What is an XLSForm?

XLSForm is a form standard created to help simplify the authoring of forms in Excel. Authoring is done in a human readable format using a familiar tool that almost everyone knows - Excel. XLSForms provide a practical standard for sharing and collaborating on authoring forms. They are simple to get started with but allow for the authoring of complex forms by someone familiar with the syntax described below.

The XLSForm is then converted to an ODK XForm, a popular open form standard, that allows you to author a form with complex functionality, like skip logic, in a consistent way across a number of web and mobile data collection platforms.

Basic format

Each Excel workbook usually has two worksheets: **survey** and **choices**. A third optional worksheet, called **settings**, can add additional specifications to your form and is described below.

The survey worksheet

This worksheet gives your form its overall structure and contains most of the content of the form. It contains the full list of questions and information about how they should appear in the form. Each row usually represents one question; however, there are certain other features described below that you can add to the form to improve the user experience.

The choices worksheet

This worksheet is used to specify the answer choices for multiple choice questions. Each row represents an answer choice. Answer choices with the same **list name** are considered part of a related set of choices and will appear together for a question. This also allows a set of choices to be reused for multiple questions (for example, yes/no questions).

Setting up your worksheets

Both of these worksheets have a set of mandatory columns that must be present for the form to work. Additionally, each worksheet has a set of optional columns that allow further control over the behavior of each entry in the form, but are not essential to have. Every entry must have values for each of the mandatory columns, but the optional columns may be left blank.

- The **survey** worksheet has 3 mandatory columns: **type**, **name**, and **label**.
  - The **type** column specifies the type of entry you are expecting for the question.
  - The **name** column specifies the unique variable name for that entry. No two entries can have the same name. Names have to start with a letter or an underscore. Names can only contain letters, digits, hyphens, underscores, and periods. Names are case-sensitive.
  - The **label** column contains the actual text you see for the question in the form. Alternatively, label translation columns can be used.

type	name	label
today	today	
select_one gender	gender	Respondent's gender?
integer	age	Respondent's age?

◄ ► **survey** | choices | settings ⊕

- The **choices** worksheet has 3 mandatory columns as well: **list name**, **name**,

and **label**.
  - The **list name** column lets you group together a set of related answer choices, i.e., answer choices that should appear together under a question.
  - The **name** column specifies the unique variable name for that answer choice.
  - The **label** column shows the answer choice exactly as you want it to appear on the form. Alternatively, label translation columns can be used.

list_name	name	label
gender	transgender	Transgender
gender	female	Female
gender	male	Male
gender	other	Other
◄► | survey | **choices** | settings | ⊕		

The columns you add to your Excel workbook, whether they are mandatory or optional, may appear in any order. Optional columns may be left out completely. Rows or columns may be left blank to aid readability, but data after 20 adjacent blank columns or rows on a sheet will not be processed. All .xlsx file formatting is ignored, so you can use dividing lines, shading, and other font formatting to make the form more readable.

One thing to keep in mind when authoring forms in Excel is that the syntax you use must be precise. For example, if you write **Choices** or **choice** instead of **choices**, the form won't work.

Question types

XLSForm supports a number of question types. These are just some of the options you can enter in the **type** column in the **survey** worksheet in your XLSForm:

Question type	Answer input
integer	Integer (i.e., whole number) input.
decimal	Decimal input.
range	Range input (including rating)
text	Free text response.
select_one [options]	Multiple choice question; only one answer can be selected.
select_multiple [options]	Multiple choice question; multiple answers can be selected.
select_one_from_file [file]	Multiple choice from file; only one answer can be selected.
select_multiple_from_file [file]	Multiple choice from file; multiple answers can be selected.
rank [options]	Rank question; order a list.
note	Display a note on the screen, takes no input. Shorthand for type=text with readonly=true.
geopoint	Collect a single GPS coordinate.
geotrace	Record a line of two or more GPS coordinates.
geoshape	Record a polygon of multiple GPS coordinates; the last point is the same as the first point.
date	Date input.
time	Time input.
dateTime	Accepts a date and a time input.
image	Take a picture or upload an image file.
audio	Take an audio recording or upload an audio file.
background-audio	Audio is recorded in the background while filling the form.
video	Take a video recording or upload a video file.
file	Generic file input (txt, pdf, xls, xlsx, doc, docx, rtf, zip)
barcode	Scan a barcode, requires the barcode scanner app to be installed.
calculate	Perform a calculation; see the Calculation section below.
acknowledge	Acknowledge prompt that sets value to "OK" if selected.
hidden	A field with no associated UI element which can be used to store a constant
xml-external	Adds a reference to an external XML data file

GPS

For example, to collect the name and GPS coordinates of a store, you would write the following:

type	name	label
text	store_name	What is the name of this store?
geopoint	store_gps	Collect the GPS coordinates of this store.
◄ ►	**survey**	choices | settings ⊕

To collect a line or shape of GPS coordinates, you can use one of the following:

type	name	label	hint
geotrace	pipe	Pipeline	Please walk along the pipeline and record the coordinates of each corner point
geoshape	border	Border	Please walk along the border and record the coordinates of each corner point
◄ ►	**survey**	choices | settings ⊕	

See the question_types XLSForm for a look at each question type being used in a form.

GPS with accuracyThreshold

When recording GPS coordinates in ODK Collect, ODK collect automatically collects the gps when an accuracy level of 5 meters or less is reached. You can change this default behaviour by specifying an **accuracyThreshold**; this could be less than 5m or more than 5m. You will need to add a column with heading **body::accuracyThreshold** on the survey sheet of your XLSForm. Then specify your preferred accuracy threshold value for this column on your geopoint question, as in the example shown below:

type	name	label	body::accuracyThreshold
geopoint	store_gps	Collect the GPS coordinates of this store.	1.5
◄ ►	**survey**	choices | settings ⊕	

See gps_accuracy_threshold form for an example that uses this attribute.

Multiple choice

XLSForm supports both **select_one** (select only one answer) and **select_multiple** (select multiple answers) questions. Writing a multiple choice question requires adding a **choices** worksheet to your Excel workbook. Here is an example of a **select_one** question:

type	name	label
select_one yes_no	likes_pizza	Do you like pizza?
◄ ►	**survey**	choices | settings ⊕

list name	name	label
yes_no	yes	Yes
yes_no	no	No
◄ ►	survey	**choices** | settings ⊕

Note that the **yes_no** in the **survey** worksheet must match the **yes_no** in the **list name** column in the **choices** worksheet. This ensures that the form displays the correct list of answer choices for a particular question.

