create the following script:

```
Dim i
For i = 1 to ^Q13a^.AllIterations.Count
   Assert.Check( ^Q13a^.Iteration[i].HasValidData,
   "Q13a doesn't have valid data")
Next i
```

Suppose Q3 should be asked only when Q2=1 (and in this situation, it should always be asked). To validate this, there are two checks you would need to do. First, you would want to generate an error if the value in Q2 is 1 and there is no data in Q3. Secondly, you want to generate an error if you have data in Q3, but Q2 does not equal 1. You could structure your script as follows:

```
If Q2 has {1} Then
      Assert.Check(Q3.HasValidData,"Skip executed when it
      should not have been")
Else
      Assert.Check(Q3.HasNoData,"Expected skip not
      executed")
EndIf
```

Another way to write this is as follows:

```
Assert.Check(Q2 has {1} = Q3.HasValidData,"Skip error
found")
```

Both of our conditions should be simultaneously true or simultaneously false, and if they are equal, then the error will be generated.

Detecting speed cheats

→ Show slide "*Checking for speed cheats*"

→ Explain the how to use `Interview.Duration()` to obtain the time in seconds that the respondent took between two questions

Notes for participants

The **`Interview.Duration()`** function will return the elapsed time in seconds that the respondent took to progress between two stated questions, and can be used to find records where the interview was completed in much less time than you would expect to answer the questions properly.

```
Interview.Duration(start_question,finish_question)
```

For example:

```
Interview.Duration(Screen01,AnyComments)
```

will return, as a numeric value, the number of seconds the respondent took between answering the question `Screen01` and the question `AnyComments`.

You can use the function within an `Assert.Check` to report interviews where a the respondent appears to have taken less time than you would reasonably expect to answer all the questions in that section.

Using the same example, to check that each participant spent at least two minutes on the specified section of the questionnaire, you would write:

```
Assert.Check(Interview.Duration(
Screen01,AnyComments) >= 120,
"The respondent took less than 2 minutes to complete the
interview.")
```

Note When setting a time limit, bear in mind any other factors that could have resulted in a faster than expected interview time, such as:

- Many questions being skipped due to routing logic.
- A participant that was very familiar with the subject material.

Both of these situations could result in an interview much shorter than the typical time.

One way to set a suitable time limit is to look at the median interview length for all interviews completed and then use a fraction of that, such as 50% or less.

Removing bad records

Dealing with bad records.

→ Explain:

- Verification scripts don't fix bad data, they merely allow you to identify it. It's up to you what to do with the bad records.
- When you discover bad interview data, you have two ways to deal with it in askiatools:
  - you can fix the data (with a data edit script), or
  - you can delete the entire record (or records) affected.
- Writing a data edit script to make changes was what we covered sections 1 and 2 of this course. This allowed you to change the values in a data record, but not to delete records.
- Here, we will look at how to delete entire data records.
- Askiatools lets you delete records in several ways, but in this case, we want to delete with a script, so only the affected records are removed.

→ Demonstrate:

- **IMPORTANT:** Explain that we should always back up our data before we delete any records. **This procedure cannot be undone**.
- In askiatools, open the file *deleting records.qes*.
- Select **edit-delete a selection of interviews…**.
- Select **incompletes** and **completes**.
  - Explain that we need to do this, otherwise no records will be selected, and therefore nothing will be changed by our script (alternatively, we could select interviews by date range).
- Next to **additional filter**, click **…**.
- In the condition editor, enter the following script (and show how to use the question list at the bottom to quickly add questions and responses to the script):

```
brandlist.Value has {1}
AND q1.Iteration(1).HasNoData
```

- Click **compile** to confirm your script syntax is correct.
    - Explain that this does not guarantee our script will select the records we want; we still need to provide the correct script logic.
- Run the script.

Notes for participants

When you have identified bad interview data, there are two ways to deal with it in askiatools. Verification scripts do not change your data, they merely check it. It is up to you to decide how to deal with any problematic data that they find. You can either:

- change the individual data items (e.g. set them to blank or DK) or;
- delete the entire data records containing problematic data.

In the data edits course, we cover how to change data. In this section, we will explain how to delete entire records in askiatools.

Record deletion is permanent, so be sure to always back up your data before proceeding.

To remove a bad record in askiatools:

1. **IMPORTANT:** Before proceeding, back up your data (for example, you could use Windows explorer to make a copy of the QES file).
2. In askiatools, open your QES file.
3. In the **edit** menu, select **delete a selection of interviews…**.
4. Ensure **incompletes** and **completes** are both selected, as we want to delete both incomplete and complete interviews.

