19 Form Building Tools

20 Tips and Best Practices

21 ODK Briefcase

22 Setting Up ODK Briefcase

23 Contributing to ODK Docs

24 OpenRosa

25 ODK JavaRosa

26 Launching ODK Collect from External Apps

27 Briefcase Aggregate API

28 Encrypted Forms

29 Security and Privacy

30 Glossary
**ODK** is a suite of open source tools that help organizations collect and manage data. For a quick start, read *Getting Started With ODK*. In most cases, users of ODK:

- Create survey forms using the XLSForm standard in Excel or Google Sheets.
- **Upload forms to an ODK Central server.**
- **Download forms** into **ODK Collect** on an Android device.
- Use Collect to **fill out forms** with participants.
- **Upload survey data** from Collect to Central.
- **Analyze or export data from Central.**

This requires:

- **Installing Collect on an Android device**
- **Installing Central on a server**

While this is the recommended workflow, it is not the only way to do things. ODK is a very flexible set of tools, and organizations will find their own best practices for adopting it.

Additional ODK tools are:

- **ODK Build**, a drag-and-drop form designer. We generally recommend XLSForm for its flexibility but users only building very simple forms may prefer Build.
- **ODK Briefcase**, a desktop tool that pulls and exports data from Aggregate and Collect.

The specifications and libraries that power the tools are:

- **OpenRosa**, APIs for how ODK clients communicate with ODK servers.
- **ODK XForms spec**, a subset of the W3C XForms specification, for use in the ODK ecosystem.
- **ODK JavaRosa**, a Java library that renders forms complying with ODK XForms.
- **XLSForm spec**, a high-level Excel-based form specification.
- **pyxform**, a Python library that converts XLSForms into ODK XForms.

For a complete list of our tools, check out ODK on GitHub.
Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
GETTING STARTED WITH ODK

This document walks you through the recommended workflow to get started with ODK. You will:

- Get a Central server
- Get the Collect app
- Create a form with XLSForm and upload it to Central
- Load a form into Collect from Central
- Fill out a form and upload it to Central

1.1 Get a Central server

The easiest way to get a Central server is by using ODK Cloud.

ODK Cloud provides fast Central servers with regular feature updates, automatic security patches, daily backups, uptime management, enterprise security, and guaranteed support on any issues.

If you are technical and prefer to self-host, we recommend you install Central on DigitalOcean.

1.2 Get the Collect app

The easiest way to get the Collect app is to download it from the Google Play Store.

You can also install manually from an APK.
1.3 Create a form with XLSForm and upload it to Central

1. Create a document in your favorite spreadsheet tool (Excel, Google Sheets, etc).
2. Design your form using XLSForm or try a sample XLSForm.
3. Upload the form to Central.

1.4 Load a form into Collect from Central

1. Create an App User in Central.
2. Open Collect on your Android device, tap Configure with QR code and scan the Client Configuration Code created for your App User.

1.5 Fill out a form and upload it to Central

1. Select Fill Blank Form to complete a survey.
2. Select Send Finalized Form to upload your completed survey to Central.
Now log back into Central and see your completed survey results.
ODK Collect is an open source Android app that replaces paper forms used in survey-based data gathering. It supports a wide range of question and answer types, and is designed to work well without network connectivity.

ODK Collect renders forms into a sequence of input prompts that apply form logic, entry constraints, and repeating sub-structures. Users work through the prompts and can save the submission at any point. Finalized submissions can be sent to (and new forms downloaded from) a server.

Collect supports location, audio, images, video, barcodes, signatures, multiple-choice, free text, and numeric answers. It can even accept answers from other apps on your device. See a complete list of supported question types here.

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
Chapter 2. ODK Collect

2.1 Learn more about ODK Collect

- Setting Up ODK Collect
- Using ODK Collect
- Getting Started With ODK

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
3.1 Installing Collect

3.1.1 Installing from Google Play Store (recommended)

ODK Collect is available in the Google Play Store.

3.1.2 Installing manually

The Google Play Store always has the latest stable release.

If you need a different version of Collect, you can download from the web and install manually.

1. From your device’s Settings, choose Security.
   - Make sure Unknown Sources is checked.
   - (On older versions of Android, this setting is in Applications rather than Security)

2. Open a web browser on your phone.

3. Navigate to https://github.com/getodk/collect/releases/latest and download the ODK Collect APK.

4. In the download window, you will see ODK_Collect_vN.N.N.apk. - Select it to download the file.
   - On older devices, the APK will automatically install after you approve the security settings.
   - On newer devices, you must go to the download list, rename the file to restore the .apk extension (the extension will have been renamed to .man during the download process), then click on it to install it.

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
3.1.3 Updating Collect

We generally recommend letting the Android Play Store automatically update Collect. We strongly encourage anyone managing a data collection project to sign up for the beta program to read about upcoming changes and try their forms with any new version before it is released to all users.

The Collect version is displayed at the bottom of the Collect first launch screen and main menu. If you are doing an in-person training, we encourage you to walk the room and verify that everyone is using the same version. You can also ask enumerators to check this version themselves when an update rolls out.

Collect uses date-based versioning. For example, v2021.2.3 came out in 2021, was the 2nd release, and the 3rd patch on that 2nd release. Releases add new features and patches only fix bugs. There are 3 to 6 releases of Collect per year.

3.1.4 Understanding permissions

The ODK Collect application requires permissions to access device resources.

If your device runs a version of Android below 6, you will be prompted to accept all of the permissions on app install.

If your device runs Android 6 or above, permissions will be requested as they are needed. For example, the camera permission will only be requested the first time that you try to take a picture. Once the permission is granted, you won’t be prompted again. If you deny the permission, you will be asked again the next time you try to access the feature that requires the permission unless you check ”Don’t ask me again.” If you deny the permission, you will be shown a dialog explaining why the permission is needed.

The only permission that is required for using Collect is the storage permission and it will be requested on first launch on Android 6+. Other permissions may or may not be needed depending on the forms and features used. The permissions that may be requested by Collect are:
### 3.2. Connecting to a Server

<table>
<thead>
<tr>
<th>Permission</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage</td>
<td>required for Collect to be able to access and save form data</td>
</tr>
<tr>
<td>Camera</td>
<td>required by <em>image</em> and <em>video</em> questions to capture new media</td>
</tr>
<tr>
<td>Contacts</td>
<td>required to configure a Google account for Google Drive, Google Sheets submissions</td>
</tr>
<tr>
<td>Location</td>
<td>required for <em>location</em> questions</td>
</tr>
<tr>
<td>Microphone</td>
<td>required by <em>audio</em> and <em>video</em> questions to capture new media</td>
</tr>
<tr>
<td>Phone</td>
<td>optional on form send to include deviceID in the submission and required for forms that capture <em>device metadata</em></td>
</tr>
</tbody>
</table>

**Changing permissions from settings in Android 6+**

If you are setting up Collect for someone who is not familiar with Android, you may want to make sure all required permissions are granted in advance. To do this, open the Settings application and tap on the Apps heading. You should now see a list of all your installed applications. Scroll down to find ODK Collect and tap on it to see details about Collect. Tap on the Permissions heading. You can now grant all of the permissions that will be needed.

You can also use Settings application to grant a permission after denying it or to revoke a previously-granted permission.

### 3.2 Connecting to a Server

ODK Collect is used to fill forms with participants. Filled forms then need to be sent to a central location for review and analysis. Generally, organizations do this by configuring Collect to send forms to a server. For those working in environments without any internet connectivity, there are other options.

We recommend using ODK Central as your server and configuring Collect by QR code. Central provides user and project management features as well as tools for viewing and exporting data. The easiest way to get a Central server is by using ODK Cloud. If you have technical skills, you can also self-host on your own infrastructure.

If you’d just like to try out ODK Collect without setting up a server, you can use the demo server which provides some sample forms. You can set this up by tapping *Try a demo* at the bottom of the screen when you first launch Collect. The demo project can later be deleted.

If you’d like to connect Collect to more than one server (or to the same server using different users) you can add a new Project for each server (or user).
3.2.1 Configure server from QR code

1. When you first launch Collect, tap on *Configure with QR code*.
2. Position the QR code in the center of the camera field, under the red line. When the camera focuses on the code, it will beep and scan the code.
3. Collect will apply the settings from the code and move you to the Main Menu.

See also:
- *Setting Up ODK Central*
- *Central App Users*
- *Settings QR codes*

3.2.2 Configure server manually

If you’re not using a server that uses QR codes you can still connect using a URL (and username/password if required):

1. When you first launch Collect, tap on *Manually enter project details*.
2. Enter the URL (and username/password if required) for your server.
3. Click *Add*.

3.2.3 Other options

Connecting to a Google Drive Account

**Warning:** Google Sheets have a cell limit (currently 5 million) above which errors will be reported.

1. Add the Google account you want to use to manage forms and submissions for Collect to your device if you haven’t already
2. When you first launch Collect, tap on *Manually enter project details*
3. Tap on *Configure* next to *My project uses Google Drive*
4. Select your Google account
3.2. Connecting to a Server

Setting a Fallback Submission URL

You can set a URL for a Sheet that submissions should be sent to if their form does not specify a submission URL:

1. Tap the *Project* icon on the Main Menu
2. Tap *Settings* and then *Server*
3. Tap *Fallback submission URL*, type or paste the URL for the Sheet in and then tap *OK* to save

Using Android Debug Bridge with Collect

Android Debug Bridge is a tool that provides command-line access to a USB-connected Android device. It can be used to move files between the device and your computer, install applications, and take screenshots and videos. When connected to a device that has ODK Collect installed, it can be used to push blank form definitions and pull completed forms.

Installing and setting up ADB

Android Studio

The easiest and most well-supported way to install ADB is to install *Android Studio*, which includes ADB. After installing, you’ll need to enable USB Debugging.

Standalone ADB

You can also install ADB without *Android studio*. This is not well supported, though, and should only be done if you cannot install Android Studio on your computer.

Managing forms with ADB

Form definitions are stored on the device in the *forms* subdirectory of *your Collect directory*.

Loading blank forms

```sh
$ adb push local/path/to/form.xml <collect-directory>/forms/form.xml
```
Note: The target path on the phone (the last part of the command) must include the file name.

**Deleting forms**

$ adb shell rm -d <collect-directory>/forms/form.xml

**Downloading forms to your computer**

To download all filled records from a device:

$ adb pull <collect-directory>/instances/*

**Developer tasks and troubleshooting with ADB**

**Downloading Collect databases**

Collect stores form definition and form record state information in a few SQLite databases, which you can pull onto your local computer.

$ adb pull <collect-directory>/metadata/*.db

**Taking screenshots**

$ adb exec-out screencap /sdcard/image-name.png

To pull the saved image locally:

$ adb pull /sdcard/image-name.png

Note: ODK Docs contributors can use the `screenshot utility script`, which wraps the `adb` commands and assists with saving the images to the correct location and inserting appropriate markup in the documentation source.
## 3.2. Connecting to a Server

### Recording video

```bash
$ adb shell screenrecord /sdcard/video-name.mp4
```

When you hit `Enter`, the video starts recording immediately.

To stop the recording, press `CTRL-C`. If you don’t interrupt the recording, it will stop after three minutes.

To pull the video locally:

```bash
$ adb pull /sdcard/video-name.png
```

### Capturing logs for debugging

#### Capturing in-progress logs with logcat

If you are experiencing crashes or other serious glitches, and you can reproduce the bug, use `adb logcat` to capture log events during the crash.

1. Before reproducing the bug, begin piping logs to a file:

   ```bash
   adb logcat > logfile.txt
   ```

   This will write all logged errors to your local file `logfile.txt` as they occur.

2. Reproduce the bug or crash event.

3. Type `CTRL-C` to stop logging.

You can then upload the `logfile.txt` file to a support forum post or post in the ODK Forum.

### Pull a bug report

If more in-depth information is needed, you can pull a complete bug report from the device.

```bash
adb bugreport
```

This copies a ZIP file locally containing all system messages, error logs, and diagnostic output, along with information about the device’s hardware, firmware, and operating system.

**See also:**

Identifying the Collect directory on your device

The ODK Collect directory location on your device depends on which version of Collect you have:

- \( \leq v1.26.0 \): /sdcard/odk. Was available to other applications to integrate, but as of August 2020, Google no longer allows globally accessible storage.


- \( \geq 2021.2 \): /sdcard/Android/data/org.odk.collect.android/files/projects. Only accessible by Collect. The Project directories will contain a blank file with the same name as the Project itself.

Using ODK Briefcase

Pulling forms

To pull blank forms and submissions:

1. Open the Pull tab.

2. Select a pull source option from the Pull from drop-down, and click on the Configure button. Fill in any information needed to use the selected source.

   See more on the various sources below.

3. Select the forms you want to pull and click Pull.

   You can see the details of the operation by clicking on the button.

4. You can cancel an ongoing pull operation at any point by clicking Cancel.

Central server

Briefcase will ask for the following information when choosing a Central server as the pull source:

- A URL
- A Project ID number
- An Email address
- A Password
3.2. Connecting to a Server

**Warning:** The *Start pull from last submission pulled* setting will have no effect while pulling forms from a Central server.

**Tip:** See *Using Central with Briefcase* for more on the limitations of using Briefcase with Central.

**Aggregate server**

Briefcase will ask for the following information when setting an Aggregate server as the pull source:

- A *URL*
- A *Username* (optional)
- A *Password* (optional)

**Collect directory**

Briefcase will ask for the directory on your computer where you have placed Collect’s `/odk` directory. We recommend following these steps to get a copy of Collect’s `/odk` directory into your computer:

1. Ensure all filled-in forms are finalized.
   
   If you have incomplete forms that you cannot finalize before pulling into Briefcase, delete them. If you need to keep them, make a copy of your *Collect directory* before deleting them, and restore it after you are finished.

2. Using your device, create a zip archive of your *Collect directory* with a file managing app such as *OI File Manager*.

3. Transfer the zip file to your local hard drive via a USB cable. You can also use the Share feature in your file manager to transfer it to a third-party service like Google Drive then download it to your local hard drive.

4. Once the zip file is on your local hard drive, unzip the file.

**Warning:** When pulling from Collect, Briefcase pulls incomplete, saved, or finalized forms. After you pull forms into Briefcase, it is important that you delete them from Collect. Otherwise, the next time you pull, you will create duplicates.
Tip: If you have Android developer tools installed, another option is to use the Android Debug Bridge to pull filled forms:

$ adb pull <collect-directory>/instances

**Form definition**

Briefcase will ask for the location of the blank form definition in your computer.

**Warning:** Ensure that all attached media is available relative to the form definition file location.

Tip: This enables a workflow to upload blank form definitions with many media attachments to Aggregate:

1. Pull the form using the *Pull from* option.
2. *Push the form to your Aggregate server.*

**Pushing forms**

To push blank forms and submissions:

1. Open the *Push* tab.
2. Select a push target option from the *Push to* drop-down, and click on the *Configure* button. Fill in any information needed to use the selected source.

   See more on the various targets below.