We can also add multiple choice questions that allow multiple answers to be selected, like so:

type	name	label
select_multiple pizza_toppings	favorite_toppings	What are your favorite pizza toppings?

◄ ► **survey** | choices | settings ⊕

list name	name	label
pizza_toppings	cheese	Cheese
pizza_toppings	pepperoni	Pepperoni
pizza_toppings	sausage	Sausage

◄ ► | survey | **choices** | settings ⊕

Choice names

The `name` column of the choices sheet defines the values that will be saved when each choice is selected during data collection. Choice names for **select_multiple** must not contain spaces because spaces are used as a separator when an answer with multiple selected choices is saved. Choice names for **select_one** questions may contain spaces. However, we recommend avoiding them to make analysis easier. Additionally, this makes it possible to convert the question to a **select_multiple** in a future form version.

In general, choice names should be unique within a single choice list. If two choices from the same list have the same name, they will be impossible to tell apart in analysis. If you have duplicate choice names, you will get an error, and your form will not be converted. However, it may sometimes be appropriate to have duplicate choice names. An example would be if you use a cascading select, and the choices with the same name are differentiated by a preceding question. If you do need to use duplicate choice names, you can suppress the error by using the `allow_choice_duplicates` setting:

allow_choice_duplicates
yes

◄ ► | survey | choices | **settings** ⊕

Specify other

For multiple choice questions, surveys often include an option of marking **other** when their answer choice is not listed. Then, they are usually asked to specify the other option. This is possible through XLSForm by including **or_other** after the answer choice list name in the survey worksheet. The choices worksheet stays the same. See below:

type	name	label
select_multiple pizza_toppings or_other	favorite_topping	What are your favorite pizza toppings?

◄ ► **survey** | choices | settings ⊕

list name	name	label
list name	name	label
pizza_toppings	cheese	Cheese
pizza_toppings	pepperoni	Pepperoni
pizza_toppings	sausage	Sausage

◄ ► | survey | **choices** | settings ⊕

Click on the link to look at the complete pizza_questionnaire.

Caveat
When you export data using this **or_other** option, in the **favorite_topping** column, you will see a value **other**. A separate column will have the answer for the questions in which the user selected **other**. This makes data analysis more cumbersome, so we do not recommend the **or_other** construct for large scale data collection efforts. See the **Relevant** section below for an alternative method more appropriate for large scale projects.

Location widget

A user may want to select a location from a map view during data collection. To enable this feature, you need to add the **map** or **quick map** appearance attribute to a **select_one** question. The choices sheet will also need a **geometry** column added for the list_name noted in the select_one questions. The geometry must be specified using the ODK format. This feature is only currently available on ODK Collect. See below:

list name	name	label	geometry
list name	name	label	geometry
site	shofco	Shofco	36.7965483 -1.3182517 0 0
site	gemkam	Gemkam Medical Clinic	36.7967088 -1.3170903 0 0
site	silanga	Silanga Pharmacy	36.7955008 -1.3167834 0 0
site	undugu	Undugu Medical Clinic	36.7990986 -1.3179328 0 0
◄ ► | survey	**choices**	settings	⊕		

Multiple choice from file

The options in a multiple choice question can also be taken from a separate file instead of the choices sheet. This is particularly useful if the options are dynamic or if the list of options is used in multiple surveys. Three types of files are supported: CSV, XML, and GeoJSON files. See usage examples below:

| type | name | label | choice_filter | | ——————————— | —- | ——————————— | ——— | | select_multiple_from_file country.csv | liv | In which countries did you live? | | | select_one_from_file countries.xml | cou | In which country do you live now? | | | select_one_from_file countries.xml | cit | What is the closest city? | name=${cou} | | select_one_from_file households.csv | hh | Select household number | | |
========== | ==== |
==========|==========| |
survey | | | | Caption: CSV and XML usage

| type | name | label | appearance | | ——————————— | —- | ——————————- | ——— | | select_one_from_file health_facility.geojson | site | Select the health facility visited | map | |
========== |
==== |
==========|==========|
| survey | | | | Caption: GeoJSON usage

The files require a specific format. A CSV file requires a `name` and `label` column which represent the value and label of the options. An XML file requires a structure as shown below:

```
<root>
  <item>
    <name/>
    <label/>
    ...
  </item>
</root>
```

A GeoJSON requires each feature, or point, to have an id and title property, or an attribute of the point. The GeoJSON must be defined by a single top-level FeatureCollection, and it currently only works for point geometry, as noted in detail on the ODK documentation site.

CSV, XML, and GeoJSON files may have additional columns, XML nodes, or features and custom properties as long as the above-mentioned basic requirements are met.

If the CSV, XML, or GeoJSON files use different names for the choice `name` and `label`, add a column to the survey sheet named `parameters`, and specify the custom names with the `value` and `label` parameters. See usage examples below:

| type | name | label | parameters | | ——————————— | —- |

——————————— | ——————— | | select_multiple_from_file country.csv | liv | In which countries did you live? | value=ccode | | select_one_from_file countries.xml | cou | In which country do you live now? | label=cname | | select_one_from_file households.csv | hh | Select household number | value=housenum, label=housename | |
========== | ==== |
==========|==========|
| survey | | | | Caption: CSV and XML usage

| type | name | label | appearance | parameters | | ——————————— | —- | ——————————- | ——— | ————————— | | select_one_from_file health_facility.geojson | site | Select the health facility visited | map | value = id, label = name | |
========== |
==== |
==========|==========|
========== | | survey | | | | |
Caption: GeoJSON usage

Note that, this question type is generally the preferred way of building select questions from external data as it is the most versatile and works across applications. However, if your external data file consists of many thousands of lines, please test carefully whether the performance is satisfactory on the lowest spec device you intend to use. If it is too slow, consider using External Selects or Dynamic selects from preloaded data if your data collection application supports it.

