Help - Collecting data - Data-collection workflow

Below, please find topics related to the overall management of your data-collection workflow. This concerns workflow for the data-collection itself; for monitoring of incoming data, managing the review of that data, and related topics, see the sections on basic management and monitoring and managing for quality.

Designing workflow

With SurveyCTO, much of the "workflow" of filling out survey forms depends on the design of your forms: users move through the various form fields that you have defined, filling them out, being subjected to constraints that you have configured, etc. But when you're deploying your forms on Android devices, key elements of the workflow are controlled not by individual survey forms but instead by how the *SurveyCTO Collect* Android app is configured. This help topic discusses that configuration.

You can find *SurveyCTO Collect*'s settings in two places: from the main Collect menu, press your device's menu button (or the three vertical dots in the corner on some newer devices) and choose either *Admin Settings* or *General Settings*. These settings control aspects of the user experience before, during, and after the actual filling out of survey forms.

Before

You can control the options a user sees in the *SurveyCTO Collect* main menu, as well as the settings that they can edit. Once you have configured devices with the desired survey forms, for example, you might limit the main menu choices to only *Fill Blank Form*, *Edit Saved Form*, and *Send Finalized Form*; you might also prevent users from changing any of the settings (without a special password). See *Managing device settings* for details.

In the standard workflow, users run the *SurveyCTO Collect* app and choose *Fill Blank Form* to start filling out a form. However, you can also place a shortcut to that form directly on the device's home screen, so that users can boot up their devices and start filling out a new form with just one click. The process differs a bit by device: on some devices, you hold your finger down on an empty part of the home screen, then choose "Add Shortcut", "SurveyCTO Form", and the form you'd like the shortcut to launch; on others, you click into the Apps view, then choose "Widgets", "SurveyCTO Form", and, after dropping the widget onto an empty location, select the form to launch.

You can also organize the filling out and editing of forms around specific "cases," where cases might be households, spouses, health centers, schools, etc. See *Case management* below for details.

During

While the user is filling out a form, most of the experience is controlled by the design of the form itself. This primarily involves a sequential process by which the user fills out form fields, one by one. However, *Designing for easy navigation* discusses more advanced navigation options open to the form designer.

Typically, the user navigates forward and backward by "swiping" with his or her finger. However, there is a *Navigation* setting in *General Settings* that allows you to enable on-screen forward and back buttons; these buttons can be used in addition to or instead of swiping.

It is also typical to enforce field constraints field-by-field, as the user progresses through each survey form: if a user enters an invalid value (e.g., an implausible age, such as 122), he or she is not allowed to continue until the entry is corrected. The *Constraint processing behavior* setting in *General Settings* allows you to override this behavior and instead defer all field validation to the point at which the survey is finalized (at the very end). For the vast majority of cases, the default – validating field-by-field – is the way to go, but it depends on your preferences.

While within-form navigation is mostly controlled by the form designer, there is an option for the user to jump from one place to another, outside the standard sequence. This option is typically used, for example, when the user realizes that an earlier response was mistaken and wishes to correct it. He or she can do so by clicking the device's menu button, choosing *Go To Prompt* (or clicking the button with the arrow pointing to the dot), scrolling to the desired question, and clicking on it to jump there. If you want to prevent users from being able to jump around in this way, open *Admin Settings*, scroll down to the *User can access form entry items* section, and un-check *Go To Prompt*.

In the standard, field-by-field sequence, the user cannot skip fields that are marked as *required* or proceed if an entry violates a constraint. There may be circumstances where the user cannot yet fill out a particular response and wants to move on and circle back to fix it later (e.g., the response concerns entering a child's weight but the user is awaiting the necessary scale, or the particular respondent necessary for a question is currently busy). In such a circumstance, the user can press the device's menu button and select *Skip to Next*; this is effectively a shortcut for choosing *Go To Prompt*, scrolling down to the next question, and clicking on it to jump there. If you want to prevent users from being able to jump in this way, open *Admin Settings*, scroll down to the *User can access form entry items* section, and un-check *Skip to Next*.

If an interview is interrupted for some reason, the user needs to be able to save, exit, and resume later. He or she does this by clicking the device's built-in back button to exit the form. A pop-up menu will ask whether to *Save Changes* or *Ignore Changes*. The latter option will delete any data entered since the form was last saved, so Collect will ask the user to confirm that they wish to exit without saving their changes. If the user instead chooses to *Save Changes*, he or she can resume later by selecting *Edit Saved Form* from the main Collect menu and choosing which form to edit. If going back to edit forms later is common in your workflow, see *Dynamically naming filled-in forms* for information on how to name forms in a way that makes them easier to identify and edit.