3. Select the forms you want to push and click *Push*.

   You can see the details of the operation by clicking on the button.

4. You can cancel an ongoing push operation at any point by clicking *Cancel.*
3.2. Connecting to a Server

Central server

Briefcase will ask for the following information when using a Central server as the push target:

- A URL
- A Project ID number
- An Email address
- A Password

**Warning:** Pushing forms and submissions to Central currently has the following limitations:

- Central will reject files that might have already been pushed before, even if they’re different the second time.
- Central will reject submissions belonging to a form version that it doesn’t know about.

Aggregate server

Briefcase will ask for the following information when using an Aggregate server as the push source:

- A URL
- A Username (optional)
- A Password (optional)

Pull and push settings

The settings for push and pull can be configured in the Settings tab:

- You can set a number of *Maximum simultaneous HTTP connections*. This can be increased to speed-up big pull operations or decreased to prevent saturating server bandwidth.

- You can enable *Start pull from last submission pulled* to save time and bandwidth by not pulling from the first submission.
  - This is only available for Aggregate servers at this moment and only benefits forms with more than 100 submissions.
– You can clear the pull history and pull every submission by clicking on *Clear pull history*.

• You can enable *Remember passwords (unencrypted)*. This will enable a couple of features:
  – Briefcase will remember the pull sources and push targets you configure when they require user credentials. As a result, you won’t need to configure them when launching Briefcase again.
  – Briefcase will let you enable the *Pull before export* option when exporting forms.

• You can enable *Use HTTP proxy* to route your HTTP requests through a proxy host. You will have to provide the proxy’s *Host* (IP address or hostname), and the *Port* number.

**Export forms to CSV**

1. Open the *Export* tab.

2. Click on the *Set Default Configuration* button.

• Set an *Export directory*.

• If exporting *Encrypted Forms*, set the corresponding *PEM file location*. See the *Encrypted forms section* for more information.

• If you wish, select a *Start date* and an *End date* to specify a limited date range to export.

• Toggle export parameters as needed:
  – *Export media files* enables exporting media files into the chosen export directory
  – *Overwrite existing files* enables overwriting form submission data in the output files. The default behavior is to append data.
  – *Split select multiples* enables splitting select multiple fields. Enabling this setting will create an extra output column per select choice, with a 1 if the choice was selected, or 0 otherwise. This only affects select fields without a choice filter and that are not from an external file (including widgets with search appearance).
  – *Include GeoJSON* enables generating a GeoJSON file with spatial data from all exported submissions.
  – *Remove group names* enables removing non-repeat group names from column names in the CSV.
  – *Pull before export* enables trying to pull the selected forms in case there are new form submissions to be exported.
3.2. Connecting to a Server

1. Select the forms to export.
   
   If you are selecting and exporting more than one form, you may need to set individual export settings. To do this, click the gear icon ( ) next to the form name.

2. Click Export.

**Tip:** To import CSVs into Excel, you cannot download and open in one step; nor can you double-click on the CSV. You must open Excel and choose Import. If you are asked, the file origin or encoding is UTF-8.

**Output files**

Briefcase will generate a different number of files and directories depending on the form’s contents and the export configuration selected by the user. This can include, per form:

- One main CSV file. For example: *Form Name.csv*
- If the form includes any repeat group, one CSV file for each one of them. For example: *Form Name-repeat group name.csv*
- If any submission includes binary attachments, they are copied to a *media* directory, relative to the export directory. For example: *media/1538040007350.jpg*
- If the user enables the Include GeoJSON export configuration option, one GeoJSON file with spatial data. For example: *Form Name.geojson*
- If the form includes audit metadata:
  - One CSV file with audit data from all submissions. For example: *Form Name-audit.csv*
  - One CSV audit file for each exported submission in the *media* directory, relative to the export directory. For example: *media/audit-uuid56880d5e-ee8a-4832-b69d-6dfdd526e2dc.csv*
Table 3.1: Summary Table

<table>
<thead>
<tr>
<th>Output file</th>
<th>How many?</th>
<th>Conditions</th>
<th>Path</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main CSV</td>
<td>One</td>
<td></td>
<td>./</td>
<td>Form Name.csv</td>
</tr>
<tr>
<td>Repeat CSV</td>
<td>One per repeat group</td>
<td></td>
<td>./</td>
<td>Form Name-repeat group name.csv</td>
</tr>
<tr>
<td>Binary attachment</td>
<td>As many as there are in submissions</td>
<td></td>
<td>./media</td>
<td>media/1538040007350.jpg</td>
</tr>
<tr>
<td>GeoJSON</td>
<td>One</td>
<td>The user enables Include GeoJSON export</td>
<td>./</td>
<td>Form Name.geojson</td>
</tr>
<tr>
<td>Audit CSV</td>
<td>One</td>
<td>The form includes audit metadata</td>
<td>./</td>
<td>Form Name - audit.csv</td>
</tr>
<tr>
<td>Individual audit CSV</td>
<td>One per submission</td>
<td>The form includes audit metadata</td>
<td>./media</td>
<td>audit-uuid56880d5e-ee8a-4832-b69d-6dfdd526e2dc.csv</td>
</tr>
</tbody>
</table>

There’s more information available about the CSV file content structure and filename patterns in the export format documentation.

Working with the command line

Briefcase has a command line interface (CLI) to enable scripting of many of the actions that can be taken in the graphical user interface (GUI).

New in version 1.4.4: A CLI was added.

New in version 1.9.0: The CLI first takes an operation parameter and then modifiers to that operation

Getting CLI help

To get help about the command line operation:

```
$ java -jar {path/to/briefcase-jar-file} --help
```

Pulling forms from Aggregate

- CLI flag: -plla or -pull_aggregate
- Usage:
3.2. Connecting to a Server

$ java -jar {path/to/briefcase-jar-file} --pull_aggregate --storage_directory {path/to/briefcase-storage-location} --aggregate_url {aggregate-url} --odk_username {username} --odk_password {password}

- **Help section:**

<table>
<thead>
<tr>
<th>Params for -plla operation:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-p,--odk_password &lt;arg&gt;</td>
<td>ODK Password</td>
</tr>
<tr>
<td>-sd,--storage_directory &lt;arg&gt;</td>
<td>Briefcase storage directory</td>
</tr>
<tr>
<td>-u,--odk_username &lt;arg&gt;</td>
<td>ODK Username</td>
</tr>
<tr>
<td>-url,--aggregate_url &lt;arg&gt;</td>
<td>Aggregate server URL</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Optional params for -plla operation:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-id,--form_id &lt;arg&gt;</td>
<td>Form ID</td>
</tr>
<tr>
<td>-ii,--include_incomplete</td>
<td>Include incomplete submissions</td>
</tr>
<tr>
<td>-mhc,--max_http_connections &lt;arg&gt;</td>
<td>Maximum simultaneous HTTP connections (defaults to 8)</td>
</tr>
<tr>
<td>-sfd,--start_from_date &lt;arg&gt;</td>
<td>Start pull from date</td>
</tr>
<tr>
<td>-sfl,--start_from_last</td>
<td>Start pull from last submission</td>
</tr>
</tbody>
</table>

**Warning:** This CLI operation will pull all forms Briefcase has permissions to if no -id parameter is defined.

### Pulling forms from Collect

This command assumes you have already copied and unzipped the odk file as described here.

- **CLI flag:** -pc or -pull_collect
- **Usage:**

$ java -jar {path/to/briefcase-jar-file} --pull_collect --storage_directory {path/to/briefcase-storage-location} --odk_directory {path/to/unzipped-odk-file}

- **Help section:**

<table>
<thead>
<tr>
<th>Params for -pc operation:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-od,--odk_directory &lt;arg&gt;</td>
<td>ODK directory</td>
</tr>
<tr>
<td>-sd,--storage_directory &lt;arg&gt;</td>
<td>Briefcase storage directory</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Optional params for -pc operation:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-id,--form_id &lt;arg&gt;</td>
<td>Form ID</td>
</tr>
</tbody>
</table>
Warning: This CLI operation will pull all forms present on the odk directory if no -id parameter is defined.

Pushing forms to Aggregate

- CLI flag: -psha or -push_aggregate
- Usage:

  $ java -jar {path/to/briefcase-jar-file} --push_aggregate --form_id {form-id} --storage_directory {path/to/briefcase-storage-location} --aggregate_url {aggregate-url} --odk_username {username} --odk_password {password}

- Help section:

<table>
<thead>
<tr>
<th>Params for -psha operation:</th>
</tr>
</thead>
<tbody>
<tr>
<td>-id,--form_id &lt;arg&gt;</td>
</tr>
<tr>
<td>-p,--odk_password &lt;arg&gt;</td>
</tr>
<tr>
<td>-sd,--storage_directory &lt;arg&gt;</td>
</tr>
<tr>
<td>-u,--odk_username &lt;arg&gt;</td>
</tr>
<tr>
<td>-url,--aggregate_url &lt;arg&gt;</td>
</tr>
<tr>
<td>Optional params for -psha operation:</td>
</tr>
<tr>
<td>-fsb,--force_send_blank</td>
</tr>
<tr>
<td>-mhc,--max_http_connections &lt;arg&gt;</td>
</tr>
</tbody>
</table>

Warning: This CLI operation will only update the blank form if it does not already exist, whereas the GUI will always update the form.

Exporting forms to CSV

- CLI flag: -e or -export
- Usage:

  $ java -jar {path/to/briefcase-jar-file} --export --form_id {form-id} --storage_directory {path/to/briefcase-storage-location} --export_directory {path/to/output-directory} --export_filename {output-file-name.csv}

- Help section:
3.2. Connecting to a Server

<table>
<thead>
<tr>
<th>Params for -e operation:</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-ed,--export_directory &lt;arg&gt;</td>
<td>Export directory</td>
</tr>
<tr>
<td>-f,--export_filename &lt;arg&gt;</td>
<td>Filename for export operation</td>
</tr>
<tr>
<td>-id,--form_id &lt;arg&gt;</td>
<td>Form ID</td>
</tr>
<tr>
<td>-sd,--storage_directory &lt;arg&gt;</td>
<td>Briefcase storage directory</td>
</tr>
<tr>
<td>Optional params for -e operation:</td>
<td></td>
</tr>
<tr>
<td>-em,--exclude_media_export</td>
<td>Exclude media in export</td>
</tr>
<tr>
<td>-end,--export_end_date &lt;arg&gt;</td>
<td>Export end date (inclusive) (yyyy-MM-dd or yyyy/MM/dd)</td>
</tr>
<tr>
<td>-ig,--include_geojson &lt;arg&gt;</td>
<td>Include a GeoJSON file with spatial data</td>
</tr>
<tr>
<td>-oc,--overwrite_csv_export</td>
<td>Overwrite files during export</td>
</tr>
<tr>
<td>-pb,--pull_before</td>
<td>Pull before export</td>
</tr>
<tr>
<td>-pf,--pem_file &lt;arg&gt;</td>
<td>PEM file for form decryption</td>
</tr>
<tr>
<td>-rgn,--remove_group_names</td>
<td>Remove group names from column names</td>
</tr>
<tr>
<td>-ssm,--split_select_multiples</td>
<td>Split select multiple fields</td>
</tr>
<tr>
<td>-start,--export_start_date &lt;arg&gt;</td>
<td>Export start date (inclusive) (yyyy-MM-dd or yyyy/MM/dd)</td>
</tr>
</tbody>
</table>

Clear saved preferences

- CLI flag: -c or --clear_prefs
- Usage:

```
$ java -jar {path/to/briefcase-jar-file} --clear_prefs
```

Briefcase log files

Briefcase creates a log file with warnings and errors that might be useful for troubleshooting.

Default log file location

If something goes wrong while using Briefcase and you look for help, it’s possible that you’re asked to provide your log file.

The default location for the log file is the directory where you are when launching Briefcase, and the default filename is "briefcase.log"

Briefcase will create the log file on launch if it doesn’t previously exist. Otherwise, it will append lines at the end of a pre-existing log file.
How to use a custom log configuration

Optionally, you can use a custom log configuration file to override the default log settings on Briefcase.

First, you need to create a "logback.xml" file somewhere in your computer to contain your custom log configuration. This is a sample configuration file you can use as a template:

```xml
<configuration>
  <appender name="ROLLINGFILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
    <file>briefcase.log</file>
    <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
      <fileNamePattern>briefcase.%d{yyyy-MM-dd}.log</fileNamePattern>
      <maxHistory>30</maxHistory>
      <totalSizeCap>100MB</totalSizeCap>
    </rollingPolicy>
    <encoder>
      <pattern>%d [%thread] %-5level %logger{36} - %msg%n</pattern>
    </encoder>
  </appender>
  <root level="info">
    <appender-ref ref="ROLLINGFILE" />
  </root>
</configuration>
```

Check the full syntax of Logback configuration files here.

You can set all sorts of new log configurations to adapt Briefcase to your needs:

- Set a fixed log file location
- Fine tune the log’s verbosity by setting a different log level
- Silence specific log lines while keeping others
- Set a custom log format (see the Encoders chapter)
- Set custom appenders, to define a file rolling policy (daily, by log file size, for example), for example (see the Appenders chapter)

Once you have your configuration file ready, you can use it by adding a -Dlogging.config argument when launching Briefcase:

```
$ java -Dlogging.config="{path/to/logback.xml}" -jar {path/to/briefcase-jar-file}
```
3.3 Configuring Collect

ODK Collect is intended to be a flexible tool that can support many different types of data collection workflows. For many organizations, the only configuration needed will be connecting to a server. However, there are many cases where additional configuration is helpful. Using Collect Settings describes all of the settings available in the application.

To configure multiple devices, we recommend generating a QR code with settings from one device or an external application as described in Settings QR codes and then scanning that code from the devices that need to be configured.
CHAPTER FOUR

USING ODK COLLECT

This is an in-depth guide to using the Collect App.

To learn about the main features and try out ODK, see the Getting Started Guide for a quick walk-through.

ODK Collect is used by enumerators to collect survey data by filling out forms with participants.

Collect is used to:

1. Get blank forms from a server or Google Drive
2. Fill out surveys with participants
3. Upload completed surveys to a server or Google Drive

ODK Collect User Guide

4.1 Managing Projects in Collect

New in version v2021.2.

Starting with Collect v2021.2, Collect can be set up with multiple Projects. Each Project in Collect has its own set of Forms and Settings.

Some examples of scenarios where having more than one project might be useful:

- An enumerator working on different data collection efforts across different servers
- Multiple enumerators using one device for the same data collection project but with different user accounts on the server
- Form designers testing forms on different staging/development servers

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
Each Project is identified by a name, icon and color. These are shown at the top of the Main Menu:

Tapping the Project icon allows you to add, manage, and switch Projects:
4.1. Managing Projects in Collect

4.1.1 Adding a new Project

1. Tap the Project icon on the upper right of the Main Menu screen
2. Tap Add project
3. Scan a QR code or tap Manually add project details to set the server for the new project

Warning: If a Project is added with the same URL as an existing one, Collect will warn the user before proceeding. This is to prevent users from accidentally adding the same Project multiple times.