Rank

The rank widget can be used to let respondents order a list of options. The answer is saved as an ordered, space-separated list of option values where all options are always included. The syntax is very similar to multiple-choice questions.

type	name	label
rank pizza_toppings	toppings	Order pizza toppings with your favorite on top
◄ ► **survey** choices settings ⊕		

list name	name	label
pizza_toppings	cheese	Cheese
pizza_toppings	pepperoni	Pepperoni
pizza_toppings	sausage	Sausage
◄ ► survey **choices** settings ⊕		

To prevent bias it is often recommended to use the randomize feature in conjunction with this widget.

Range

To restrict integer or decimal inputs to a specific range, you can use the **range** question. This question can be used with 3 optional space-separated parameters: **start**, **end**, and **step** in a **parameters** column. The default values are 0, 10, and 1 respectively. The example below will create a question that allows input from 0 until 17 with a step of 1. Using a decimal step will result in decimal values being collected.

type	name	label	parameters
range	amount	What is the age of the child?	start=0 end=17 step=1
◄ ► **survey** choices settings ⊕			

To display a range question as a **rating widget** using stars, you can add the rating appearance as shown below:

type	name	label	appearance	parameters
range	rated	What rating do you give?	rating	start=1 end=5 step=1

◄ ► **survey** | choices | settings | ⊕

Image

To upload an image file the **image** question type can be used. To ensure the images are not too large, you can optionally set the **max-pixels** parameter which will automatically downsize the uploaded image to match the largest side of the image with the pixel value provided.

type	name	label	parameters
image	img	Upload an image	max-pixels=1000

◄ ► **survey** | choices | settings | ⊕

Audio recording quality

Certain clients use a value for **quality** in the **parameters** column to configure audio recording quality for question types **audio** or **background-audio**. Both question types accept **quality** values `voice-only`, `low` and `normal`. **audio** additionally accepts a **quality** of `external` to specify that an external application should be used for recording.

type	name	parameters
audio	animal_sound	quality=normal

◄ ► **survey** | choices | settings | ⊕

Metadata

XLSForm has a number of data type options available for meta data collection:

Metadata type	Meaning
start	Start date and time of the survey.
end	End date and time of the survey.
today	Day of the survey.
deviceid	Unique client identifier. Can be user-reset.
phonenumber	Phone number (if available).
username	Username configured (if available).
email	Email address configured (if available).
audit	Log enumerator behavior during data entry

Note that some metadata fields only apply for mobile phone-based forms.

For example, if you wanted to collect all of these types of metadata, put the following in your form (typically at the beginning, but can be at any point of your form):

type	name	label	parameters
start	start		
end	end		
today	today		
deviceid	deviceid		
phonenumber	phonenumber		
username	username		
email	email		
audit	audit		[optional, see below]

◄ ► **survey** | choices | settings | ⊕

Notice that there are no labels associated with the metadata question types. This is because the phone captures these variables automatically. These questions will not appear on the screen of the phone, but you will see them when viewing your submitted survey data. The Tutorial XLSForm shows how metadata is used in a form.

Audit enumerator behavior and location tracking

Note: For now this feature is only available in Collect, but not in Enketo webforms.

The **audit** metaquestion will enable ODK Collect to log how people navigate through a form during data entry. For example, this can be used to measure how much time an enumerator took to fill in a question, responses that were edited later on, or when the form was saved.

Optionally, the **audit** metaquestion can be configured to also record the location of the enumerator throughout the interview. This may be useful for quality control or to record exact paths taken between each respondents. To do this, add a column called **parameters** to your form and enter three required parameters: **location-priority**, **location-min-interval**, and **location-max-age**.

This example below would collect the precise GPS location every 180 seconds and will discard coordinates collected more than 300 seconds ago.

type	name	label	parameters
audit	audit		location-priority=high-accuracy location-min-interval=180 location-max-age=300

◄► **survey** | choices | settings | ⊕

See this page in the ODK Collect documentation for full details about the **audit** metaquestion, available location tracking parameters, and the format of the **audit.csv** log file created for each submission.

External XML data

For advanced users, who need to perform complex queries on external data without restrictions, an external XML data file can be added with question type **xml-external**. The value in the **name** column can be used to refer to this data in any formula (e.g. for a calculation, constraint, relevant, or choice_filter) using the **instance('name')** function. A file with the same name and the **.xml** extension should be uploaded with the form. See below for an example that requires uploading a file called houses.xml with the form.

If your external data file consists of many thousands of lines, please test carefully whether the performance is satisfactory on the lowest spec device you intend to use. If it is too slow, consider using External Selects instead if your data collection application supports this.

type	name	label	calculation
xml-external	houses		
integer	rooms	How many rooms?	
calculate	count		count(instance('houses')/house[rooms = current()/../rooms])

◄► **survey** | choices | settings | ⊕

Hints

Regular hints

Sometimes you want to add a small hint to a question on your form, instructing the user how to answer the question, but you don't want the hint to be part of the question itself. It's easy to add hints to questions in XLSForms. Simply add a **hint** column and add your hint message. See below for an example.

type	name	label	hint
text	name	What is the name of this store?	Look on the signboard if the store has a signboard.
geopoint	geopoint	Collect the GPS coordinates of this store.	

◄ ► **survey** | choices | settings ⊕

The Tutorial XLSForm provides more examples of questions with hints.

Guidance hints

There is a special kind of hint that is normally not shown in the form. It is only shown in special views. An example would to show these hints on print-outs or during a training for enumerators. These hints are called *guidance hints* and can be added in the **guidance_hint** column. See below for an example.

type	name	label	guidance_hint	relevant
integer	age	Age?		
text	name	Name?	This will only be shown for age > 18.	${age} > 18

◄ ► **survey** | choices | settings ⊕

Formulas

Formulas are used in the constraint, relevant, calculation, and trigger columns and optionally also in the default, and required columns. Formulas allow you to add additional functionality and data quality measures to your forms.

Formulas are composed of functions and operators (+,*,div,etc.). A well-documented full list of operators and functions can be found in the ODK documentation. For the technically inclined, the underlying XForms specification is the actual source document for the supported functions.