Validating, saving, and finalizing

Before a form can be submitted to the server, it must be "finalized", and before a form can be finalized, it must be free of any errors – in other words, all required responses must be entered, and all entries must satisfy their respective field constraints. This is because the survey designer's specifications for which fields are required and what constraints should be enforced is what produces "clean" data (i.e., data that is ready for review and analysis). Finalized forms cannot generally be edited, so finalization is the last (in-the-field) step of the data-collection process.

In some projects, individual enumerators finalize their respective forms upon concluding a field survey. Other projects require enumerators to save their forms and review them later, before finalizing. And some projects may require that a supervisor review the form for mistakes (using *Edit Saved Form*) before it can be finalized.

When the user reaches the end of a survey form, he or she is presented with a final screen that displays a *Save and Exit* button. That screen also has a checkbox for whether or not to finalize the form. By default, that checkbox is checked – but you can change this default by un-checking the *Default to finalized* option in *General Settings*. You can also hide the checkbox altogether by unchecking the *Mark form as finalized* option in *Admin Settings*.

When a user marks a form as finalized, SurveyCTO will validate the entire form to confirm that all required fields are complete and that all constraints have been met. If there are any problems, the user will be forced to correct them.

SurveyCTO does not validate a form or force the user to make corrections when the user saves without finalizing. You can override this default with the *Saving behavior* option in the *Advanced Settings* section of *Admin Settings*: if you choose *Force corrections on Save and Exit*, the form will be validated and the user will be forced to make corrections before being allowed to *Save and Exit* – even when he or she is not yet finalizing. Note, however, that the user will still have the option to exit the form if an interview cannot be completed by using the device's built-in *back* button to exit and save changes. If you do not want to force corrections on *Save and Exit*, there are two options in the *Saving behavior* setting: (1) *Don't check answers (quick save)* and (2) *Don't check answers (slow save)*. Of the two, the "quick save" option is generally preferred; the slow-save option is available only for backward compatibility with older versions of SurveyCTO.

Finally, if users jump around within a survey – and/or if you have set validation to happen at the very end rather than field-by-field – they can easily lose track of what has already been completed and what remains incomplete. While any constraint violations or missing required responses will be flagged at the very end, users may want to jump back and start cleaning up required or invalid responses earlier in the interview process. To prompt SurveyCTO to validate every field from beginning to end, users can choose *Validate Form* in the the device's menu. If Collect identifies any problems, the user will be sent back to correct them. This feature allows the user to jump around the form without missing required responses. Similar to other user options, you can disable this feature if you wish: simply un-check *Validate Form* in *Admin Settings*.

After

Presuming that you are not pulling data directly off your devices via local wi-fi sync, collected data has to be sent to your server, where it will be aggregated, de-duplicated, and made available for download, export, and/or publishing. This might be immediate (e.g., if all of your devices have sim cards and data connections), it might be nightly (e.g., if all of your devices can reach a wi-fi hotspot, such as a data connection shared as a hotspot by a supervisor's device), or it might be less often. There are a few basic workflows through which forms can be submitted:

1. The most standard workflow is manual: once an Internet connection is available, the user selects *Send Finalized Form* from the main Collect menu, checks any or all listed forms, and clicks *Send Selected*. By default, Collect will only list forms that have been finalized but not yet sent to the server, so the user's task is to check all, send, and await confirmation.

2. There are also two options to automatically send finalized forms: in Collect's *General Settings* you can choose *Auto send with Wi-Fi* and/or *Auto send with network* ("network," in this case, meaning a cellular data network). If you check one or both of these options, Collect will automatically attempt to send all finalized forms to the server whenever the allowed connection type is available. (There are also two "auto update" settings that can be used to automatically check for and download form updates – which are particularly useful when forms include pre-loaded data that changes from time to time.)

Case management

By default, *SurveyCTO Collect* is set for the most standard workflow: users click *Fill Blank Form* and *Edit Saved Form* to fill out and edit forms. But you can instead organize the filling out and editing of your forms around *cases* in order to shift the workflow to be centered not on the forms themselves, but instead on the households, schools, health centers, patients, or other subjects of your data-collection efforts.

Organizing your data collection around case management may be appropriate when (a) you can pre-define a specific set of cases, and (b) you want one or more forms filled out for each case. As noted above, these "cases" might be households, schools, health centers, patients, or any other subject of data collection.