4.1.2 Switching between Projects

1. Tap the Project icon on the upper right of the Main Menu screen
2. Select the Project you want to switch to
4.1.3 Deleting a Project

1. Tap the Project icon on the upper right of the Main Menu screen
2. Select the Project you want to delete if it is not already the current Project
3. Tap Settings and then Project Management
4. Tap Delete and confirm that you want to delete the Project

4.2 Managing Forms in Collect

See also:
For an overview on forms and form design, see Intro to Forms in ODK.

4.2.1 Loading Blank Forms

A blank form is a .xml file containing a form definition consistent with the XForm specification. Blank forms can be created using ODK Build or XLSForm.

In order to fill out forms with survey participants, you must first load blank forms into the Collect App.

Loading Forms from a server

If you have connected ODK Collect to a server or Google Drive Account:

1. Select Get blank forms on the app home screen to browse available forms and download them to your device.
2. Find and download forms.
   
   - If you are using a server, you will see a list of available forms. Select the ones you would like to download, and tap *Get Selected*. 
Note: Before downloading blank forms from your server to Collect, a form has to be uploaded. See Form Management in ODK Central.

- If you are using Google Drive, the Get Blank Form screen will display the folders in your Google Drive account and any XML documents. Select and download the forms you want.
4.2. Managing Forms in Collect

Warning: All XML documents in Google Drive will appear in Collect. XML documents will be listed even if they are not valid XForms.
And you can actually download any XML document, even if it isn’t a real XForm.
4.2. Managing Forms in Collect

But you can’t Fill Out a non-form.
**Loading Forms directly**

**Loading forms with adb**

You can load forms directly from a computer to your device’s *Collect directory* via USB, using *Android Debug Bridge*.

```
$ adb push path/to/form.xml <collect-directory>/forms/form.xml
```

**Loading forms from device storage**

You can also download forms to your device via a web browser, and move them to the *forms/* directory, using the device’s file manager (*Settings -> Storage & USB -> Explore*).

1. Go to the Settings menu ( ) on your device and find *Storage & USB*
2. From the internal storage screen, select *Explore* to open the file manager.
Fig. 4.1: The settings menu may look different on your device.
Fig. 4.2: The *Explore* option opens a file manager that you can use to move forms into Collect.
4.2. Managing Forms in Collect

Loading Form media

If a form includes images or other media, those files have to be loaded to the device along with the form.

Media files must be placed in a folder labeled form-name-media.

- When using ODK Central, first upload your form definition. Central will then prompt you to add media files if necessary. The files are downloaded automatically when fetching blank forms.
- When using Google Drive, the -media folder should be uploaded to the same location as the form. If you share forms with another user, you need to share the parent folder which contains a form and a folder with media files. Sharing both of them separately wouldn’t be enough.
- If loading forms directly to your device, the -media folder needs to be placed in the forms subdirectory of your Collect directory, alongside the form itself.

4.2.2 Editing Saved Forms

Completed (filled-in) form instances can be edited after they have been saved.

1. Select Edit Saved Form
2. Select a form by name

This will reopen the form instance, which you are then free to edit. Form instances are listed by name, which is one reason it can be helpful to name the form instance after filling it out.

Note:

- Sent forms will not appear in the Edit Saved Forms list.
- Sent forms will be available for viewing in View Sent Forms list, along with the details which cannot be edited.
- You may freely edit Saved and Finalized forms.

4.2.3 Transferring Finalized Forms

To perform analysis on data collected with the Collect app, you will need to get the filled forms off the devices. Generally, this is done by uploading them to an ODK server or Google Sheets. To do this, you will first need to configure a server.

In some cases, you may want to pull filled forms directly from a device. This can be simpler than setting up a server if you are only using a small number of devices or when there is no Internet access. It can also be helpful to recover from submission failures.
4.2. Managing Forms in Collect

Sending Finalized Forms to a Server

If you are connected to a server or Google Drive Account, use Send Finalized Forms to upload Finalized form instances.

Uploading a filled form from within the Collect app marks that form as sent. Sent forms are no longer editable, but they remain viewable until they are deleted.

Note: Blank values in the form are sent to google sheets as cells with a space and not as empty cells. When you are testing for empty cells, you might not get the correct results. To make sure you get the correct results, you could:

- Use the TRIM function in the google sheets to remove the leading and trailing spaces from the cells.
- Define empty cell in your tests to be a cell that is either empty or contains a single space.

Values from questions: text, select_multiple, and barcode will be preceded by an apostrophe. This prevents Google Sheets from guessing at the data type and applying a format that may not be appropriate (e.g. making 1940 10 5 into a date when it actually is the value of a select multiple). Raw values may be used in any kind of computation and the apostrophes only are visible when editing a value. You may choose to manually apply a format for certain
Chapter 4. Using ODK Collect

columns if desired.

Note: Using Google Drive as a server, filled forms are sent to the first sheet in a given spreadsheet, no matter what its name is. If you use one spreadsheet to keep a form definition and to collect filled forms make sure the sheet you expect to be filled is in the first place.

Sending Previously-Sent Forms

If you can’t find a submission that you expect on your server or need to re-send a submission for other reasons, you can change the view of the Send Finalized Forms screen to show both sent and unsent forms.

To show sent and unsent forms: → Change View → Show Sent and Unsent Forms

Pulling Forms into Briefcase

ODK Briefcase is a desktop application that can be used to pull filled forms to your local computer. You will first need to transfer the filled forms to your computer. This will not update the state of the form to Sent.

4.2.4 Deleting Forms

Warning: In versions prior to v1.28, deleting a blank form makes it impossible to edit filled instances of that form.

In Collect v1.28 and later, filled instances of forms will still be editable after their blank form is deleted. This means that the form definition and media files will remain on the device until all the filled instances have been deleted.
4.2. Managing Forms in Collect

You can delete Blank forms as well as filled forms in any state (Saved, Finalized, or Sent). Deleting a Sent form deletes the form contents but metadata associated with it including the deletion date and the instance name are maintained for display in the View Sent Form list.

1. Select Delete Saved Form on the app home screen.

![ODK Collect v1.11.1 Main Menu](image)

2. Select the Saved Forms or Blank Forms tab.

Note: Deleted Forms are listed in the View Sent Forms page, but cannot be viewed. They are indicated with the crossed-out eye icon.
Fig. 4.3: The *Saved Forms* tab lists form instances that are saved, finalized, or sent.
4.2. Managing Forms in Collect

Fig. 4.4: The **Blank Forms** tab lists blank forms.
Deleting Forms with adb

You can also delete form instances directly with Android Debug Bridge. They are stored in the instances subdirectory of your Collect directory, with a directory for each instance.

4.3 Filling out Forms with Collect

ODK Collect is used by enumerators to complete survey forms with participants.

Before you get started...

Before you can fill out a form, you need to load at least one form into Collect.

1. Select Fill Blank Form from the Main Menu
4.3. Filling out Forms with Collect

2. Select a form to fill out from the form list
3. Once you have completed the form, prepare it for upload by finalizing it.

### 4.3.1 Answering questions

**Free response**

Free-entry text and number answers are entered using the device keyboard. The appropriate keyboard (letters or numbers) opens when the question appears.

Video showing text keyboard popup when a string input is required and number keyboard popup when a number input is required.

**Select response**

Questions with response choices can be answered by touching the selected items. These include radio buttons (single-select), dropdowns (single-select), check boxes (multi-select), and image choices (single and multi-select).

![Radio buttons accept one selection.](image)

Fig. 4.5: Radio buttons accept one selection.
4.3. Filling out Forms with Collect

Fig. 4.6: Check boxes accept multiple answers.
Fig. 4.7: Select images by touching them.
4.3. Filling out Forms with Collect

**Capture answer from device**

Many question formats launch additional features, form widgets, or apps to collect an answer. This includes audio and video recording, signature collection, photo capture, date and time widgets, location widgets, and barcodes.

In all cases, buttons below the question text will guide you through providing the response.
4.3. Filling out Forms with Collect

See also:

For a (mostly) complete guide to form question appearance, see Question Types.

4.3.2 Removing answers

To remove a response, Long Press on the question label.
4.3.3 Adding instances of repeats

If you have a repeating group, you can add new instances of that repeat in the following ways:

1. By *navigating* into an empty repeat, or to the next question at the very end of the repeat, you will automatically be prompted to add a new instance of that repeat.

2. By clicking the "add" button while in a repeat:
3. By clicking the “add” button in the jump menu:
4.3.4 Removing instances of repeats

If you have a repeating group, you can remove existing instances of that repeat in the following ways:

1. By long pressing on the question label in the same way as for removing answers
2. By clicking the "remove" button in the jump menu:

![Image of a form with a repeating group and a remove button]

4.3.5 Navigating the form

Note: Since Collect v1.29, both swiping and button navigation are enabled by default on new installations. Prior to Collect v1.29 or for existing installations, only swiping was enabled by default.

Swipe

To move between questions, Swipe Left or Right.
4.3. Filling out Forms with Collect

Next and Back Buttons

If you prefer Next and Back buttons for navigation, you can change your navigation mode in -> Project settings -> User Interface.

1. Open the Action Menu ( )
2. Select *Project settings.*
3. Select *User Interface*

4. Select *Navigation*
5. Update your form navigation preference
Jumping to questions

The arrow icon ( ) in the top right corner opens the jump menu. From the jump menu, you can go to any question or go to the beginning/ending of the form.
If you’re inside of a group of questions, you can navigate “up” in the hierarchy using the “go up” button:
4.3. Filling out Forms with Collect

The jump menu also provides shortcuts to add or remove instances of repeating groups.

**Note:** If a form contains questions in a repeats, those questions will only appear in the Jump menu once an actual record is created.

**Autoadvance Questions**

Some questions will *automatically advance to the next question* after being answered.

Video showing auto-advance after the questions are answered.

**Required Questions**

Required questions will not allow you to advance unless answered.
4.3.6 Changing language of a form

If a form is available in multiple languages, you can choose a language in which you want the questions to appear.

1. Open the Action Menu ( )
2. Select *Change Language*. 
Note: Change Language option is only visible if a form is available in more than one language.

3. Select the language you want the form questions to appear in.

4.3.7 Saving a partially filled form

If you wish to save a partially filled form, you can click on the save icon (_FACTORY ) beside the form name.
To *edit the saved form*, select *Edit Saved Form* in the Main Menu and select the form you wish to edit.

### 4.3.8 Completing a Form

Once you have reached the end of a form, you will have the opportunity to *Save* and *Exit* the form.
At this point, you may also:

**Name the form**

The last form screen provides a default name for the form (defined by the form designer). You can rename it. This name only applies to that particular instance of a completed form (not to the blank form).
4.3. Filling out Forms with Collect

The Form Name identifies the form in lists throughout the app. For this reason, a meaningful name may be important to you. After you’ve saved the name, the form automatically moves to the Send Finalized Form section, from where you can send it.
Mark the form as *Finalized*

Only Finalized forms can be *uploaded to a server*.

### 4.4 Using Collect Settings

Collect has many settings to support a number of different data collection workflows. Settings are specific to each *project* and are separated into *general* and *protected* settings. Protected settings make it possible to remove access to certain parts of Collect and can be *password-protected*.

**Settings for the current project** are accessed from the project list dialog:

*Project Icon → Settings*
4.4. Using Collect Settings

4.4.1 General Settings

Server Settings

Server settings configure the connection to an OpenRosa server (Central, etc) or a Google Drive account.

To access Server Settings: Project Icon → Settings → Server
See also:

*Connecting to a Server*

**Project Display Settings**

Project display settings control the name of the current project as well as the appearance of its icon in the upper right of the main screen.

To access Project Display Settings: *Project Icon → Settings → Project Display*
4.4. Using Collect Settings

See also:

*Managing Projects in Collect*

**User Interface Settings**

User Interface settings control Collect’s appearance and behavior.

To access User Interface settings: *Project Icon → Settings → User interface*
Chapter 4. Using ODK Collect

**Theme**

Selects the color scheme the app will use. The default *Use device theme* option (added in ODK Collect v2021.3) uses the device’s color theme (based on the the *Dark theme* system setting in Android 10 and above).
4.4. Using Collect Settings

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
Chapter 4. Using ODK Collect

**Language**

Forces the Collect interface to use a specific language. By default, Collect matches the device language. Note that this only sets the language for the Collect user interface and not for the form. For forms with multiple languages, the form language is set while filling out the form.

*Note:* Collect’s translations are provided by the ODK community through the Transifex service. You can join Transifex to add or correct translations in your language.

---

**Text font size**

Sets the size of fonts used in the form-filling interface.

**Navigation**

Sets form navigation style for moving between questions.

Options:

- Horizontal swiping
- Forward and back buttons
- Both

**Splash Screen**

Sets an image to display while Collect loads.

**Maps Settings**

Maps settings configure the maps shown by the location question types.

To access Maps settings: Project Icon → Settings → Maps

*Note:* Prior to ODK Collect v1.23, map settings were available in the User Interface Settings. The basemap was configured by first selecting a Mapping SDK and then a Basemap.
4.4. Using Collect Settings

Basemap settings

Basemap settings configure the background of maps shown by the location question types. Basemaps are provided by several different Sources which may each make several different map Styles available. A basemap is intended to provide details that help users orient a map and to make the map easy to use in a particular data collection environment. For example, if the data to be collected relates to elevation, consider selecting a topographic basemap.

Sources A basemap source provides one or more map styles:

- Google basemap styles are used by Google Maps and other Google products.
- Mapbox basemap styles are used in many familiar products.
- OpenStreetMap provides one style which also powers openstreetmap.org. OpenStreetMap data is used in basemaps provided by all other sources as well.
- USGS is the United States Geological Survey. It provides topographic and satellite basemaps for the United States only.
- Stamen provides a terrain basemap with large labels.
- Carto basemap styles are designed to be used with data layers.

Reference layer settings

Reference layer settings configure map data shown on top of the basemap. Currently, a reference layer can only be defined by an offline MBTiles file as described in Using Offline Maps. The reference layer will appear when the zoom level is within the range supported by the file. If a reference layer has no transparency, it will fully cover the basemap selected above and behave like an offline basemap. Vector MBTiles files will only be available in the Layer data file menu if a Mapbox basemap is selected. Raster MBTiles files will be available for any basemap source and style.

Form Management Settings

Form Management settings control default behavior when editing, finalizing, and importing forms.

To access Form Management settings: Project Icon → Settings → Form management
Chapter 4. Using ODK Collect

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
4.4. Using Collect Settings

Form update

**Blank form update mode**

Specifies how blank forms should be updated:

- **Manual** The default mode in Collect. Enumerators manually manage blank forms on the device using *Get Blank Form* and *Delete Saved Form*.

- **Previously downloaded forms only** Enumerators will receive a notification when one or more forms on the device have an update available to their form definition or media files. Tapping on the notification will go to *Get Blank Form* where the user can choose to download some or all of the updated forms.