Constraints

One way to ensure data quality is to add constraints to the data fields in your form. For example, when asking for a person's age, you want to avoid impossible answers, like -22 or 200. Adding data constraints in your form is easy to do. You simply add a new column, called **constraint**, and type in the formula specifying the limits on the answer. In the example below, the answer for the person's age must be less than or equal to 150. Note how the `.` in the formula refers back to the question variable.

type	name	label	constraint
integer	age	How old are you?	. <= 150

◄ ► **survey** | choices | settings ⊕

In this example, the formula `. <= 150` is saying that the value entered `.` for the question must be less than or equal to 150. If the user puts 151 or above as the answer, s/he will not be allowed to move on to the next question or submit the form.

Other useful expressions to use in the **constraint** column can be found here. Look under the **Operators** section.

Constraint message

If you want to include a message with your constraint, telling the user why the answer is not accepted, you can add a **constraint_message** column to your form. See the example below.

type	name	label	constraint	constraint_message
integer	respondent_age	Respondent's age	. >=18	Respondent must be 18 or older to complete the survey.

◄ ► **survey** | choices | settings ⊕

In this example, if the user enters an age less than 18, then the error message in the **constraint_message** column appears. More examples on constraints have

been illustrated in this XLSForm.

Relevant

One great feature of XLSForm is the ability to skip a question or make an additional question appear based on the response to a previous question. Below is an example of how to do this by adding a **relevant** column for a **select_one** question, using our pizza topping example from before:

type	name	label	relevant
select_one yes_no	likes_pizza	Do you like pizza?	
select_multiple pizza_toppings or_other	favorite_topping	Favorite toppings	${likes_pizza} = 'yes'
◄ ► **survey** choices settings ⊕			

In this example, the respondent is asked, “Do you like pizza?” If the answer is **yes**, then the pizza topping question appears below. Note the ${ } around the variable **likes_pizza**. These are required in order for the form to reference the variable from the previous question.

In the next example, below, we use relevant syntax for a **select_multiple** question, which is slightly different from the **select_one** question example above.

type	name	label	relevant
select_one yes_no	likes_pizza	Do you like pizza?	
select_multiple pizza_toppings or_other	favorite_topping	Favorite toppings	${likes_pizza} = 'yes'
text	favorite_cheese	What is your favorite type of cheese?	selected(${favorite_topping}, 'cheese')
◄ ► **survey** choices settings ⊕			

list name	name	label
pizza_toppings	cheese	Cheese
pizza_toppings	pepperoni	Pepperoni
pizza_toppings	sausage	Sausage
◄ ► survey **choices** settings ⊕		

Since the pizza topping question allows multiple responses, we have to use the `selected(${favorite_topping}, 'cheese')` expression, because we want the cheese question to appear every time the user selects **cheese** as one of the answers (regardless of whether additional answers are selected).

Earlier we mentioned there was an alternative method for specifying other for multiple choice questions which is more appropriate for large scale surveys. This can be done using the same relevant syntax from the example above:

type	name	label	relevant
select_multiple pizza_toppings	favorite_toppings	What are your favorite pizza toppings?	
text	favorite_toppings_other	Specify other:	selected(${favorite_toppings}, 'other')
◄ ► **survey** choices settings ⊕			

list name	name	label
pizza_toppings	cheese	Cheese
pizza_toppings	pepperoni	Pepperoni

pizza_toppings	sausage	Sausage
pizza_toppings	other	Other

◄ ► | survey | **choices** | settings | ⊕

Note that you must include **other** as an answer choice in the **choices** worksheet.

Calculation

Your survey can perform calculations using the values of preceding questions. In most cases using a **calculate** type question is appropriate. For example, in the survey below, we have calculated the tip for a meal and displayed it to the user:

type	name	label	calculation
decimal	amount	What was the price of the meal?	
calculate	tip		${amount} * 0.18
note	display	18% tip for your meal is: ${tip}	

◄ ► | **survey** | choices | settings | ⊕

Note that the **${tip}** in the last line will be replaced with the actual tip amount when viewing and filling out the form.

The calculate type calculates **text** but calculations can also be added to any other question types. Non-text types can be useful for data analysis, e.g if a date or date-time is calculated. **If no label and no hint is included, the calculation will be hidden.** See example below which is the equivalent of the previous form:

type	name	label	hint	calculation
decimal	amount	What was the price of the meal?		
text	tip			${amount} * 0.18
note	display	18% tip for your meal is: ${tip}		

◄ ► | **survey** | choices | settings | ⊕

And this is an example when a non-text type is needed because of data analysis requirements:

type	name	label	hint	calculation
date	day			today()

◄ ► | **survey** | choices | settings | ⊕

Note that using non-text calculation types has no effect on using the calculation result within the form itself. This is a common misunderstanding.

If a label or hint is included, the question will be visible on the form and the calculated value will be shown in the input field or widget. This is generally only recommended for **readonly** questions to avoid re-calculating (erasing) a user-entered value. See example below:

type	name	label	readonly	calculation
decimal	amount	What was the price of the meal?		
note	display	18% tip for your meal is:		${amount} * 0.18
date	today	Today's date is:	true	today()

◄ ► | **survey** | choices | settings | ⊕

Note the difference with the first form in this section is how the calculated tip value is displayed. In the first example it was shown in the label and in the last example it is shown inside a readonly input field.

Trigger

A trigger column can be used to run a calculation only **when another visible question in the form changes**. This means that the question that is serving as the trigger **has to have a label or a hint** (otherwise it will be hidden). See a simple but very useful example below:

type	name	label	calculation	trigger
integer	temp	Enter the current temperature		
dateTime	temp_ts		now()	${temp}
◄ ►	**survey**	choices	settings	⊕

This will calculate a timestamp immediately after a respondent enters a temperature. If the user goes back and changes the temperate, the timestamp will be re-calculated.

All the regular calculation features apply to these special value-change-triggered calculations as well. So you can e.g. use a label or hint to display the calculation question on the form to the user.

Multiple questions may have the same trigger. See this example, where two calculations are triggered by the temperature question (one is hidden, and the other is shown):

type	name	label	calculation	trigger	readonly
integer	temp	Enter temperature in Celsius			
dateTime	temp_ts		now()	${temp}	
text	temp_F	Temperature in Fahrenheit	32 + 1.8 * ${temp}	${temp}	true
calculate	temp_K		273.15 + ${temp}	${temp}	
◄ ►	**survey**	choices	settings	⊕	

In the form above the temp_F question is shown to the user and the temp_K question is hidden, just as they would be if trigger was not used.