   Note that if you don't select **Incompletes**, **completes**, or provide a date range, then no interviews will be affected by the script.
5. Next to **additional filter**, click **…**. The **condition editor** opens.
6. Enter the script that will identify the records you want to delete. For example, if you wanted to delete records where the respondent selected 2 at Q1 and 4 at Q15, you would enter:

   ```
   Q1.Value has {2} And Q15.Value has {4}
   ```

   To help you enter the correct question names and response numbers, you can select questions and responses from the lists at the bottom of the condition editor. Double-click a question name or response to add it to the script.
7. To test your script's syntax, click **compile…**. This checks whether your script conforms to correct askiascript syntax. Note that in no way does this indicate the script will have the result you intend; check your script's logic carefully to ensure that the correct records will be selected.
8. Click **OK** to close the condition editor.

9. Click **OK** to run your script. If any matching records are found, you will be asked to confirm that you want to delete them.

Recap

→ Show slide "*Summary: Verification Scripts*" and provide a one-minute summary of the topics covered.

In this session, you have learned how to:

- Write verification scripts using the `Assert.Check()` command
- Run your verification script in **askiatools**
    - Including different ways to run it on a subset or in step-by-step mode
- View the verification script output in **askiatools**
- Use syntax that is preforms common verification scripts
    - `Interview` properties,
    - loop properties `.Iteration` and `.AllIterations`,
    - question properties `.HasNoData`, and `.HasValidData`
- Generate simple verification scripts automatically

→ Point out to participants that there is some information in their notes (under the heading *Further reading*, on how you can automate processes by running a verification script in Batch mode. Then prepare participants for the exercise.

Further reading

If you need to perform the same validation on data you are collecting on a regular basis, you can simplify the task and automate the process by using the command line interface to askiatools. This allows you to execute your validation script, or any other askia script run by tools, automatically from a batch file.

The syntax for the command line is as follows:

```
Tools.exe "X:\folder\subfolder[… etc]\project.qes"
/runverification /completes
/definitionfile:"X:\folder\subfolder[…]\script.txt"
/logfile:"X:\folder\subfolder[… etc]output.txt
```

For example, to run a verification in batch mode using two examples files covered earlier in this session: *verify open response length.txt,* containing the script and *coding open question example.qes* containing the survey, the command would look like this:

```
Tools.exe "C:\projects\test\course\coding open question
example.qes" /runverification /completes
/definitionfile:" C:\projects\test\course\verify open
response length.txt" /logfile:"
C:\projects\test\course\verification_log.txt
```

This also assumes all the files are located in the folder. It will write the output to the file verification_log. However, the output file, as with the other files, can be any name of your choosing.

Batch files are especially useful when you have several different commands to execute on a regular basis, one after another.

This knowledge base article provides more information and a further example of how to set up a Windows batch file to execute a validation script. Either follow this link or search the knowledge base for "command line in askiatools" then look for example #26, or follow this link:

https://support.askia.com/hc/en-us/articles/200003451-Command-line-in-Askiatools#example26

Exercise

→ Ask participants to complete this exercise. Provide participants with the file *Mobile Internet survey - exercise 3.qes*. There are model answers provided in the script files *Verification script model answers.txt* and *Verification script model answers 2.txt*, which will be useful if the participants need help with the scripts in this exercise.

Checking data quality

For this exercise, your tutor will provide a QES file to use. You will be writing a verification script to check several aspects of the data. You will be checking for errors and interview cheats (people who have rushed through, not properly answering the questions), and checking you have valid data in your other questions.

If you need help with any of the scripting, your tutor can provide a set of model answers.

Please follow these steps:

1. Open the QES file in askiatools, and then open the verification script editor (in the **tools** menu, select **run a verification script…**.
2. Write an `Assert.Check` statement to catch records where there is data in Q9_OTHER, but it is very short (fewer than 3 characters).