- **Exactly match server** Collect will automatically download and update forms based on what’s on the server. In addition, forms not on the server will be deleted from Collect. This mode hides *Get Blank Form* and the *Blank Forms* tab in *Delete Saved Form* as they are not required. The enumerator can trigger an update from the server on the *Fill Blank Form* screen. Filled instances of blank forms deleted during server updates will still be editable.

**Tip:** If your server is configured to provide the exact set of forms enumerators need and you’d like to ensure they always have the most up to date versions on device then we’d recommend using *Exactly match server*.

However, if your server is set up to provide forms that aren’t relevant to every enumerator then we’d recommend using *Previously downloaded forms only* (ideally with *Automatic download*) so that enumerators are still notified when the forms they do use are updated.

*Manually* makes the most sense when forms only need to be downloaded once and will never change or if you are extremely bandwidth-limited.

**Automatic update frequency**

Specifies how frequently Collect should check for updates to the forms on the server when using *Previously downloaded forms only* or *Exactly match server*. This option is not available if *Manually* is selected.

**Automatic download**

Only available if *Previously downloaded forms only* is selected. When *Automatic download* is enabled, the form update check will trigger an automatic download
of any forms on the device that have updated definitions or media files. The user will receive a notification when the automatic download completes with either a success or failure. Tapping on the notification will go to Get Blank Form where the user will see success or failure messages for each form for which an update was attempted.

*Hide old form versions*

When enabled, if there are multiple versions of the same form, only the most recently downloaded will be displayed on the Fill Blank Form screen.

*Form submission*

*Auto send*

When enabled, forms are sent immediately when they are finalized, if the device can connect to the internet. If an internet connection is not available at the time of finalization, your finalized forms will be queued to send as soon as connectivity is established. You can specify whether to send over WiFi, cellular data, or both.

*Delete after send*

When enabled, form instances are deleted once they are sent.

*Form filling*

*Default to finalized*

When enabled, records are set to be finalized when saved at the end of a form-filling session. You can opt out of this at the end of filling any specific record. This is particularly important to consider when using encrypted forms because encryption happens on finalization. Finalized records for encrypted forms can’t be opened because they are encrypted. Records for encrypted forms that have not been finalized are not encrypted and can be edited.

*Constraint processing*

Sets when form responses are validated against constraints.

Options:
4.4. Using Collect Settings

- Upon forward swipe. (That is, right after the question is answered.)
- At finalization.

**High res video**

When enabled, *Video widgets* widgets will record high resolution video if possible.

**Image size**

New in version 1.11.0.

Sets the default maximum size for images added to forms, as measured by the number of pixels on the longest edge. Images larger than the maximum are scaled down immediately after being added.

Options:

*Original size from camera (default)* Images are unchanged when added to a form. Recommended for use only when images must contain a lot of detail and when the internet connection used to send submissions is fast.

*Very small (640px)* Recommended when images don’t need to be detailed or the internet connection used to send submissions is slow.

*Small (1024px)* Sufficiently detailed for most on-screen viewing but too small for printing.

*Medium (2048px)* Sufficiently detailed for most uses, including printing.

*Large (3072px)* Recommended when a lot of detail is needed, but you want to reduce the size of image files as much as possible.

**Show guidance for questions**

Guidance hints on questions can be used to display additional information that is not always needed. For example, they can be used to show extra instructions to be used during training or valuable only on a printout. If set to *Yes - always shown*, guidance hints will always be displayed below regular hints. If set to *Yes - collapsed*, the user will need to tap to view guidance hints.

**Use external app for audio recording**

By default, an external application is used for audio recording. Uncheck this setting to use the built-in audio recorder instead. When unchecked, recordings
will be created as mono .m4a files using the AAC codec with a sample rate of 32kHz and a bitrate of 64kbps. This corresponds to a file size of about 30MB/hour. We typically recommend configuring audio quality in the form definition instead of using this setting but it can be useful for older forms that can’t be modified.

Form import

Finalize forms on import

When enabled, forms added directly to the instances/ directory are automatically set to Finalized. This is particularly relevant when putting records for an encrypted form directly to the device because encryption happens on finalization.

User and Device Identity Settings

User and device identity settings control how personally identifiable information and device ID are used.

To access User and device identity settings: Project Icon → Settings → User and device identity
4.4. Using Collect Settings

Form metadata settings

Form metadata settings control identifying information \textit{added to forms} filled on the device.

\textbf{To access form metadata settings:} Project Icon $\rightarrow$ Settings $\rightarrow$ User and Device Identity $\rightarrow$ Form Metadata

\begin{itemize}
  \item User-defined
  \item Username
  \item Phone number
  \item Email address
  \item Device-defined
    \item Device ID
    \item Subscriber ID
    \item SIM serial number
\end{itemize}

\textbf{User-defined}

You can edit the following:

\begin{itemize}
  \item Username
  \item Phone number
  \item Email address
\end{itemize}

\textbf{Note:}

\begin{itemize}
  \item If no username is set here, the username from \textit{Server settings} is used instead.
  \item You can restrict editing of the username in \textit{Protected settings}.
\end{itemize}
Device-defined

You cannot edit these:

- Device ID
- Subscriber ID
- SIM serial number
- Install ID

Device ID is currently set to the device IMEI. Starting in August 2020, Google will no longer allow Android applications to read the IMEI. At that time, the Collect-generated Install ID will be used as the Device ID. Both are currently displayed to allow organizations to transition over. Install ID can be copied by long-pressing on its text.

Usage data

When enabled, ODK Collect sends anonymous usage and error data back to the ODK development team, which helps us improve the application.

4.4.2 Protected Settings

Protected settings manage other settings and features, letting you import or export settings, reset settings and delete cached data, and restrict which features are available to users of the app.

Protected settings are useful when you would like to limit the options available to enumerators so that they must follow a specific workflow.

You can password protect the protected settings, so enumerators cannot adjust settings or access restricted features.

Set admin password If a password is set, when settings are opened, the Protected section will only contain Unlock protected settings. Tapping on that will display a dialog to provide the admin password. Before the correct admin password is provided, access controls will be in place and some settings may be hidden. Saving a blank password disables password protection.

Project management settings

Reconfigure with QR code Replace all settings from those in a QR code. See configuring Collect via QR code.
4.4. Using Collect Settings

*Reset* Reset to default settings, delete forms, and empty caches. There is a prompt to select which aspects of the project to reset.

*Delete* Delete the current project.

**Access control settings**

*Main Menu Settings* Displays a list of buttons shown on the main screen. To prevent access to certain features, uncheck them and their button will be hidden.

*User Settings* Displays a list of user settings and other features accessible in the settings screen. To hide features, uncheck them.

*Form Entry Settings* Displays a list of features related to viewing and filling out forms. To disable features, uncheck them.

*Moving backwards* If you disable moving backwards, the enumerator cannot use the back button or swipe right to move backwards through a form.

However, disabling this feature does not completely restrict a user’s ability to access already-answered questions. So, when you uncheck this box to restrict backward movement, the app will suggest several additional restrictions which will prevent a non-admin user from revisiting already-asked questions:

- Disable *Edit Saved Form* option in the main menu
- Disable *Save Form* option in the Form entry menu
- Disable *Go To Prompt* option in the Form entry menu
- Set *Constraint processing* to validate upon forward swipe in the Form Management settings

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
Select **YES** to set these additional restrictions.

**Note:** When you enable the moving backwards option, you have to configure the other changed settings since they are not automatically changed back.

### 4.5 Settings QR codes

Collect’s settings can be imported and exported using QR codes. These can be provided by servers (like ODK Central) or can be used to share settings from one device to another.

#### 4.5.1 Scanning a QR code

Settings QR codes can be scanned when adding a new Project or can be scanned to reconfigure the current Project by navigating to *Reconfigure with QR code* in *Project Management* settings.

**Note:** QR codes contain settings with non-default values. When a code is scanned in, settings not explicitly included in the code are reset to their default values.
4.5. **Settings QR codes**

**Importing settings from an image saved on your device**

You can import settings from a QR code saved on your device by clicking on the icon when scanning a QR code. Then select the *Import QRCode* option.

### 4.5.2 Sharing settings with another device

Navigate to *Reconfigure with QR code* in *Project Management* settings and tap *QR Code* to display the current Project’s settings QR code. This can then be scanned by another device to either add a new Project or reconfigure the current one.

**Warning:** Settings QR codes contain the admin and server passwords *in plain text*. To remove them from the code, tap the warning on the QR code screen.

**Sharing a QR code as an image**

You can tap on the icon to share the QR code as an image. When you click on it, it displays a list of applications and services like *whatsapp*, *facebook*, *hangouts*, *bluetooth*, *MMS* which can be used to share the QR code. This is useful when there are several different data collection sites and all devices have to be configured in the same way, in which case the QR code can be shared from one reference device.

**Warning:** Since the QR code encodes compressed plain text settings and may contain the admin and server passwords, you should be careful about how you share it. For example, if you print out the QR code and tape it on wall, someone could use a standard QR code scanner to get the admin password.

### 4.5.3 Making your own QR code

The contents of a settings QR code is a JSON object with objects for unprotected (“general”) and protected (“admin”) settings, as well as the project details themselves. The JSON structure is the following:

```json
{
    "general": {
        "server_url": "https://demo.getodk.org",
        "constraint_behavior": "on_finalize"
    },
    "admin": {
        "edit_saved": false
    }
}
```
The JSON object is compressed using zlib before building the QR code. To build your own code:

1. Write a JSON object containing the changed settings with a structure as shown above.
2. Compress it using zlib.
3. Build a QR code from the resulting data.

### 4.5.4 List of keys for all settings

Here are the keys for all settings and the set of values they can take:

```json
{
    "project": {
        "name": "QR code project",
        "icon": "Q",
        "color": "#ff0000"
    },

    "admin": {
        "admin_pw": String,

        // User access control to the main menu. The default value is true.
        "edit_saved": Boolean,
        "send_finalized": Boolean,
        "view_sent": Boolean,
        "get_blank": Boolean,
        "delete_saved": Boolean,
        "qr_code_scanner": Boolean,

        "change_server": Boolean,
        "change_app_theme": Boolean,
        "change_app_language": Boolean,
        "change_font_size": Boolean,
        "change_navigation": Boolean,
        "show_splash_screen": Boolean,
        "maps": Boolean,
        "periodic_form_updates_check": Boolean,
    }
}
```

Our documentation is updated frequently. Get the latest version at [https://docs.getodk.org](https://docs.getodk.org).
### 4.5. Settings QR codes

<table>
<thead>
<tr>
<th>Setting</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>automatic_update</td>
<td>Boolean</td>
</tr>
<tr>
<td>hide_old_form_versions</td>
<td>Boolean</td>
</tr>
<tr>
<td>change_autosend</td>
<td>Boolean</td>
</tr>
<tr>
<td>delete_after_send</td>
<td>Boolean</td>
</tr>
<tr>
<td>default_to_finalized</td>
<td>Boolean</td>
</tr>
<tr>
<td>change_constraint_behavior</td>
<td>Boolean</td>
</tr>
<tr>
<td>high_resolution</td>
<td>Boolean</td>
</tr>
<tr>
<td>image_size</td>
<td>Boolean</td>
</tr>
<tr>
<td>guidance_hint</td>
<td>Boolean</td>
</tr>
<tr>
<td>external_app_recording</td>
<td>Boolean</td>
</tr>
<tr>
<td>instance_form_sync</td>
<td>Boolean</td>
</tr>
<tr>
<td>change_form_metadata</td>
<td>Boolean</td>
</tr>
<tr>
<td>analytics</td>
<td>Boolean</td>
</tr>
<tr>
<td>moving_backwards</td>
<td>Boolean</td>
</tr>
<tr>
<td>access_settings</td>
<td>Boolean</td>
</tr>
<tr>
<td>change_language</td>
<td>Boolean</td>
</tr>
<tr>
<td>jump_to</td>
<td>Boolean</td>
</tr>
<tr>
<td>save_mid</td>
<td>Boolean</td>
</tr>
<tr>
<td>save_as</td>
<td>Boolean</td>
</tr>
<tr>
<td>mark_as_finalized</td>
<td>Boolean</td>
</tr>
<tr>
<td>appTheme</td>
<td>String</td>
</tr>
<tr>
<td>app_language</td>
<td>String</td>
</tr>
<tr>
<td>protocol</td>
<td>String</td>
</tr>
<tr>
<td>server_url</td>
<td>String</td>
</tr>
<tr>
<td>username</td>
<td>String</td>
</tr>
<tr>
<td>password</td>
<td>String</td>
</tr>
<tr>
<td>formlist_url</td>
<td>String</td>
</tr>
<tr>
<td>submission_url</td>
<td>String</td>
</tr>
<tr>
<td>selected_google_account</td>
<td>String</td>
</tr>
<tr>
<td>google_sheets_url</td>
<td>String</td>
</tr>
<tr>
<td>font_size</td>
<td>Integer</td>
</tr>
<tr>
<td>navigation</td>
<td>Object</td>
</tr>
<tr>
<td>showSplash</td>
<td>Boolean</td>
</tr>
</tbody>
</table>

Our documentation is updated frequently. Get the latest version at [https://docs.getodk.org](https://docs.getodk.org).
"splashPath": String, // Absolute path to splash image

// Maps
"basemap_source": {"google", "mapbox", "osm", "usgs", "stamen", "carto"},
"google_map_style": {1, 2, 3, 4},
"usgs_map_style": {"topographic", "hybrid", "satellite"},
"carto_map_style": {"positron", "dark_matter"},
"reference_layer": String, // Absolute path to mbtiles file

// Form management
"form_update_mode": {"manual", "previously_downloaded", "match_exactly"},
"periodic_form_updates_check": {"every_fifteen_minutes", "every_one_hour", "every_six_hours", "every_24_hours"},
"automatic_update": Boolean,
"hide_old_form_versions": Boolean,
"autosend": {"off", "wifi_only", "cellular_only", "wifi_and_cellular"},
"delete_send": Boolean,
"default_completed": Boolean,
"constraint_behavior": {"on_swipe", "on_finalize"},
"high_resolution": Boolean,
"image_size": {"original", "small", "very_small", "medium", "large"},
"external_app_recording": Boolean,
"guidance_hint": {"no", "yes", "yesCollapsed"},
"instance_sync": Boolean,

// User and device identity
"analytics": Boolean,
"metadata_username": String,
"metadata_phonenumber": String,
"metadata_email": String,
}

4.6 Viewing filled forms on a map

New in version 1.25: ODK Collect v1.25.0

If you are collecting data related to physical locations, you can see a map of all filled forms from the Fill Blank Form menu. Use the map view to see your progress, to identify filled forms with bad locations and to plan your travel to the next location to survey.
4.6. Viewing filled forms on a map

**Note:** Blank forms must be downloaded with Collect v1.25 or later. Previously-downloaded forms will not have corresponding maps. If you update a form downloaded before Collect v1.25 and open its map, only filled forms created with the new form version will be displayed. Any previously-filled forms will not be included.