An important and powerful difference with regular calculations is that **the calculation value with a trigger may also be empty**, which serves to clear a value from the form. See example below:

type	name	label	calculation	trigger
text	name	What is the name of the oldest person here?		
integer	age	How old is this person?		${name}
◄ ►	**survey**	choices	settings	⊕

If the respondent using this form has entered the name and age of person A and subsequently finds out there is an older person B, the age field will be cleared as soon as the name of person B has been entered.

Required

It's simple to mark certain questions as required in your form. Marking them as required means the user will not be able to move on to the next question or submit the form without entering an answer for that question.

To make questions required, add a **required** column to your survey worksheet. Under that column, mark questions as required by writing **yes**. See the example below:

type	name	label	constraint	required
integer	age	How old are you?	. <= 150	yes
◄ ►	**survey**	choices	settings	⊕

Required message

If you want to customize the message displayed to users when they leave a required question blank, you can add a **required_message** column to your form. See the example below.

type	name	label	required	required_message
integer	respondent_age	Respondent's age	yes	Sorry, this answer is required.

◄ ► **survey** | choices | settings ⊕

Randomize Choices

For any question type that shows a **list of choices** the shown order of the choices displayed to the user can be randomized with the **parameters** column. See below:

type	parameters	name	label
select_one toppings	randomize=true	top	Favorite?

◄ ► **survey** | choices | settings ⊕

For reproducible randomization, a **seed** can be explicitly provided as shown below. To learn more about the randomization algorithm used, see here.

type	parameters	name	label	calculation
calculate		sd		once(decimal-date-time(now()))
select_one toppings	randomize=true, seed=${sd}	top	Favorite?	

◄ ► **survey** | choices | settings ⊕

Note that `once()` is used to prevent re-randomizing for example when a draft record is loaded for editing.

Grouping questions

To create a group of questions in your form try the following:

type	name	label
begin group	respondent	Respondent
text	name	Enter the respondent's name
text	position	Enter the respondent's position within the school.
end group		

◄ ► **survey** | choices | settings ⊕

This is a good way to group related questions for data export and analysis. Notice how **end group** doesn't require a name or label, because it is hidden in the form.

Nesting groups within groups

Groups of questions can be nested within one another:

type	name	label
begin group	hospital	Hospital
text	name	What is the name of this hospital?
begin group	hiv_medication	HIV Medication
select_one yes_no	have_hiv_medication	Does this hospital have HIV medication?
end group		
end group		

◄ ► **survey** | choices | settings ⊕

You always have to end the most recent group that was created first. For instance, the first **end group** you see closes the HIV medication group, and the second one closes the beginning hospital group. When working with groups and you keep getting error messages when trying to upload your form, double-check that for each **begin group** you have one **end group**.

Skipping

One neat feature of XLSForm is the ability to skip a group of questions by combining the group feature with relevant syntax. If you want to skip a group of questions all at once, put the relevant attribute at the beginning of a group like follows:

type	name	label	relevant
integer	age	How old are you?	
begin group	child	Child	${age} <= 5
integer	muac	Record this child's mid-upper arm circumference.	
select_one yes_no	mrdt	Is the child's rapid diagnostic test positive?	
end group			

◄ ► **survey** | choices | settings | ⊕

In this example, the two child group questions (**muac** and **mrdt**) will only appear if the child's **age** from the first question is less than or equal to five.

Repeats

A user can repeat questions by using the **begin repeat** and **end repeat** construct:

type	name	label
begin repeat	child_repeat	
text	name	Child's name
decimal	birthweight	Child's birthweight
select_one male_female	sex	Child's sex
end repeat		

◄ ► **survey** | choices | settings | ⊕

list name	name	label
male_female	male	Male
male_female	female	Female

◄ ► | survey | **choices** | settings | ⊕

In this example, the **name**, **birthweight**, and **sex** fields are grouped together in a repeat, and the user can collect the same information about multiple children by selecting the option in the form to add another repeat.

The **label** column is optional for **begin repeat**. Assigning a label to a repeat will add the label as a title to the block of repeat questions in the form.

When a repeat is shown in a table of contents, the label used to represent each repeat is the label of the first group inside that repeat. In the example below, if a repeat is filled out with values `Preity` for `first_name`, `Zinta` for `last_name` and 71 for age, that repeat will be summarized as "Preity Zinta - 71":

type	name	label
begin repeat	person_repeat	
begin group	person	${first_name} ${last_name} - ${age}
text	first_name	First name
text	last_name	Last name
integer	age	Age
end group		
end repeat		

◄ ► **survey** | choices | settings | ⊕

The Delivery Outcome XLSForm is another repeat example.

Fixed repeat counts

Instead of allowing an infinite number of repeats, the form designer can specify an exact number of repeats by using the **repeat_count** column:

type	name	label	repeat_count
begin repeat	child_repeat		3
text	name	Child's name	
decimal	birthweight	Child's birthweight	
end repeat			

◄ ► **survey** | choices | settings | ⊕

In the above example, exactly **3** child repeats will be created.

Dynamic repeat counts

The repeat count can be set to an expression that refers to other fields in the form. In the example below, the number that the user inputs for the **num_hh_members** field dictates the number of **hh_member** repeats added:

type	name	label	repeat_count
integer	num_hh_members	Number of household members?	
begin repeat	hh_member		${num_hh_members}
text	name	Name	
integer	age	Age	
end repeat			

◄ ► **survey** | choices | settings | ⊕

Only add repeats in certain conditions

Like with groups, all of the questions in a repeat can be skipped based on some condition. In the example below, the person filling out the form will only be given the opportunity to add children if they first indicate that there are children to add:

type	name	label	relevant
select_one yes_no	has_child	Do any children live here?	
begin repeat	child_repeat		${has_child} = 'yes'
text	name	Child's name	
decimal	birthweight	Child's birthweight	
end repeat			

◄ ► **survey** | choices | settings | ⊕

list_name	name	label
yes_no	yes	Yes
yes_no	no	No

◄ ► | survey | **choices** | settings | ⊕

Representing zero repeats

By default, the person filling the form will see the questions corresponding to one repeat before getting the option to add more. To represent 0 repeats, there are three options:

- teach the people filling out the form to delete the first repeat added
- if the exact number of repeats is known ahead of time, use a dynamic repeat count
- if the exact number of repeats is not known ahead of time, use `relevant` to only prompt the user for repeats if there are some to add

Multiple language support

It's easy to add multiple languages to a form. You simply have to name your **label::language1 (code)**, **label::language2 (code)**, etc., and your forms will be available in multiple languages. See the example below. Select a different form language from the pulldown menu of data collection application (this may be located under the **Menu** key). For the form below, English and Español will show up as the possible options.

type	name	label::English (en)	label::Español (es)	constraint
integer	age	How old are you?	¿Cuántos años tienes?	. <= 150

◄ ► **survey** | choices | settings | ⊕

You can also add different language columns for hints and media files by using the same `::language (code)` construct, as shown in the example below. See also the XLSForm reference table, which includes a list of all column headers that can accept a language modification.

hint::English (en)	hint::Dutch (nl)	image::English (en)	image::Dutch (nl)
a hint	een hint	old_person_cartoon.png	ouwe_strip.png

◄ ► **survey** | choices | settings | ⊕

Form language and user interface language may be the determined separately by the application and may not match. To facilitate matching both (in the future), it is recommended, though optional, to add a 2-character language code after the language name. The official 2-character language codes, called *subtags* are published here (search the page with Ctrl-F or Cmd-F).

Media

You can include questions in your form that display images or that play video or audio files. If using the ODK mobile client for form submission, you need to put the media files that you want to include in the **/odk/forms/formname-media** folder on your phone, and then reference the exact file name in the **media** column in your form. See below for an example of how to do this.

type	name	label	image	video
note	media_example	Media example	example.jpg	example.mp4

◄ ► **survey** | choices | settings | ⊕

Check out the Birds XLSForm which illustrates the use of media files. You can also click on the link to see the Birds webform .

Media is translatable in the same way as labels and hints as explained in the languages section.

Pre-loading CSV data

Pre-loading data is done when one wants to reference pre-existing data in a survey form. You can be able to reference data in your survey form (the survey you are now authoring), from a pre- existing data in a specific survey form or any other source. For example if you have pre-existing data from a household survey and you want to collect follow-up data about the household occupants. You can be able to reference the household survey data in your survey form. To reference pre-existing data in a survey form:

- Upload one or more .csv files as support files when you upload your form definition (the same way you upload media support files as explained in the Media section).The first row of each .csv file should be a header that includes short:
  - unique names for each column
  - subsequent rows which should contain the data itself

Each csv file should contain at least one column that can be used to uniquely

identify each row. Such columns will be used, at survey time, to look up which row's data to pull into the survey. For the columns that will be used for looking up rows add **_key** to the end of the column name in the first row. Any columns with names ending in **_key** will be indexed for faster look-ups on your survey devices. See below an example of the columns on a .csv file:

name_key	name
mango	Mango
orange	Orange

How to pull data from CSV

You can be able to pull data from .csv file by including one or more .csv files in your form during the survey time. For each data field that you want to pull into your survey:

- Add a **calculate field** to your survey.
- Give that field a **name**
- Then in its **calculation** column, call the **pulldata()** function, indicating which field to pull from which row of which .csv file.

See below for an example:

type	name	label	calculation
calculate	fruit		pulldata('fruits', 'name', 'name_key', 'mango')
note	note_fruit	The fruit ${fruit} pulled from csv.	

◄ ► **survey** | choices | settings ⊕

Once you have loaded .csv data into a survey field using the **pulldata()** function, you can reference that field in later relevance conditions, constraints, and labels, just as you would reference any other field that was filled in by the user.

Click on the link to see an example of a pre-loading sample form and the .csv file used with form can be found here

Important notes on usage of pre-loaded data

- Compress a large .csv file into a **.zip archive** before uploading it.
- Save .csv file in **UTF-8 format** if pre-loaded data contains non-English fonts or special characters this enables your Android device to render the text correctly.
- Data fields pulled from a .csv file are considered to be text strings therefore use the **int()** or **number()** functions to convert a pre-loaded field into numeric form.
- If the .csv file contains sensitive data that you may not want to upload to the server, upload a blank .csv file as part of your form, then replace it with the real .csv file by hand-copying the file onto each of your devices.

Dynamic selects from pre-loaded data

If the recommended methods described in Multiple Choice from File do not meet your requirements you can consider the method below if your data collection application supports it.

Once your form has one or more pre-loaded .csv files, you can dynamically pull the choice lists for **select_one** and **select_multiple** fields from those .csv files. Multiple-choice fields with dynamic choice lists follow the same general syntax as regular, static select_one and select_multiple fields as previously covered in the Multiple choice questions section.

The following should be done:

- specify **select_one listname** or **select_multiple listname** in the type column (where **listname** is the name of your choice list)
- specify any special **appearance styles** in the appearance column
- include one or more rows for your listname on the choices worksheet.

Below is an example of the **survey worksheet**:

type	name	label	appearance
select_one fruits	fruits	Select a fruit	search('fruits')
◄ ► **survey** choices settings ⊕			

There are three differences when the choice list should be pulled from one of your pre-loaded .csv files:

- In the appearance column:
- Include a **search() expression** that specifies which .csv rows to include in the choice list.
- If the field should use a non-default appearance style. The non-default appearance style goes into the column first, followed by a **space**, then the **search() expression**. [e.g., **quick search()**]
- On the **choices worksheet**:
- a row should indicate which .csv columns to use for the label and selected value. As follows:
  - **list_name** column: specify the name of your choice list as you normally would.
  - **name** column: include the name of the .csv column to use for uniquely identifying selected choices.
  - **label** column: include the name of the .csv column to use for labeling the choices.
    Note:
    If you wish to include multiple columns in the labels, include a comma-separated list of all columns to include. The name column will be dynamically populated based on the column name you put there, and the label column will be dynamically populated based on the column name(s) you put there.
- In your choices worksheet row, you may also include a .csv column name in the image column. If you do, the image filename to use will be pulled from the specified .csv column.
  Note:
  If you refer to image files in this way, you must always upload those image files as media file attachments when you upload your form to the server.
  See below an example of the choices worksheet:

list name	name	label
fruits	name_key	name
◄ ► survey **choices** settings ⊕		

Click on the link to see an example of a search-and-select sample form and the .csv file used with form can be found here.
There are a series of options to indicate which .csv rows to include in the choice list using the **search() expression**, see this post for additional information on these search() expressions.