Here's how it works. First, you create a "cases" dataset on your server, to hold your list of cases. To do this, navigate to the Design tab, scroll down to the *Your forms and datasets* section, and click + and then *Add server dataset*; switch to the *New dataset for cases* tab, leave the title and ID alone, and click *Create dataset*. Next, click the *Edit* button to start adding cases to the dataset, or upload a .csv file that contains details for each of your cases. Each case should be a row with columns for each of the following details. And if you're uploading a .csv file, the bolded names below are exactly what should appear in your .csv file's first row (as the names of the columns):

1. **id**. This is the unique ID that identifies each case. It might be a household ID, a clinic ID, a patient ID, a student ID, etc. It can be in any format, but each case must have a unique value, and your cases list should only include a single row for each unique **id**.
2. **label**. This is the label that will be used to identify each case when shown in Collect's user interface. While the **id** is the unique identifier behind the scenes, the **label** is what users will see.
3. **formids**. This is the comma-separated list of unique form ID's that are associated with a case. In other words, for each case, this is the list of forms that the user can fill out. If you only have one form for a case, just include that form's unique ID here (which is the ID on its *settings* worksheet); if you have multiple forms, separate each ID with a comma (as in "formid1, formid2, formid3").
4. **users** (optional). If present, this is the comma-separated list of users for whom the case should be listed; the entire column can be omitted to show all cases to all users, or individual rows can have blank entries for a subset of cases that you want all users to see. Most often, you will either not specify any user names (so that the case will be listed for all users), or you will list just one user name (to list the case only for devices configured with that user name or web users logging in with that user name). If you want to list a case for more than one user (but not all of them), separate the user names with a comma (as in "user1, user2, user3").
5. **roles** (optional). If present, this is the comma-separated list of role IDs for which the case should be listed. (This column is only useful for enterprise subscribers who create custom roles for their users.)
6. **sortby** (optional). If present, cases will be listed in the order indicated by the numeric values you enter in the **sortby** column (from low to high); if not present, cases will be sorted by their **id** values.

When you first create your "cases" dataset, you can also choose how the case list should appear to users: as a "tree list" in which all cases appear with all relevant forms indented underneath, or as a more compact "table" that summarizes information about each case and allows the user to click to see the relevant forms for each. If you select the table view, you can choose which columns, from your cases dataset, to display, in which order; you can include the standard columns listed above – or other custom columns if you've included them in your dataset – and users can scroll to find the appropriate cases, click column headings to sort, and enter text to filter the list as needed. If you want to change how your case list displays, you can always go back to edit the *Settings* for your cases dataset.

Once you have a cases dataset on your server, logged-in web users will automatically see a *Manage Cases* button. For users of *SurveyCTO Collect* on Android, you will need to manually enable the *Manage Cases* button: from the main Collect menu, click your device's menu button, choose *Admin Settings*, and then enable the checkbox next to *Manage Cases*; when you return to the main menu, you should see a *Manage Cases* button at the top.

Web users will always see the latest set of cases when they click *Manage Cases*. Android users will need to click *Manage Cases* and then click *Refresh* to load the latest cases from the server. The list of cases shown to the user will be filtered based on the current user name and role, according to the **users** and/or **roles** columns in the cases list (if any).

For each listed case, the user will have the opportunity to fill out any of the forms listed in the case, edit saved forms, and (on Android) see a list of finalized forms; if an Android user hasn't already used *Get Blank Form* to download all of those forms to the device, they might see some errors about missing forms. If all of your forms will be filled out via *Manage Cases*, you can disable the *Fill Blank Form* and *Edit Saved Form* buttons in *Admin Settings* of the Android app.

If you have enabled auto-downloading of form updates in the Android app's *General Settings*, then updates to the cases list will be automatically downloaded as well (with Collect checking for updates roughly once per hour when connected); alternatively, you can always re-download the latest cases by clicking *Manage Cases* and then *Refresh*.

Finally, you can automatically store the unique case ID as part of your form data – and even use that ID to, e.g., pre-load data for the case – by including a *caseid* field in your survey. Actually, new forms begun with SurveyCTO automatically include such a field, just in case you're using case management; in the online form designer, you can check to make sure that it's included as a meta-data field in the *Form settings*. The *caseid* field will be automatically populated with the unique **id** of the case for which the user filled out the form (if they filled out the form using the *Manage Cases* interface; otherwise it will be blank). You can then use that field to pre-load data and/or track data associated with the case.

You can use this case-management interface to customize your data-collection workflow. See also *Designing workflow* for more options available to you.

Cases list as server dataset

In SurveyCTO, the cases list is always stored in a server dataset with the unique ID "cases" – so you can also attach that list to any form as pre-loaded data, publish form data directly into the cases list, publish the cases list to Google Sheets, and more. See *Advanced publishing with datasets* and *Pre-loading data into a form* for more about your options for working with server datasets.