In the *Fill Blank Form* list, form definitions with a geopoint question outside of any repeat have a map button ( ). Tap on the map button to open a map showing the filled forms for that form definition. The first geopoint question in the form definition that is outside of any repeat is used to map filled forms. This question can either be a visible geopoint question or a question that gets the location in the background. On the map, you will also see your current position and be able to fill a new instance of the current form definition.

---

4.6.1 Form instance map markers

Filled forms are represented by colored map markers. The color of each map marker indicates the status of the filled form:

- Blue: saved and unfinalized
- Purple: finalized
- Green: sent

---

Our documentation is updated frequently. Get the latest version at [https://docs.getodk.org](https://docs.getodk.org).
Chapter 4. Using ODK Collect

- Red: send attempt failed

Tapping on a map marker shows details about the filled form at the bottom of the screen. The filled form will be identified by its instance name. In the screenshot above, first name, last name, and job title are displayed. Strategically choosing data from the form to show in the instance name can make the form map into a useful dashboard.

The detail view also shows a button for viewing or editing the selected filled form depending on its status. By default, forms with Saved, Finalized, and Submission Failure status are opened for edit and Sent forms are opened for viewing. However, if the Edit Saved Form button is made unavailable from the admin settings, forms with Saved, Finalized, and Submission Failure status are opened as view-only.

4.6.2 Map controls

There are three control buttons clustered at the top right of the map. The top button is used to zoom to the current location. The middle button adjusts the zoom level to ensure that all mapped filled forms are displayed on the screen. The last button is used to change layers if offline layers are available. The basemap and reference layer settings are used across all of Collect so the same ones will be used for this form map as for location widgets with maps.

The button at the bottom right of the screen can be used to fill a new instance of the current form definition. After you save a new filled form, you will be returned to the map and the filled form will be displayed if it has a geopoint associated with it.

4.6.3 Status bar

The bar at the bottom of the screen displays the total number of saved forms and how many of these are shown on the map. All filled forms with a value for the first geopoint question will be displayed. To ensure that all filled forms are displayed, make the identifying geopoint question required.

Note: Deleted or encrypted filled forms are not shown on the map. However, forms that were successfully sent and then deleted and forms that are encrypted both contribute to the total number of saved forms. See Deleting Forms for more on how filled form deletion works.

4.7 Using Offline Maps

Collect’s geopoint, geotrace, and geoshape questions can be configured to display different map data. The Maps Settings let you select a basemap to show, as well as a reference layer to show on top of the basemap.
4.7. Using Offline Maps

The data for all the available basemaps comes from services on the Internet, so the basemap will only be visible to users who are online. To choose a basemap, select a Source and then a Style if multiple styles are available.

For the reference layer, however, you can select a file on the device, and it will be visible offline. Offline layers are useful to present custom geospatial data layered over standard basemaps or as basemaps for low-connectivity environments. Use them to display high-resolution imagery, annotated maps, heatmaps, and more. ODK Collect can display any map layer saved as a set of tiles in the MBTiles format.

Warning:

- Vector MBTiles (with data in Mapbox Vector Tile format (pbf)) are only supported if Mapbox is selected as the basemap source.
- Vector MBTiles are currently displayed without styling. Each layer’s lines are displayed in a different color picked by ODK Collect. This color will be the same across all devices using the same MBTiles file. Fills are not displayed.

4.7.1 Offline maps quick start

1. Get or create your MBTiles file with TileMill or other software.
2. Transfer tiles to devices. The MBTiles file must be placed on your device in the layers subdirectory of your Collect directory, and the filename must end in .mbtiles.
3. Select your offline layer in the reference layer settings.
4. Open a geopoint, geotrace, or geoshape question.
5. While viewing the map, you can also select the offline layer using the button that looks like a stack of layers.

MBTiles files typically contain metadata that specifies the range of zoom levels in which they are visible. If you are viewing at an appropriate zoom level, your offline layer should be displayed. If you don’t see it, you might need to zoom in or out until the zoom level is in the range specified by the MBTiles file.

If the tileset has transparency (PNG or PBF tiles only), the selected basemap will show through. If it does not have transparency or you are offline, only your offline layer will be displayed.

4.7.2 Getting map tilesets

For non-commercial community mapping activities, Mapbox can arrange for offline MBTiles, including processed streets, satellite, and custom data. Contact community[at]mapbox.com
for offline Terms of Service exemptions and to receive technical guidance.

To create MBTiles files, use one of the compatible applications. Commonly used free software packages are TileMill and QGIS with the QTiles plugin. In general, you should build raster (jpg or png) MBTiles files. Vector (pbf) MBTiles files are only supported with Mapbox basemaps and are currently displayed without styling.

If you have existing geospatial data that is not in an MBTiles file, you may be able to convert it for use in Collect. For example, Tippecanoe is a tool to build vector MBTiles files from GeoJSON features (see warning above: vector MBTiles files are only supported with Mapbox basemaps and are displayed without styling).

### 4.7.3 Transferring offline tilesets to devices

MBTiles files must be manually transferred to Android devices to be available to Collect. Place the MBTiles files in the layers subdirectory of your Collect directory, and ensure their filenames end in .mbtiles.

To transfer files, you can upload them to an online service such as Google Drive, connect your device to a computer and transfer them via USB, or use adb.

### 4.7.4 Selecting offline tilesets

Once an MBTiles file has been transferred to the layers subdirectory of your Collect directory, it will be available for selection as a reference layer. A reference layer provides useful reference information for a data collector. A reference layer with no transparency acts like a basemap.

There are two ways to set the reference layer:

- from Maps Settings
- by tapping on the button that looks in a stack of layers in a geopoint, geotrace, or geoshape question

Both options set the reference layer for all geopoint, geotrace, and geoshape questions. The choices in the Collect layer selection menu will show the name of the tileset (from the Metadata table in the MBTiles file), as well as the path to the file.

- Advanced Use and Best Practices
  - Using Android Debug Bridge with Collect
  - Creating Shortcuts to Collect Forms
  - Improving Location Performance
  - Projecting ODK Collect onto another screen
5.1 Creating Shortcuts to Collect Forms

You can create an app-like shortcut to a specific form in Collect.

1. Click on *Apps* button at the bottom of the screen.
2. Touch and hold on an empty space on the Home screen and then click on *Widgets* button at the bottom of the screen.
3. Find the *ODK Form* widget and then touch and hold it.

4. A menu pops up listing all the available forms. Select the form you wish to create a shortcut for.
5. Shortcut for the selected form will appear on your home screen. You can move the shortcut to the desired position by drag and drop.
5.1. Creating Shortcuts to Collect Forms

**Note:** To delete the shortcut, hold and press it. A *Remove* button will appear on the top right corner. Drag and drop the shortcut to the *Remove* button. The shortcut will be removed.

**Tip:** If you are unable to find *Widgets* option:

1. Click on *Apps* and then click on three dots at the top right corner of your screen.
2. Click on Help option in the dropdown.
3. Click on *Adding Items* option in the menu which appears.

4. Information about adding widgets will be displayed. You can then follow it or try it to find the *Widgets* button.
5.2 Improving Location Performance

ODK Collect uses Android’s Fused Location Provider to get a device’s geolocation. This means data from multiple sensors including GPS, WiFi and Bluetooth are combined to quickly compute accurate geolocation readings. When Collect is accessing your device’s geolocation, you will see a location ( ) icon in the upper right corner of your screen. Learn more about geolocation-based questions in the question types section.

**Warning:** Some versions of ODK Collect prior to v1.28.0 use raw location sensors because of an Android bug. If you are having trouble getting a geolocation reading, first make sure you are using the latest version of ODK Collect.

5.2.1 Devices

Different devices can have dramatically different quality sensors. If you are having trouble getting a geolocation reading from a specific device, it is likely either because the device model does not have high-quality sensors or because the specific device has a faulty sensor. If you can, try going to the same geolocation with a different device to compare performance.
Sensor quality has dramatically improved over the years so in general, newer devices will get higher-accuracy readings more quickly.

### 5.2.2 Android Location Settings

Different Android versions provide different settings for controlling how geolocation is captured. See how to configure Android location in the Android documentation. In general, the more different sensors are available to Android, the better its geolocation estimation will be so make sure they are all enabled.

### 5.2.3 Helping the GPS Sensor

GPS works by receiving signals from specific satellites so to help your device get a geolocation reading, get as clear of a view of the sky as possible. GPS does not work inside of buildings and will be affected by obstacles like dense forest canopy or tall buildings. It is typically the main technology used to identify a device’s geolocation anywhere other than in urban areas where multiple WiFi access points and cell towers can be accessed at the same time.

It can take a while to get satellite lock when first attempting to get a geolocation. You can use a *background location question* at the beginning of your form definition to “warm” the GPS before data collectors to answer a *location question*. You can also use an external app such as GPS Status & Toolbox for this purpose. Additionally, this application can help you troubleshoot GPS issues. See its documentation for more details.

### 5.3 Android security recommendations

#### 5.3.1 Turn on data encryption

Turning on Android-level data encryption means when the device is locked, no one can see the data. Unlocking your encrypted device decrypts your data. Encryption can add protection in case your device is stolen. It’s an easier alternative to using encrypted forms that offer most of the benefits.

**Tip:** Encryption takes an hour or more to complete. Before you start, ensure that battery is charged and keep the device plugged in until encryption is complete. Make sure your data is backed up, just in case something goes wrong.

**Warning:** Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
• Interrupting encryption process may lead to loss of some or all of your data.
• The process is irreversible. The device cannot be decrypted once the encryption is setup and you will have to wipe out all the data for decryption or removing encryption.

Note: For devices running any version older than 4, you’ll need to either upgrade your operating system or consult the manufacturer’s instructions.

For devices running Android version 4 or later:

1. Open your device’s Settings app.

2. Tap Lock screen in the Device section. Then tap on Screen lock and create a pin or password.
5.3. Android security recommendations

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
Note: Encryption cannot be performed until you have setup either a PIN or a password lock. Pattern lock is not allowed with encryption.

Tip: Setting a strong passcode is imperative to protect your Android phone. The longer the passcode (or alphanumeric password), the tougher it will be for an attacker to gain access to your device. Even a lock screen won’t necessarily prevent a thief or hacker from getting access to your data. You can use apps that destroy all the data after few failed attempts to unlock the device. For more details, see this.

3. Tap Security in the System section.
5.3. Android security recommendations

4. Go to Encryption section. Now select Encrypt device to start encryption. Follow the onscreen instructions. During encryption, your device might restart several times.
**Note:** On some phones, you’ll need to choose *Storage*, then *Storage encryption* or *Storage → Lock screen and security → Other security settings* to find the *Encrypt device* option.

**Note:** If you are using SD card for storage, you can encrypt that too by choosing *Encrypt SD card* in the *Encryption* section. This not only encrypts the contents of the SD card, but it also means that the card cannot be used on another device unless it is wiped.

![Encrypt SD card in settings menu](image)

**Tip:** You should also ensure that device debugging (via adb) is disabled when collecting data, as that can enable users to pull data from the device after it has been successfully booted (when the sdcard encryption key is entered). i.e., if the debugging interface is enabled, someone could steal the device, connect it to a laptop, and pull data off it as long as it has not been shut down as they don’t need to successfully unlock the device’s lock screen to gain access.

For more details on encryption, see [this](https://docs.getodk.org).
5.3.2 Adjust Google Play to require a password for every purchase

You can set up Google Play to require a password for every purchase, which makes sure that anything you buy is done so with your consent. This can prevent enumerators from installing apps which can bypass certain specified requirements.

1. Open the Play store app, tap on the left-hand slide-out menu, and then choose Settings.
2. Look for *Require password for purchases* and tap on it. You’ll be asked to input your password.
3. Choose the password input frequency as *For all purchases through Google Play on this device.*
Note: The password will not be set for free downloads. If you want to lock free downloads as well, use an app locking app like AppLock.

5.3.3 Disable cloud-based backup

Though storing your data in the cloud is good for backing it up, law enforcement can demand that Google turn over your data. The best way to keep your Android phone from sending your personal data to its servers is to turn off backup. The downside is if you lose your phone, you may lose your data. Remember, you always have the option to manually back-up to your personal computer.

To disable backup:

1. Go to Settings app.
5.3. Android security recommendations

2. Then tap `Backup & Reset` in `Personalisation` section.
3. Now switch off the option to *Back up my data*.
5.3.4 Limit who can use Google Now

Google Now is your own intelligence assistant by bringing information to you when you need it but that gives Google a lot of access to your data to know what to dig up. The best way to use it is by turning it off from the lock screen, so only you with your passcode can use the feature and get access to your personal data. The steps to do this are as follows:

1. Open the Google app, tap on the left-hand slide-out menu, and then choose *Settings*. 
2. Tap on Voice in the Search section and then choose 'OK Google’ detection.
3. Turn off the feature *Say "OK Google" any time.*
5.3.5 Lower your phone’s sleep timeout

Lowering your phone’s sleep timeout can prevent opportunistic people from getting access to your unlocked device. The lower the figure, the quicker it locks you out.

1. Start by going to Settings app.
5.3. Android security recommendations

2. Tap on **Display and wallpaper** under the **Device** section.
3. Tap on *Screen timeout* and lower the screen timeout by choosing an appropriate timeout from the list.
4. Once you’ve lowered your phone’s sleep timeout setting, you need to make sure that your Android device locks and presents the lock screen when it wakes up. Tap on *Lock screen* in the *Device* section and then tap on *Lock automatically* option and choose an appropriate timeout again.
Chapter 5. Tips and Best Practices
5.3.6 Limit your lock screen notifications

Your lock screen can show a lot about your life. Your Android phone or tablet can limit what’s shown on the lock screen in order to prevent others from seeing your personal content as it comes in.

1. Go to Settings app then Tap on Sounds & notifications under the Device section.
2. Scroll down and tap on *Notifications on lock screen* under the *Notification* section. You can change how notifications are shown when device is locked setting. The most privacy conscious setting is to Hide sensitive notification content so that you know which app is alerting you, without showing its contents.
5.3. Android security recommendations

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
5.3.7 Prevent unauthorized apps from installing

Android devices can run third-party content outside of the Google Play app store. This can open up a device to malware attacks.

The easiest way to ensure that only verified and malware-checked apps can be installed on your phone or tablet is:

1. Go to the Settings app and then tap on Security in the System section.
2. Make sure that the Unknown sources option is turned off. If this option is turned on, installation of apps from trusted as well as unknown sources will be allowed.
5.3.8 Make sure you keep Android up-to-date

Many Android phone makers will now offer monthly security patches to ensure that any known vulnerabilities will be patched. Install these patches every month. It’s one of the best ways to ensure that you won’t be attacked by hackers and malware.

1. To periodically check for software updates, go to Settings app.

2. Then tap on About device under the System section.
5.3. Android security recommendations

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
Chapter 5. Tips and Best Practices

5.4 Projecting ODK Collect onto another screen

This guide helps the users to project ODK collect onto a screen. There are various methods available to do this, some of the methods are discussed below:

5.4.1 Using Vysor

Vysor is an extension for the Google Chrome browser that connects to an app on your smartphone. This extension enables you to control your phone from your PC or Mac using the mouse/trackpad and keyboard.