Cascading selects

A lot of forms start out by asking the location of the respondent, with each location selection specifying what the subsequent location choices will be (e.g., state » district » village). Instead of adding a **select_one** field for each location option, you can use cascade select. In order to use cascade selects, you will need to create a **choice_filter** column in your survey worksheet and add the location attribute columns in your choices worksheet. Check out an example XLSForm here.

External selects

If a form has selects with a large number of choices (e.g., hundreds or thousands), that form can slow down form loading and navigation if Multiple Choice from File is used. The best workaround to this issue is to use external selects in those data collection applications (such as ODK Collect) that support it.

Enabling external selects is straightforward.

- Instead of **select_one** for the prompt type, use **select_one_external**.
- Instead of the **choices** sheet, put external choices in the **external_choices** sheet.

See select_one_external form for an example that uses normal and external choices.

When an XLSForm with external choices is converted to an XForm, two files will be produced, the **XForm** (e.g., form-filename.xml) with all the normal choices, and an **itemsets.csv** with the external choices.

The **itemsets.csv** file can be uploaded to any ODK-compatible server (e.g., ODK Aggregate) as a media file. It will be downloaded to any ODK-compatible (e.g., ODK Collect) like any other media file and saved to the [form-filename]-media folder. Clients like ODK Collect load media files from the SD card and so your form with a large number of choices will now load very quickly.

Default

Adding a default field means that a question will be pre-populated with an answer when the user first sees the question. This can help save time if the answer is one that is commonly selected or it can serve to show the user what type of answer choice is expected. See the example below.

type	name	label	default
date	survey_date	Survey date?	2010-06-15
decimal	weight	Respondent's weight? (in kgs)	51.3
◄ ►	**survey**	choices	settings ⊕

The respondent can simply change the answer by tapping in the answer field and entering another answer.

You can also add a default calculation, which will only be calculated only once when the form loads or - if the question is inside a repeat - when the repeat is added.

type	name	label	default
date	d	Enter the date the event occurred?	today()
◄ ►	**survey**	choices	settings ⊕

Read only

Adding a read only field means that a question can not be edited. Read only fields can be combined with default fields to deliver information back to a user.

type	name	label	read_only	default
integer	num	Please patient is:	yes	5
◄ ►	**survey**	choices	settings ⊕	

Appearance

The **appearance** column allows you to change the appearance of questions in your form. The following table lists the possible appearance attributes and how the question appears in the form.

Appearance attribute	Question type	Description
multiline	text	Best if used with web clients, makes the text box multiple lines long.
minimal	select_one, select_multiple	Answer choices appear in a pull-down menu.
quick	select_one	Relevant for mobile clients only, this attribute auto-advances the form to the next question

		after an answer is selected.
no-calendar	date	For mobile devices only, used to suppress the calendar.
month-year	date	Select a month and year only for the date.
year	date	Select only a year for the date.
horizontal-compact	select_one, select_multiple	For web clients only, this displays the answer choices horizontally.
horizontal	select_one, select_multiple	For web clients only, this displays the answer choices horizontally, but in columns.
likert	select_one	Best if used with web clients, makes the answer choices appear as a Likert scale.
compact	select_one, select_multiple	Displays answer choices side by side with minimal padding and without radio buttons or checkboxes. Particularly useful with image choices.
quickcompact	select_one	Same as previous, but auto-advances to the next question (in mobile clients only).
field-list	groups	Entire group of questions appear on one screen (for mobile clients only).
label	select_one, select_multiple	Displays answer choice labels (and not inputs).
list-nolabel	select_one, select_multiple	Used in conjunction with **label** attribute above, displays the answer inputs without the labels (make sure to put **label** and **list-nolabel** fields inside a group with **field-list** attribute if using mobile client).
table-list	groups	An easier way to achieve the same appearance as above, apply this attribute to the entire group of questions (might slow down the form a bit).
signature	image	Allows you to trace your signature into your form (mobile clients only).
draw	image	Allows you to sketch a drawing with your finger on the mobile device screen.
map, quick map	select_one, select_one_from_file	Allows a user to select a choice from many features on a map

An XLSForm with all of the appearance attributes in this table is available here.

Settings worksheet

The **settings** worksheet is optional, but it is highly recommended to specify **form_title**, **form_id** and **version** at a minimum. Other settings allow you to further customize your form, including setting an overall style theme or encrypting your records.

An example **settings** worksheet is below:

form_title	form_id	version	instance_name	default_language	public_
Example	ex_id	2017021501	concat(${firstname}, ' ', ${lastname})	English (en)	IIBIjANBg

◄ ► | survey | choices | **settings** ⊕

The settings column headings available are:

- **form_title**: The title of the form that is shown to users. The form title is pulled from **form_id** if **form_title** is blank or missing.
- **form_id**: The name used to uniquely identify the form on the server. The form id is pulled from the XLS file name if **form_id** is blank or missing.
- **version**: String that represents this version. A common convention is to use strings of the form 'yyyymmddrr'. For example, 2017021501 is the 1st revision from Feb 15th, 2017.
- **instance_name**: Expression using form fields to identify for each form submission. Learn more.
- **default_language**: In localized forms, this sets which language should be used as the default. The same format as described for adding translations should be used, including the language code.

- **public_key**: For encryption-enabled forms, this is where the public key is copied and pasted. Learn more.
- **submission_url**: This url can be used to override the default server where finalized records are submitted to. Learn more.
- **style**: For web forms, specify the form style. Learn more.
- **name**: XForms root node name. This is rarely needed, learn more.

Encrypted forms

Encryption-enabled forms provide a mechanism to keep *finalized* records private at all times. This includes the time *after a record is marked as final* that it is stored on the device and server as well as during transport, even when http is used for communication. Encrypted records including their uploaded files, such as photos, are completely inaccessible to anyone not possessing the private key.