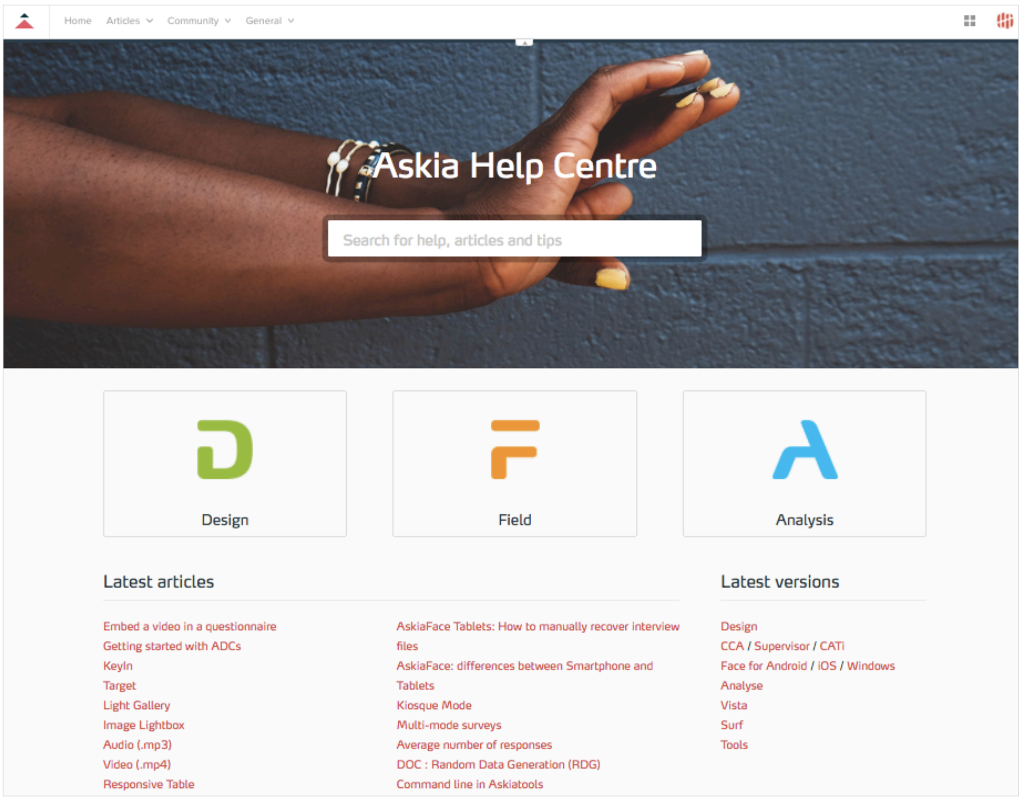
   For a hint, refer to *Writing a verification script* on page 33. In order to check there is data in Q9_OTHER, you will need to use an If/EndIf block around your `Assert` statement.
3. Next, write a script to check whether a respondent has selected response 1 at all of the single-coded questions (Q1, Q2, Q9, Q10 and Q11), which may indicate that the respondent was rushing through the survey without paying attention to the questions.
4. Add to this script an `Assert.Check` statement to look for participants that do not seem to have spent enough time on the survey because the time they took, between Q1 and Q17, was less than 3 minutes.
5. Save your script to the computer's desktop.
6. Close the script window, open the program options (tools-options), and in the verification settings, enter the following in single question and multiple question:

```
Assert.Check(%q.HasValidData, "%q does not appear to have
valid data.")
```

7. Re-open the script editor and load your script.
8. Generate validation script for any questions you haven't already mentioned in your script (click **generate script for questions not in current script**).
9. Compile your script to test it for syntax errors. If any errors are found, correct them and then keep compiling until your script is free from syntax errors.
10. Run your script, and then view the results.
11. In the toolbar, click **new script**.
12. Write two `Assert.Check` statements to catch records the value in Q8 and Q17 are greater than 100 (this means where the respondent said they have installed more than 100 apps on their device).
13. Re-compile your script to test it for syntax errors.
14. Test and run your script.
15. Optionally, if time permits, try to execute your verification script from the Windows command line.

Afterword

Though you have reached the end of this training course, there is much more to learn about scripting and veritfication. You will find more information online at **askia.com**.
If you have not done so already, sign up for access to our extensive **askia support** site at support.askia.com. This is full of useful resources for beginners and experienced **askia** users alike. You can register by clicking **sign in** at the top right, and then clicking **sign up**.



Latest articles

Embed a video in a questionnaire
Getting started with ADCs
KeyIn
Target
Light Gallery
Image Lightbox
Audio (.mp3)
Video (.mp4)
Responsive Table
AskiaFace Tablets: How to manually recover interview files
AskiaFace: differences between Smartphone and Tablets
Kiosque Mode
Multi-mode surveys
Average number of responses
DOC : Random Data Generation (RDG)
Command line in Askiatools

Latest versions

Design
CCA / Supervisor / CATi
Face for Android / iOS / Windows
Analyse
Vista
Surf
Tools

At the support site, you can also access all of the current Askia documentation, in particular:

- The Askia Analysis Assistant (complete software documentation).
- A searchable database of articles containing specific examples of editing and verifying data, as well as many worked examples using the software to solve different problems.

These will help you to learn about the many other features of Askia. It is worth visiting the site regularly, as new articles are added all the time.