Before proceeding further make sure USB Debugging mode is enabled:

Enable USB Debugging Mode on Android

1. Go to Settings, choose About phone.
2. Scroll down and tap Build number seven times.
3. Go back to Settings and there you will find Developer options in the menu.
5.4. Projecting ODK Collect onto another screen

4. Scroll down and enable *USB debugging* mode.
5. Confirm the action when prompted.

**Note:** Windows users should make sure that they have downloaded Universal ADB Drivers.

Follow the instructions given below to use Vysor:

1. Make sure you have Google chrome installed, if not download it from [here](#).
2. Download Vysor extension.
3. Click on *Add To Chrome*

4. Confirm your action by clicking on *Add app.*
Warning: On Windows, you may get an error WebGL is not supported.

To fix this, follow the procedures given below:

First, we need to enable hardware acceleration:

1. Go to chrome://settings, scroll down and click on Advanced.

2. In the System section, ensure the Use hardware acceleration when available is en-
abled. You’ll need to relaunch Chrome for any changes to take effect. Click on RELAUNCH

3. Then, we need to enable WebGL, go to chrome://flags, scroll down and search for WebGL 2.0. From the drop-down list choose Enabled.
5.4. Projecting ODK Collect onto another screen

Now you can return to Vysor extension and install it again.

5. After adding the extension, you would be able to see it in the chrome toolbar, if it is not visible there go to chrome://apps and you would be able to see there.

6. Launch the extension and connect your phone through USB.
7. Click on *Find Devices*, select your device and click on *Select*.

8. After clicking *Select*, Vysor would be automatically downloaded to your phone, and you will be able to see your phone screen.

9. Click on the Collect app and there you go, you have successfully projected your phone screen.
5.4.2 Using Android Studio

Android Studio is the official IDE for Android. It provides tools for building apps for every type of Android device. Android Emulator can be used to test your app virtually on any Android device configuration.

Follow the procedures given below to run your app on the emulator:

1. Download Android Studio with SDK according to your platform.
2. Here is a tutorial on how to set up Android Studio according to different platforms.
3. After installing, launch Android Studio and click on Start a new Android Studio project or if you have an existing project click on Open an existing Android Studio project.
4. To create a new project follow further steps but if you have an existing project skip to the step 9.

5. Choose your project location and fill out the Application name and click on Next.
5.4. Projecting ODK Collect onto another screen

6. Select the *Phone and Tablet* option and choose your *Minimum SDK*. Click on *Next*.

7. Select an Activity, **Empty Activity is preferable**. Click on *Next*.
8. In Customize the Activity window, don’t change the default options and click on *Finish*. 

[Diagram of Customize the Activity window]
5.4. Projecting ODK Collect onto another screen

9. After few minutes, you will be able to see the Android Studio main window, click on icon, alternatively, you can click on *Tools* then select *Android*, from the drop-down menu select *AVD Manager*.

![Android Studio main window with AVD Manager highlighted](image1.png)

10. If you are an existing user list of all virtual devices would appear on the screen, to create a new virtual device, click on *+ Create Virtual Device*…

![Create Virtual Device in AVD Manager](image2.png)

11. In the Select Hardware window, choose a device definition for your virtual device. I have chosen *Nexus 5*, click on *Next*.

![Select Hardware window with Nexus 5 highlighted](image3.png)
12. Select a system image, I have chosen Lollipop version. Click on Next.

13. Enter your AVD Name, choose startup orientation and click on Finish.
5.4. Projecting ODK Collect onto another screen

14 Now you would be able to see your virtual device in Android Virtual Device Manager. Click on the play button to run your Android emulator.

Note: Please wait for some time as Android emulator takes very long time to start.

15. After the emulator is started, you would be able to see the screen of your emulator.
16. Now click on ![Symbol](image) to see the location of Android SDK.

17. Open the terminal and move to the `platform-tools` of the `SDK` directory.

```
$ cd platform-tools
```

18. Copy the `collect.apk` into `platform-tools` folder. You can download the apk file from here.

19. Type the following command to see the list connected devices:
5.4. Projecting ODK Collect onto another screen

```
$ adb devices
```

You should be able to see the emulator along with its port number, e.g. emulator-5554, Here 5554 is the port number. If the emulator is not present in the list, restart the emulator.

To install apk file, in the emulator type the following command:

```
$ adb install collect.apk
```

If the command is successfully executed, you will find your file in the launcher of your emulator.
Chapter 5. Tips and Best Practices

Using Command Line

You can also run the emulator using command line. Follow the steps given below to start your emulator using the command line:

**Note:** If SDK installation has been put in another drive or folder instead of in its default location of `$USER_HOME` or `$HOME`. Make sure you have set the environment variables according to that. In the command line type the following command to set environment variables.

```
set ANDROID_SDK_ROOT=path\sdk\n```

1. Open the terminal and move to the `emulator` folder of the `SDK` directory.

```
$ cd emulator
```

2. For the list of available virtual devices, type the following command:

```
$ emulator -list-avds
```

**Tip:** If you are not able to locate `emulator.exe` file in `SDK` folder. Type the following command to know the location of the file:

```
$ which emulator
```

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
On Windows:

```bash
> where emulator
```

3. Use **emulator** to start the emulator. Here *avd_name* is the name of Android virtual device that you have created.

```bash
$ emulator -avd avd_name
```

**Note:**

1. You can use **sdkmanager** command to update, install, and uninstall packages for the Android SDK. This method is not recommended as it is not user-friendly and also takes time.

To create an emulator you need to download system image for a particular API level.

```bash
$ sdkmanager --verbose "system-images;android-19;google_apis;x86"
```

- The `--verbose` option or `-v` option shows errors, warnings and all messages.
- `system-images;android-19;google_apis` specifies the system image package for the Android virtual device.
- `android-19` specifies the API level. You can choose different API level if you want.

2. To create and manage Android Virtual device from the command line, you can use **avdmanager**.

After downloading system image, you can use the following command to create an emulator.

```bash
$ avdmanager -v create avd --name testAVD -k "system-images;android-19;google_apis;x86" -g "google_apis"
```

- The `create avd` option creates a new Android virtual device.
- `--name` option is a **required** option which is used to specify name of the AVD. Here, the name of the AVD is testAVD.
- The `-g` specifies the system image tag to use for the AVD.
- `-k` specifies package path of the system image for the AVD.

**See also:**

Our documentation is updated frequently. Get the latest version at [https://docs.getodk.org](https://docs.getodk.org).
You can also use Genymotion as an alternative as it is very fast as compared to custom android emulators. It is also easy to use and configure, and it is available free of cost for personal use.

5.5 Related Topics

- *Intro to Forms in ODK*
  - *Example form questions in Collect*
- *Encrypted Forms*
- Integrating with other Android Apps
  - *Launching ODK Collect from External Apps*
  - *Launching External Apps*
ODK Central is the ODK server. It manages user accounts and permissions, stores form definitions, and allows data collection clients like ODK Collect to connect to it for form download and submission upload.

Our goal with Central is to create a server that is straightforward to install, easy to use and extensible with new features and functionality both directly in the software and with the use of our REST, OpenRosa, and OData programmatic APIs.

6.1 ODK Central Features

Here are some of the major features we support today:

- Projects to organize users, permissions, and forms
- Form upload and management
- User accounts with role-based permissioning
  - With support for form version updates
  - With drafts and testing on initial creation, and on version updates
  - With form multimedia or data attachments
  - Encrypted forms (self-supplied or project managed keys)
- Submission upload and management
  - Form filling from our mobile app or a web browser
  - From permissions-managed known users or anonymous public links
  - With submission multimedia or data attachments
  - With an interactive table preview of submission data
  - Support for reviewing, commenting on, and editing submissions after upload
– Connectable to data analysis and dashboard applications like Excel, Power BI, or R over OData

• Integrated checklist-based help system
• Optional encrypted off-site data backups to Google Drive
• Clean and modern REST API for integration and extensibility
• High performance on low-cost hardware or cloud providers
• ODK Briefcase-compatible data output
• ODK Briefcase push/pull support

Central is in active development. We have a lot of exciting ideas for its future and we look forward to hearing yours as well. See What is coming in Central for more on future direction.

6.2 Who should use ODK Central?

We recommend that all new data collection projects use ODK Central because it is in active development by the core ODK team. ODK Central is a relatively new server. It replaces aggregate-intro which is no longer being updated. Central isn’t as widely deployed as Aggregate, but its developers have put it through stress testing and it is used in production by many projects including several large ones.

Central solves some of the biggest problems with Aggregate. Some of our favorite features are:

• projects let you partition your server into different sandboxes to support multiple independent teams
• direct upload of XLSForm files makes form management easier
• the OData API makes it easy to synchronize your live form data to desktop visualization and dashboard tools
• managed encryption makes the process of handling encrypted form data significantly easier and in many cases more secure

Please give Central a try and provide your feedback.

6.3 Notes On Performance

Central was designed from early on to be stable, predictable, and fast on limited hardware. Perhaps most importantly, even under extreme traffic Central is guaranteed to either wholly succeed or wholly fail each submission. If, for instance, one submission attachment fails
to upload or persist, the entire submission upload will fail and Collect will attempt the submission again. If Central reports that a submission was successfully uploaded, then all submitted data was successfully saved.

We have done some work to benchmark Central to verify these claims, and produce some guideline numbers. Every circumstance is different, and a lot will depend on your form design, your geographic location, and other factors. But in general, on the second-cheapest DigitalOcean configuration at time of writing ($10/month, 2GB memory), we found the following:

- A 250 question form without attachments could support 500 devices simultaneously uploading many submissions without issues, at a rate of roughly 41.2 submissions per second.
- A larger 5000 question form, without attachments, could also support 500 devices submitting data at once, but runs more slowly (~12 submissions/second) and fails about one submission in every 1000 (which can then be re-submitted without issues).
- Including attachments slows the process down, since there is more data to shuffle around. Realistically, the number of concurrent users supported in this scenario will decrease simply because Internet bandwidth in and out of Central will limit the number of submissions it can see at a time. But we have tried situations featuring 5MB submissions with 50 devices at once without seeing issues (though for the mentioned reasons the response rate drops to between 1 and 2 submissions/second). Additionally, data exports with attachments take longer and are more memory-intensive.

When you are planning for your installation and selecting a destination to deploy Central to, keep these numbers in mind. If 500 people submitting data all at the same time is a distant scenario, you can probably get by with a lower-performance option. If your deployment is larger than these numbers, consider bumping up to a more powerful machine. If you aren’t sure, ask around in the forums.

6.4 Learn more about ODK Central

- Setting Up ODK Central
- Using ODK Central
Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
7.1 Installing Central

Central is distributed and installed using Docker. Docker makes it possible to describe exactly how Central’s different components should be configured no matter where it is installed. Don’t worry if you don’t know about Docker yet! We have put together step-by-step instructions for our recommended solutions below.

**Warning:** Central is intended to be configured and administrated entirely with `docker-compose stop` and that is what we show in all our instructions. If you inadvertently do a `docker-compose down`, it will look like you have lost your data. See the troubleshooting section to learn how to recover.

7.1.1 Using ODK Cloud (recommended)

The easiest way to get a Central server is by using ODK Cloud.

ODK Cloud provides fast Central servers with regular feature updates, automatic security patches, daily backups, uptime management, enterprise security, and guaranteed support on any issues.

By choosing ODK Cloud, you are also directly supporting future development on ODK and helping make it better for everyone.

7.1.2 Installing on DigitalOcean

If you want to install Central on own server but you’re not sure what cloud provider to choose, we recommend DigitalOcean. Start here.
7.1.3 Installing elsewhere

If you got excited when you saw mention of Docker above, and you already have your own destination and process for managing Docker deployments, you’re all set to go. Central is entirely defined via Docker Compose, which means the docker-compose command will be all you need to manage the entire system.

**Warning:** We verify each version of Central on DigitalOcean and confirm that upgrades are possible. However, we do not verify them on other cloud providers and generally can’t provide free support for installations that deviate from the DigitalOcean instructions. You may find other community members able to help on the forum.

No matter where you plan to install Central, we recommend reviewing the instructions we’ve provided for DigitalOcean starting from this section. In particular, you’ll need to update your submodules after you clone the repository, and configure your .env file for your installation.

### Installing on Amazon EC2

Amazon Web Services (AWS) is one of the many other options for installing Central. It’s a good idea to read through the instructions we’ve provided for DigitalOcean, as many of the steps remain the same or similar.

To obtain a server you will need to first **create an AWS account**. When launching your instance, select the Ubuntu Server 20.04 LTS AMI in step 1. The `t2.micro` instance type has the 1GB of memory recommended for if you don’t expect many forms to be submitted at once and you don’t expect many large media attachments.

When adjusting the security settings open up the ports for SSH, HTTP, and HTTPS. Once you have launched your instance, go to the Elastic IPs menu option under Network & Security, then allocate a new address and associate it with your server in order to keep the IP address for your server consistent.

Before installing Central on your server, you need to install Docker and Docker Compose. Follow the instructions below.

1. Install Docker Engine on Ubuntu.
2. Install Docker Compose.


Finally, **configure an e-mail service** such as Amazon SES because Amazon restricts emails sent from EC2.
7.2. Installing Central on DigitalOcean

Installing on Windows

We strongly recommend using Linux, preferably the latest Ubuntu Server LTS, as the host operating system (natively or in a virtual machine) for your Central install.

If you wish to use Windows as the host, it’s a good idea to read through the instructions we’ve provided for DigitalOcean, as many of the steps remain the same or similar.

The major differences are that you’ll first need to install Git, Node.js, Docker, and Docker Compose. You’ll also need to clone the Central repository with the correct line endings: 