To encrypt XLS forms, add the **public_key** column to the **settings** worksheet and paste the base64-encoded public RSA key as its value.

form_id	public_key
mysurvey	IIBIjANBgklawWEserewrwesgdreewrwe32serfserfewrwerewtwer23sgfrqjwerk3423432..
◄ ► | survey | choices | **settings** ⊕	

For more information on encrypted forms and how to generate the RSA keys have a look at the ODK documentation and at this example form.

Specify alternative server

It is possible to specify an alternative server to send your submissions to in the **submission_url** column on the **settings** worksheet. Make sure to use the full URL that submissions should be sent to *including the path*.

If this column is left out or kept empty, submissions will go the default destination for the provider you are using for your surveys.

Specify form submission name

In the **settings** worksheet, you can specify a unique name for each form submission using fields filled in by the user during the survey. On the settings worksheet, add a column called **instance_name**. Write in the expression that defines the unique form instance name using fields from the survey worksheet.

Check out this example XLSForm that calculates the instance name as the user's last and first names coupled with the form submission uuid.

Specify XForms root node name

In some rare cases, it may be helpful to explicitly specify a root node name for the generated XForm. For example, this may be necessary if updating a form that was converted with an older form converter that used a root node name other than `data`. In the **settings** worksheet, you can specify an identifier to use for the XForms root node name by adding a collumn called **name**. By default, the XForms root node name is `data`.

Multiple webpage forms

Web forms can be split into multiple pages using the style theme **pages**.

An example of a form divided into multiple pages can be seen on the Widgets on Pages webform.

In the **settings** tab, create a column called **style** and set it to **pages**, as follows:

form_title	form_id	style
example title	example_id	pages
◄ ► | survey | choices | **settings** ⊕		

In your **survey** tab, group together the questions you would like to appear on each page and then set the appearance for the group to **field-list**. See the example below.

type	name	label	appearance
type	name	label	appearance
begin group	group1		field-list
text	name	Respondent's name	
integer	age	Respondent's age	
text	address	Respondent's address	
end group			
◄ ► **survey** choices settings ⊕			

See this blog post for more information on creating multi-page web forms. The XLSForm source is here.

Grid theme forms

The **theme-grid** style allows your form to mimic the look of traditional paper surveys by compacting multiple questions into one row. This style is best used with larger screens (e.g., computers or tablets). It also makes a nice print out!

Please click on the link to see an example of a Grid theme webform.

To create a Grid form, in the **settings** tab, under the **style** column, write **theme-grid**, as follows:

form_title	form_id	style
example title	example_id	theme-grid
◄ ► survey choices **settings** ⊕		

In your **survey** tab, group together the questions you would like to appear in each section and then set the appearance for each field according to the desired width (the default width is 4). See the example below.

type	name	label	appearance
begin group	group1		
text	name	Respondent's name	w3
integer	age	Respondent's age	w1
text	address	Respondent's address	w4
end group			
◄ ► **survey** choices settings ⊕			

See this blog post for more information on creating Grid forms. The Grid theme XLSForm example is here.

Styling prompts

Markdown support in XLSForm allows for increased emphasis through bold and italics, different sized headers, various fonts and colors, and clickable web links in ODK Collect 1.4.9 and Enketo.

- *emphasize* words by wrapping them inside _ or *
- **strongly emphasize** words by wrapping them inside __ or **
- add a link by using `[name of link](url)`
- add various sized headers by prepending # (biggest) to ###### (smallest) to header text
- style text for color or font with span tags (e.g., `<span style="color:#f58a1f">`orange`</span>`, `<span style="color:red; font-family:cursive">`red and cursive`</span>`)
- add a line break where you want it with Ctrl-Enter or Ctrl-Alt-Enter (may be different key combination for some spreadsheet software)
- add your favorite emojis 😍😍!
- use superscript with the `<sup>` tag (e.g. `100 m<sup>2</sup>` turns into 100 m^2)

- use subscript with the <sub> tag (e.g. H₂0 turns into H_2O)
- use the \ character before #, *, _, and \ to prevent special styling effects to be triggered by these characters

Advanced use and extensibility

It is possible to use XLSForm to create XForms with custom or experimental features. This is great for custom applications with a specific feature that is not suitable for the larger community.

The **survey** sheet has support for 3 column prefixes (**instance::**, **bind::**, **body::**) that add attributes to the XForm output, either in the *primary instance*, *bind*, or *form control*. To learn more about XForms visit the ODK XForms Specification. The example below adds a custom "hxl" attribute to the primary instance node of a question.

type	name	label	instance::hxl
integer	population	How many people present?	#population
◄ ►	**survey**	choices	settings ⊕

The **settings** sheet has support for defining (multiple space-separated) additional custom namespaces and namespace prefixes using the **namespaces** column. You'll then be able to use those namespaces in the survey sheet, for example to properly define a custom attribute with your organisation's own namespace. See example below that adds 2 additional namespaces and uses them to add custom attributes:

title	namespaces
My Form	esri="http://esri.com/xforms" enk="http://enketo.org/xforms
◄ ►	survey	choices	**settings** ⊕

type	name	label	bind::esri:fieldLength	bind::enk:for
text	desc	Describe	50	
text	desc_comment	Comments		${a}
◄ ►	**survey**	choices	settings ⊕	

Tools that support XLSForms

- Enketo
- Survey123 for ArcGIS
- SurveyCTO
- CommCare
- DataWinners
- Ona
- KoBoToolBox
- Community Health Toolkit
- Secure Data Kit (SDK)
- ODK
- Tattara

More resources

The XLSform standard document can guide you through the specific input types, column headers, and so on that are legitimate syntax in XLSForms. If you want to dig in deeper to understand XForms and go beyond XLSForms, here are some resources to understand them:

- XForms as supported by the ODK ecosytem
- ODK Form design guidelines
- Ona Form design overview
- KoBoToolbox form design help center

About this site

XLSForm.org is a community-supported project aiming to create a common reference point for the XLSForm standard.

If you want to contribute to or improve this documentation, please visit our project's GitHub repo.

History

The XLSForm was originally developed by Andrew Marder and Alex Dorey of the Sustainable Engineering Lab at Columbia University. As XLSForms became adopted by the ODK Community, SEL worked with the ODK Team to develop the current specification. PyXForm, the library used to convert XLSForms to XForms, is an open source project supported by members of ODK, SEL, Ona, SurveyCTO, and KoBoToolbox.