Then, in the second row of that new column, add an expression like the following:

concat('HH SURVEY - ', ${hhid})

Replace "HH SURVEY" with whatever prefix you want to include in form names, and replace "hhid" with whatever field name contains a unique identifier for the form. Alternatively, if you want to include multiple fields, you could enter an expression like this:

concat('HH SURVEY - ID ', ${hhid}, ' - Name ', ${respondent_name})

In this example, both a household ID and a respondent name are included in the form's name. That way, when you list filled-out forms on a device, you will be able to more easily find specific instances.

Using variations of the concat() function, you can concatenate together an arbitrary number of literal strings (enclosed by single-quotes) and an arbitrary number of form fields (enclosed by ${...}). Note that whatever you put in the instance_name column should evaluate to a string; so, for example, "today()" alone won't work to name forms using the current date, but "string(today())" will work.

top

How long can my survey forms be?

There is no fixed limit for how long your survey forms can be. The primary limiting factor will be your data-collection device: if your form is too long for your device's CPU power, it may become very slow, and if your form is too long for your device's memory, it may crash.

See *How do I choose a device?* for factors to consider when choosing a data-collection device. Importantly, devices running versions of Android prior to 3.0 severely limit the amount of memory that any one app can access; from Android 3.0 on up, apps can access more memory – but the device itself still decides how much of the overall memory to allow one app to utilize.

To get an idea of how different devices perform with very long forms, you can test devices yourself using a sample form with 500 fields, a sample form with 1,500 fields, or a sample form with 3,000 fields.

Note that, when you upload a very long form to the server, it may take several minutes to process. Note also that, when you choose to fill out a very long form in Collect, it may take a long time to load; once it has been loaded once, however, it will usually load substantially faster when selected again.

Generally, you can shorten a very long form by using repeat groups, and/or by splitting it into multiple pieces linked on the back-end with a unique ID or scanned barcode. If survey modules will always be filled out in sequence, you can even link modules automatically in your back-end processing code based upon forms' start-time, end-time, and device-id values.

You might also be able to improve your form's performance. See the next help topic for some ideas.

top

Are there ways to improve the performance of my form?

If you have a very long form that's performing badly, first consider the discussion in the help topic just above this one. The solution may be to use a newer, more powerful device if your form is very demanding; or, you might choose to split one very long form into several smaller ones.

However, you might also be able to improve the performance of your form by altering some aspects of its design. To begin with, it's helpful to understand the two key reasons why users might encounter sluggishness when filling out a form:

1. **Memory.** All form fields, choices on the *choices* sheet, and translations are stored in memory while a form is being filled out. This includes repeated fields, so if you have a 100-field repeat group that's repeated 20 times, then that's 2,000 fields in memory (even if the fields are not currently relevant and so aren't visible). To protect the OS and other apps, Android only gives a limited amount of memory to each app – and when memory gets low things start getting very sluggish because of how Android tries to conserve remaining memory.

2. **Expression dependencies.** Every reference to another field or group within a relevance, calculation, or constraint expression creates a dependency that is watched for potential changes, so that the appropriate expressions can be re-evaluated whenever something on which they depend might have changed. So if Field A refers to Fields B and C and Field C refers to Field D, then swiping forward or backward from Field D will trigger re-evaluations of the expressions for Fields C and then A. If you have 3,000 fields in memory (including all instances of repeated fields), then there will be up to 3,000 relevance, constraint, and calculation expressions that are always being considered for re-evaluation (including the relevance expressions for all of those fields not currently visible). So when you swipe from one field to the next and there is a delay, ask yourself: what can it be doing just then? Think about the field that you're swiping away from and what might depend on that field's value. Sometimes, with a long, complex form, one field can trigger several other fields to recalculate, and those can trigger several more, and the dependencies run so deep that hundreds or even thousands of expressions will be re-evaluated. Thus, in this way, performance will depend on the logic of your survey.

Understanding these causes of sluggishness, there are a few things that you can do to improve performance:

1. **Move choices to pre-loaded .csv files.** If you have a large number of choices on your *choices* worksheet (e.g., over 1,000), you should consider moving longer lists to pre-loaded .csv files. That will move those lists out of memory and will, in general, perform much better. See this help topic on pre-loading multiple-choice options for details.

2. **Reduce the number of repeats.** Some form designs involve a large number of repeated groups, most of which are never visible. For example, say that you want to ask a long series of questions about certain assets that might be found in a household; if there are 175 different types of asset, you might create a repeat group that's repeated 175 times, with only the relevant assets visible for any given household. In principle, this is a fine design – but the 175x fields in memory can cause the form to perform poorly. Generally, in these cases, there is an alternative design that achieves the same result without so many fields in memory. (In this example case, you could have a multiple-choice question with 175 choices, then repeat the list of questions just once for each selection made by the user. That way, there are only as many fields as necessary to cover the assets in any given household.)

3. **Reduce the number of dependencies.** If you've designed your form such that long, complex expressions create such intricate dependencies between fields that any change to one field results in re-evaluation of dozens or hundreds of complex down-stream expressions, the solution may be to simplify your form. However, you should be able to take full advantage of the electronic medium to constrain user entries, catch potential errors, etc. – so in some cases the solution may be to deploy your survey on more powerful devices that can handle whatever logic your survey requires.

Neither Android nor SurveyCTO is infinitely powerful, so there will always be limits to what can perform well. But devices grow more powerful by the day, and we at SurveyCTO are always pushing the performance frontier outward. If you are using an older version of SurveyCTO, you should try upgrading to the latest version to see if that performs better. We try to include performance improvements whenever we can.

top

How do I download and export my filled-out form data?