```bash
git clone -c core.autocrlf=false https://github.com/getodk/central
```

7.2 Installing Central on DigitalOcean

If you’d like to set up an ODK server that’s accessible from anywhere via the Internet, DigitalOcean provides a one-click configuration that’s nicely geared with nearly all the tools you’ll need to set up your new server. The only thing it doesn’t do is register a domain name, which you will have to do in order to obtain a security certificate for your server.

**Tip:** If you have not already created a DigitalOcean account, use our referral link. DigitalOcean will give you $100 of credit to spend during the first 60 days so that you can try things out. Once you have spent $25 with them, we’ll get $25 to put towards our hosting costs.

In general, this installation process will involve five phases:

1. Getting a server and loading it with the appropriate base system.
2. Getting a web address (domain name) and pointing it at your new server.
3. Getting the Central software and installing it on your server.
4. Preparing Central for startup and running it for the first time.
5. Creating your first Central Administrator account and logging into it.

There are also some optional other steps you can take, which you can find at the bottom of this page.

7.2.1 Getting a Server

In this phase, you will create a new server on DigitalOcean, choose a pricing tier, configure it with the correct base operating system, and start it up.
Chapter 7. Setting Up ODK Central

If you haven’t already, create an account on DigitalOcean. Then, from the DigitalOcean control panel, use the Create button at the top to create a new Droplet. This is their name for a server you can access and manage.

At the very top, under Choose an image, switch to the Marketplace tab and select the Docker option. The version does not matter.

As you continue down this page, there are a few options that may be important to you:

- There is a section for standard droplets and another for more expensive optimized
7.2. Installing Central on DigitalOcean

droplets. In general, you should not need optimized droplets.

- The **size** option affects a few things, but the most important is the amount of memory available to your server. Memory does not affect storage space, it sets the amount of "thinking room" the server gets while it’s working on things. If you don’t expect many forms to be submitted at once and you don’t expect many large media attachments, you can start with 1GB. Higher-load servers and servers which handle many image or video attachments may need 2GB or more. It is pretty easy to upgrade to a larger size later.

- The datacenter region selects where physically your server will be located. If you have security concerns, this is your chance to decide which country hosts your data. Otherwise, generally selecting the option with closest geographic proximity to your users is a good idea.

- If you are technically savvy and understand what an SSH key is, there is a field here that you will want to fill in. If not, don’t worry about it.

**Tip:** If you choose a 1GB server and you have problems with exporting attachments, read this troubleshooting tip.

Once you click on **Create**, you’ll be taken back to the Droplet management page. It may think for a moment, and then your new server should appear. Next to it will be an IP address, which should look something like 183.47.101.24. This is where your server is publicly located on the Internet. Don’t worry, nobody can do anything with it until you let them.

Congratulations! With those steps, you have now created a new server which you can access over the Internet, and started it up. Next, we will get a web domain name address (like getodk.org) to point at it.

### 7.2.2 Getting a Web Address (Domain Name)

Now is the time to set up a domain name. We will do so, and then configure it so that it sends users to the server you created in the previous step.

You’ll need to do this for two reasons: a memorable name (like google.com) will be easier to remember and access than a pile of numbers, and you cannot get a security certificate without one. It is not currently possible to host Central within a subdirectory on another domain (so, my-website.com/my-odk-server is not possible, but my-odk-server.com is allowed, as is my-odk-server.my-website.com).

If you already know how to do these sorts of things, feel free to ignore the following instructions and proceed on your own. You can rejoin us at the next section.

For the rest of us, there are some options here:
• You can pay one of the many popular commercial domain registrars for a full domain name, like MyOdkCollectionServer.com. Search for "domain registrar" to find one of these. These often cost as little as $3/year.

• You can use a free DNS service: we recommend FreeDNS, which has run for a long time and has a good reputation. With it, you can get a free name, albeit with a fixed second half (like MyOdkCollectionServer.dynet.com). If you choose this route, we recommend using one of the less popular names, as the heavily occupied names can run into trouble later on (in particular, getting a security certificate from Let’s Encrypt).

Whichever option you choose, once you get a domain name you’ll want to look at DigitalOcean’s guide on setting up domain names for your Droplet. In general, you’ll point your domain name in DigitalOcean’s direction at your registrar, then in DigitalOcean itself you’ll want to create an A record that points to the IP address we found above.

New domain names take a little bit to get working. Meanwhile, we can get working on installing the server software.

### 7.2.3 Installing Central

In this phase of installation, we will log into your new server, get the Central software, load some settings into it, and install it.

First, you’ll need to be able to log into the server itself. If you are an advanced user who filled in an SSH key above, you’re good to go. Otherwise, click your email for a message from DigitalOcean with your server password.

Once you have that password in hand, you’ll be able to use the Launch Console button to log into your server: when it asks for login, type root and press Enter. Then type the password you were emailed and press Enter again.
7.2. Installing Central on DigitalOcean

Once you are in your server, you’ll want to change your password so that people snooping your email do not gain access. You should be automatically asked for a new password the first time you log in. If you are not, type `passwd` and press **Enter**, then follow the instructions to choose a new password. From now on, you will use that password to log in.

**Changing Server Settings**

First, you’ll need to upgrade to docker-compose v1.28.3 or later. Follow these commands from Docker’s documentation.

```bash
sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```

Next, we will want to ensure that Docker starts up whenever the server starts. Docker will in turn ensure that Central has started up. To do this, run `systemctl enable docker`.

You will need to change one more thing on this server before we proceed: you will need to modify the system firewall for Enketo features in Central to work correctly.

The quickest way to do this is to run `ufw disable` while logged into your server’s command line prompt. You should see the message **Firewall stopped and disabled on system startup.** If so, you have configured the firewall correctly.

**For advanced administrators**

While it sounds dangerous, disabling your system firewall does not put your server at greater risk. In fact, most Linux operating systems come with the system firewall disabled.

If you don’t want to disable the firewall entirely, you can instead configure Docker, `iptables`, and `ufw` yourself. This can be really difficult to do correctly, so we don’t recommend most people try. Another option is to use an upstream network firewall.

The goal here is to ensure that it is possible to access the host through its external IP from within each Docker container. In particular, if you can successfully `curl` your Central website over HTTPS on its public domain name, all Enketo features should work correctly.

**Getting and Setting Up Central**

Now you’ll need to download the software. In the server window, type `git clone https://github.com/getodk/central` and press **Enter**. It should think for some time and download many things. Then type `cd central` to start working with the software.
You now have the framework of the server software, but some components are missing. Type `git submodule update -i` and press **Enter** to download them.

Next, you need to update some settings. First, copy the settings template file so you can edit it: type `mv .env.template .env` and press **Enter**.

Then, edit the file by typing `nano .env` and pressing **Enter**. This will launch a text editing application.

- Change the **DOMAIN** line so that after the `=` is the domain name you registered above. As an example: **DOMAIN=MyOdkCollectionServer.com**. Do not include anything like `http://`.
- Change the **SYSADMIN_EMAIL** line so that after the `=` is your own email address. The Let’s Encrypt service will use this address only to notify you if something is wrong with your security certificate.
- Leave the rest of the settings alone. If you have a custom security or network environment you are trying to integrate Central into, see the advanced configuration sections for more information on these options.
- Hold Ctrl and press x to quit the text editor. Press y to indicate that you want to save the file, and then press **Enter** to confirm the file name. Do not change the file name.
7.2. Installing Central on DigitalOcean

Now, we will bundle everything together into a server. Type `docker-compose build` and press **Enter** to do this. This will take a long time and generate quite a lot of text output. Don’t worry if it seems to pause without saying anything for a while. When it finishes, you should see some "Successfully built" type text and get your input prompt back. When that happens, type `docker-compose up --no-start` and press **Enter**.

Once that is complete, congratulations! You have installed your copy of Central. Next, we need to teach the server how to start it up, and do so.

### 7.2.4 Starting up Central

Now, run `docker-compose up -d` to start the server software. The first time you start it, it will take a while to set itself up. Once you give it a few minutes and you have input control again, you’ll want to see whether everything is running correctly:

- To see if ODK has finished loading, run `docker-compose ps`. Under the State column, for the `nginx` row, you will want to see text that reads `Up` or `Up (healthy)`. If you see `Up (health: starting)`, give it a few minutes. If you see some other text, something has gone wrong. It is normal to see `Exit 0` for the `secrets` container.

- If your domain name has started working, you can visit it in a web browser to check that you get the Central management website.

You’re almost done! All you have to do is create an Administrator account so that you can log into Central.
7.2.5 Logging into Central

If visiting your server domain name address in your browser does not load the Central management website, you may have to wait a few minutes or hours (possibly even a day) for the domain name itself to get working. These instructions are explained in further depth on the page detailing the Central Command Line Tools.

Once you do see it working, you’ll want to set up your first Administrator account. To do this:

- Ensure that you are in the central folder on your server. If you have not closed your console session from earlier, you should be fine. If you have just logged back into it, you’ll want to run cd central to navigate to that folder.

- Then, type `docker-compose exec service odk-cmd --email YOUREMAIL@ADDRESSHERE.com user-create`, substituting your email address as appropriate. Press Enter, and you will be asked for a password for this new account.

- The previous step created an account but did not make it an administrator. To do this, type `docker-compose exec service odk-cmd --email YOUREMAIL@ADDRESSHERE.com user-promote Enter`.

- You are done for now, but if you ever lose track of your password, you can always reset it by typing `docker-compose exec service odk-cmd --email YOUREMAIL@ADDRESSHERE.com user-set-password`. As with account creation, you will be prompted for a new password after you press Enter.

Once you have one account, you do not have to go through this process again for future accounts: you can log into the website with your new account, and directly create new users that way.

**Tip:** If you find that users are not receiving emails, read about troubleshooting emails.

7.2.6 Setting Up Monitoring

The last thing you will want to do is to set up server monitoring. Alerts and monitoring are important because they can inform you of problems with your server before they affect your data collection project.

You can find instructions for setting up alerts in the DigitalOcean Documentation.

We strongly recommend creating an alert for Disk Utilization. A threshold of 90% is usually reasonable. By far the most common operations issue we see is servers running out of disk space as large media attachments pile up. If your server runs entirely out of disk space, it can crash and become unresponsive. It is best to upgrade your storage plan before this happens.
7.3. Advanced Configuration Options

If you are familiar with server operations, you may wish to set up some other alerts: CPU usage and Memory Utilization are the most interesting remaining metrics. However, these are not as important or easily understandable as the Disk Utilization alert, so you may skip this if you’re not sure what to do here.

You’re done! Congratulations. In the future, you may wish to consult the Upgrading Central guide, but for now you may begin using Central. The Using ODK Central sections can help you with your next steps if you aren’t sure how to proceed.

7.3 Advanced Configuration Options

The following sections each detail a particular customization you can make to your server setup. Most installations should not need to perform these tasks, and some of them assume some advanced working knowledge on administering Linux web servers. If you aren’t sure what something means, the best option is probably to skip the section completely.

7.3.1 Adding Swap

To avoid Central crashing if your server runs out of memory, you may want to add swap. If you are having issues with Central running out of memory, we recommend adding more physical memory. However, adding swap can be an effective temporary workaround or a preventative measure against spikes if, for example, multiple people initiate data exports at the same time.

Whether or not you choose to add swap, we recommend monitoring memory usage and adding memory if your server is routinely running close to the physical memory limit.

To add swap, log into your server so you have a console prompt, and run these commands, adapted from this article:

```
fallocate -l 1G /swap
dd if=/dev/zero of=/swap bs=1k count=1024k
chmod 600 /swap
mkswap /swap
swapon /swap
```

Run `nano /etc/sysctl.conf` and add the following to the end of the file to ensure that swap is only used when the droplet is almost out of memory.

```
vm.swappiness=10
```

Finally, run `nano /etc/fstab` and add the following to the end of the file to ensure that the swap file is permanently available.
7.3.2 Adding External Storage

Forms with many large media attachments can fill up your droplet’s storage space. To move your database to external storage, follow these steps:

1. **Add a new volume** to your droplet.
2. **Move the Docker data directory** to the new volume.
   
   (a) To find the location of your new volume, run `df -h`. The mount location will look like `/mnt/volume_nyc1_01`.
   
   (b) Then create a `docker` folder at that location with `sudo mkdir /mnt/volume_nyc1_01/docker`.
   
   (c) `/mnt/volume_nyc1_01/docker` will be the `/path/to/your/docker` you use.

7.3.3 Using a Custom SSL Certificate

By default, Central uses Let’s Encrypt to obtain an SSL security certificate. For most users, this should work perfectly, but larger managed internal networks may have their own certificate trust infrastructure. To use your own custom SSL certificate rather than the automatic Let’s Encrypt system:

1. Generate a `fullchain.pem` (-out) file which contains your certificate followed by any necessary intermediate certificate(s).
2. Generate a `privkey.pem` (-keyout) file which contains the private key used to sign your certificate.
3. Copy those files into `files/local/customssl/` within the repository root.
4. In `.env`, set `SSL_TYPE` to `customssl` and set `DOMAIN` to the domain name you registered. As an example: `DOMAIN=MyOdkCollectionServer.com`. Do not include anything like `http://`.
5. Build and run: `docker-compose build nginx`, `docker-compose stop nginx`, `docker-compose up -d nginx`. If that doesn’t work, you may need to first remove your old nginx container (`docker-compose rm nginx`).

7.3.4 Using a Custom Mail Server

Central ships with a basic EXIM server bundled to forward mail out to the internet. To use your own custom mail server:
7.3. Advanced Configuration Options

1. Ensure you have an SMTP relay server visible to your Central server network host.

2. Edit the file `files/service/config.json.template` to reflect your network hostname, the TCP port, and authentication details. The `secure` flag is for TLS and should be set to `true` if the port is 465 and `false` for other ports. If no authentication is required, remove the `auth` section.

   ```json
   "email": {
     "serviceAccount": "my-replyto-email",
     "transport": "smtp",
     "transportOpts": {
       "host": "smtp.example.com",
       "port": 587,
       "secure": false,
       "auth": {
         "user": "my-smtp-user",
         "pass": "my-smtp-password"
       }
     }
   }
   }
   ``

3. Build and run: `docker-compose build service, docker-compose stop service, docker-compose up -d service`.

7.3.5 Using a Custom Database Server

**Warning:** Using a custom database server, especially one that is not local to your local network, may result in poor performance. We strongly recommend using the Postgres v9.6 server that is bundled with Central.

Central ships with a PostgreSQL database server. To use your own custom database server:

1. Ensure you have a PostgreSQL database server visible to your Central server network host.

2. Ensure your database has `UTF8` encoding by running the following command on the database.

   ```sql
   SHOW SERVER_ENCODING;
   ```

3. Ensure `CITEXT` and `pg_trgm` extensions exist by running the following commands on the database.

   ```sql
   CREATE EXTENSION IF NOT EXISTS CITEXT;
   CREATE EXTENSION IF NOT EXISTS pg_trgm;
   ```
4. Edit the file `files/service/config.json.template` to reflect your database host, table, and authentication details.

```json
"database": {
    "host": "my-db-host",
    "user": "my-db-user",
    "password": "my-db-password",
    "database": "my-db-table"
},
```

4. Edit the file `docker-compose.yml` to update the command for the `service` container.

```yaml
command: [ ".wait-for-it.sh", "my-db-host:my-db-port", "-", ".start-odk.sh" ]
```

5. Build and run: `docker-compose build service`, `docker-compose stop service`, `docker-compose up -d service`.

### 7.3.6 Configuring DKIM

**Tip:** Users are not receiving emails? Read [*troubleshooting emails*](https://docs.getodk.org) before configuring DKIM.

DKIM is a security trust protocol which is used to help verify mail server identities. Without it, your sent mail is likely to be flagged as spam. If you intend to use a custom mail server (see the following section), these instructions will not be relevant to you. Otherwise:

1. Ensure that your server’s name in DigitalOcean matches your full domain name, and that the hostname does as well. If you had to make changes for this step, restart the server to ensure they take effect.

2. There can be in some cases a placeholder folder that you may have to delete first. If you run this command and no file was deleted, proceed to step 3.

   ```bash
   rmdir ~/central/files/dkim/rsa.private
   ```

3. Now, you’ll need to generate a cryptographic keypair and enable the DKIM configuration. Run these commands:

   ```bash
   cd ~/central/files/dkim
   openssl genrsa -out rsa.private 1024
   openssl rsa -in rsa.private -out rsa.public -pubout -outform PEM
   cp config.disabled config
   ```

4. With the contents of the public key (cat `rsa.public`), you’ll want to create two new TXT DNS records:
7.3. Advanced Configuration Options

(a) At the location `dkim_domainkey.YOUR-DOMAIN-NAME-HERE`, create a new TXT record with the contents `k=rsa; p=PUBLIC-KEY-HERE`. You only want the messy text between the dashed boundaries, and you'll want to be sure to remove any line breaks in the public key text, so that it's all only letters, numbers, +, and /.

(b) At your domain name location, create a new TXT record with the contents `v=spf1 a mx ip4:SERVER-IP-ADDRESS-HERE -all` where you can obtain the server IP address from the DigitalOcean control panel.

5. Finally, build and run to configure EXIM to use the cryptographic keys you generated:

```bash
      cd ~/central
      docker-compose build mail
      docker-compose stop mail
      docker-compose up -d mail
```

If you see an error that says `Can't open "rsa.private" for writing, Is a directory.`, you will need to `rmdir ~/central/files/dkim/rsa.private`, then attempt `docker-compose build mail` again. If you see some other error, you may need to first remove your old mail container (`docker-compose rm mail`).

7.3.7 Customizing Enketo

**Warning:** Changing Enketo's configuration may break Central in subtle and unexpected ways. Do not make changes if you do not understand the implications of those changes.

Enketo is the software that Central uses to render forms in a web browser. It is used for form previews, web browser submission, and submission editing. Common customizations include enabling geocoding, adding analytics, and setting a default theme.

1. Read the Enketo configuration tutorial and `default-config.json` to understand what is possible.

2. Edit the file `files/enketo/config.json.template` to reflect your desired changes.

3. Build and run: `docker-compose build`, `docker-compose stop mail`, `docker-compose up -d`.

7.3.8 Disabling or Customizing Sentry

By default, we enable Sentry error logging on the backend server, which provides the Central development team with an anonymized log of unexpected programming errors that occur while your server is running. This information is only visible to the development team and
should never contain any of your user or form data, but if you feel uncomfortable with this anyway, you can take the following steps to disable Sentry:

1. Edit the file `files/service/config.json.template` and remove the `sentry` lines, starting with "sentry": { through the next three lines until you remove the matching }.

2. Build and run: `docker-compose build service`, `docker-compose stop service`, `docker-compose up -d service`.

If on the other hand you wish to use your own Sentry instance, take these steps:

1. Create a free account on Sentry, and create a new nodejs project.

2. The new project will generate a DSN of the format `https://SENTRY_KEY@sentry.io/SENTRY_PROJECT`.

3. In `files/service/config.json.template`, replace `SENTRY_KEY` and `SENTRY_PROJECT` with the values from step 2.

```json
{
    "default": {
        "database": {...},
        "email": {...},
        "env": {...},
        "external": {
            "sentry": {
                "key": "SENTRY_KEY",
                "project": "SENTRY_PROJECT"
            }
        }
    }
}
```

The error logs sent to Sentry (if enabled) are also being written to `/var/log/odk/stderr.log` in the running backend container.

### 7.4 Upgrading Central

We release new versions of Central regularly. You do not have to upgrade to the latest version immediately, but we generally recommend that you do so to get access to the newest features, bug fixes, and security updates.

**Note**

To see your version number, click on the question mark icon in the upper right section of your Central menu bar, then click Version. If you don’t see the question mark,
7.4. Upgrading Central

you can also see the version number by adding `version.txt` to the root URL (e.g., `demo.getodk.cloud/version.txt`).

To perform an upgrade, you’ll first need to get to the software. You’ll need to log into your server’s command line prompt again, like you did when you first set up the server. If you used our DigitalOcean installation steps but can’t quite remember how to do this, we suggest reviewing the section Installing Central as a reminder, or if you can’t remember your password to start at the top of that section to reset your password.

Once you are logged into your server, navigate back to the project folder (`cd central`). Then, get the latest version of the infrastructure: `git pull`.

**Note**

If you have made local changes to the files, you may have to start with `git stash`, then run `git stash pop` after you perform the pull. If you aren’t sure, just run `git pull` anyway and it will tell you.

```
Warning: If you are upgrading to Central v1.3, see Upgrading to Central v1.3 to ensure you have the correct version of docker-compose.

If you are upgrading to Central v1.2, see Upgrading to Central v1.2 to learn how to resolve any error messages using `git pull`.
```

Now, get the latest client and server: `git submodule update -i`. Then, build your server from the latest code you just fetched: `docker-compose build`.

**Note**

If you run into problems with this step, try stopping the Central software (`docker-compose stop`) and retry `docker-compose build` after it has shut down the Central website.

Next, you need to do a little bit of maintenance. Run `docker prune`. If it thinks `prune` is not a docker command, run `docker image prune` instead. You’ll be asked to confirm the removal of all dangling images. Agree by typing the letter `y` and pressing `Enter`.

Finally, restart the running server to pick up the changes: `docker-compose stop` and `docker-compose up -d`.
Chapter 7. Setting Up ODK Central

7.4.1 Upgrading to Central v1.3

Before upgrading, run `docker-compose --version` to confirm you have docker-compose v1.28.3 or later. If you don’t, upgrade with these commands from the Docker’s documentation.

```bash
sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```

7.4.2 Upgrading to Central v1.2

In v1.2, we added some advanced features to Central’s server configuration. These features will not be meaningful to most users. However, because we would like to make this change and further improvements in the future, we have modified the template `.env` configuration file you set up during installation.

Since you have made your own changes to the `.env` file to set Central up for your environment, you will see an error message when you run the `git pull` command:

```
error: Your local changes to the following files would be overwritten by merge:
    .env
Please commit your changes or stash them before you merge.
```

Don’t worry, nothing bad happens if you see this. To get around this error, run this set of commands instead of `git pull`:

```bash
mv .env env-tmp
git pull
mv env-tmp .env
```

Afterwards, `git status` should not say anything about the `.env` file at all and you can continue with the upgrade instructions above.

If `git status` still shows errors, copy and paste your entire console session into a forum thread and someone will help you out.

7.4.3 Upgrading to Central v0.9

Particularly if you are installed on DigitalOcean, you will need to modify the system firewall for Enketo features in Central to work correctly.

The quickest way to do this is to run `ufw disable` while logged into your server’s command line prompt. You should see the message `Firewall stopped and disabled on system`
If so, you have configured the firewall correctly.

For advanced administrators

While it sounds dangerous, disabling your system firewall does not put your server at greater risk. In fact, most Linux operating systems come with the system firewall disabled.

If you don’t want to disable the firewall entirely, you can instead configure Docker, iptables, and ufw yourself. This can be really difficult to do correctly, so we don’t recommend most people try. Another option is to use an upstream network firewall.

The goal here is to ensure that it is possible to access the host through its external IP from within each Docker container. In particular, if you can successfully curl your Central website over HTTPS on its public domain name from within the Enketo container, all Enketo features should work correctly.

Tip: By default, Central does not send any usage information to the Central team. Choosing to share anonymized usage analytics with the Central team helps us make Central better for you!

We are always trying to improve Central.

To do this, we need your feedback, so that we can understand how you are using Central and how it could be better for you. Without the usage information we need to understand how ODK is actually used by our global community, we cannot make the best decisions to address your most important issues.

Choosing to share anonymized usage analytics with the Central team helps us make Central better for you!
You can find the Analytics settings under → System, then → Improve Central.

You can select the middle “not interested” option to prevent the yellow reminder at the top of the screen from appearing. Or, you can see an exact sample of the usage information we send by clicking on the What metrics are sent? link.

You can opt to also include your contact information with the report. This does mean we know which server’s usage report is yours. But this also means that if we see something surprising about your use of Central, we can contact you to learn more about your unique needs.

### 7.6 Backing Up Central

Having a data backup strategy is a critical part of running a web service like ODK Central. Backups should go to a system in a different physical location from where Central is installed in order to prevent data loss across a broad range of scenarios. Many cloud providers offer backup strategies that run automatically.

If you are an experienced system administrator, you may want to set up your own backups of the PostgreSQL database that contains all of Central’s data. One strategy for doing this is to configure a separate database server and back up that database server.

You will additionally need to have a backup of Enketo data to be able to restore existing Web Form links. The strict minimum required to do this are Enketo’s Redis store and the keys generated in the Enketo configuration. **In general, we recommend making a full system backup.**
7.6. Backing Up Central

If you don’t already have a full system backup in place and don’t want to set up your own database backup, Central provides a managed backup system to Google Drive.

7.6.1 Managed backups / Direct backups via API

**Warning:** Managed backups and Direct backups via API include all of your collected data but do NOT include sufficient information to re-establish the same Web Form links.

If you use Public Links for broad surveying or share links to Web Forms through another system, we strongly recommend also making a full system backup.

If you only use web forms for previews or for making submissions directly from Central, managed backups are sufficient. You can regenerate previews by uploading the same form with a new form version.

ODK Central features an optional off-site backup system that backs up all Central data (but as detailed above, not Web Form configurations).

For each backup, we extract all your data (including user accounts, forms, and submissions), we encrypt it so that only you can access it, and we send the encrypted result to your Google Drive account for safekeeping. We send data to Google Drive because it is a service that many people already use, it is easy to set up, and it is relatively inexpensive. Because backups are encrypted, Google can’t see their contents. However, you should verify that policies and laws that govern your project allow this usage.

**About Google Drive account access**

When ODK Central requests to connect to your Google Drive account, it only requests permissions from Google to:

- Create new files and folders in your Drive
- Read and modify files and folders *that it created*

This means that ODK Central cannot read or modify any other files or folders in your Drive, no matter what.

To see your current managed backups status, navigate to → *System* at the top of the Central management website. You should see a status page for backups that looks something like this:
7.6.2 Setting up backups

1. To set up a new automated backup, click on the *Set up now* button on the right.

   **Tip:** If you see a *Terminate* button instead of a *Set up now* button, you already have an automated backup configured. Right now, you can only have one automated backup scheduled at a time. If you wish to change where the backup is saved, you will need to terminate the old one before creating a new one.

2. You’ll be asked to enter an optional passphrase. This passphrase is what the server will use to encrypt your backups. You will be unable to restore the backup without the passphrase, exactly as you type it in here. If you leave this field blank, we will still encrypt your backup data but anybody will be able to decrypt it by doing nothing more than leaving the passphrase blank.

3. The next step talks about connecting to your Google Drive account to store your backups. When you press *Next* again, a Google permissions page will appear in a popup. You will need to press *Allow* to proceed. If you are feeling unsure about granting access, please see the ”About Google Drive account access” note at the top of this page.

---

Our documentation is updated frequently. Get the latest version at [https://docs.getodk.org](https://docs.getodk.org).
4. Once you press **Allow**, you will see a screen in the popup which contains a lengthy code, and instructions to copy and paste it back into "your application." Copy the code, switch back to the ODK Central website, and paste it into the *Confirmation text* box. Press *Next* to confirm it.

Sign in

Please copy this code, switch to your application and paste it there:

4/AAB4ZK9nOLkX1ov-nGppiWGQckdJ7tC1qNgEvbw9q5KFpxbqJdyhLw
5. The setup box should close and you should see a message telling you *Success! Automated backups are now configured.*

6. Backups are scheduled to run once a day, at 02:00 server local time. If more than 24 hours pass without a backup completing successfully, you’ll want to double check that everything has been correctly set up.

**Tip:** You can verify your Google Drive usage on the Drive storage page. You may want to periodically remove older backups to free up space.

### 7.6.3 Performing an immediate backup

It is possible to immediately download a backup of your database to your own computer. As of Central v1.1, you will still need to have managed backups configured in order to access this functionality in the web management interface. Once you do, you will see a button *Download Backup Now* next to the *Terminate* button near the top.

Clicking this button will perform an immediate backup and download the result to your computer. This process can take some time, and it is normal for data to download quite slowly for many minutes before it gets faster. Take care in using this feature particularly if you have a lot of data and traffic, as performing a backup while a lot of data is being saved to the database can cause a lot of slowdown.

You can also use the API to download a backup without configuring managed backups.

### 7.6.4 Restoring a backup

Restoring a backup to a Central instance will entirely replace all of its data with the backup. Please be very sure you are restoring to the right place with the right backup snapshot before
7.6. Backing Up Central

proceeding.

Note: You cannot restore a backup to an older version of Central. For example, if you create a backup from Central v1.0, you cannot restore it to Central v0.9.

1. The first thing you’ll have to do is download your backup from Google Drive, which you can do from the Google Drive website. You will find the backups in a folder called ODK Backups. Each file is a single backup snapshot, and each snapshot should be titled backup-{date}T{time}Z.zip.

2. Once you have the file on your local computer, you will have to transfer that backup snapshot file to your ODK Central server. If you don’t know how to do this, and you used our DigitalOcean installation guide, please see their instructions on how to transfer a file to a Droplet.

3. Once the file is on the server itself, you’ll need to log back into it, like you did when you first set up the server.

Tip: If you used our DigitalOcean installation steps but can’t quite remember how to do this, we suggest reviewing the login steps.

4. Now you’ll want to put it in a special place where it can be used by the restore tool: /data/transfer. If, for example, you uploaded the file to /root/backup-2018-01-01T00:00:00Z.zip, you’ll want to run this command in order to move it:

   mv /root/backup-2018-01-01T00:00:00Z.zip /data/transfer/

5. Now you need to run the restore script. Please note again that all data on this server is about to be replaced by the backup snapshot data! Anybody currently using the server will be kicked off and all changes made since the last backup will be lost. When you are sure you wish to proceed, run the following commands:

   cd
   cd central
   docker-compose exec service node /usr/odk/lib/bin/restore.js /data/transfer/backup-2018-01-01T00:00:00Z.zip 'SECRET_PASSPHRASE'

You’ll have to replace the filename following /data/transfer with your own snapshot filename, and the text SECRET_PASSPHRASE with the passphrase you typed when back-ups were first set up. If you did not set up a passphrase, immediately press Enter after you have finished putting the :file'.zip' filename in:

   docker-compose exec service node /usr/odk/lib/bin/restore.js /data/transfer/backup-2018-01-01T00:00:00Z.zip

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
6. The server will think for a while, and then print some more instructions. You will have to refresh any browser windows you have open to ODK Central to proceed. If you run into error messages at this step, please read them carefully and then seek help on the ODK Forum if you are not sure what to do.

7.7 Central Command Line Tools

Central allows some administrative actions to be performed by anybody with direct console access to the server itself. Usually, this means only the person who set up the server and installed Central onto it. These tools can be used to:

- create new user accounts,
- reset passwords,
- and manage user permissions.

All of these actions can be done through the website, and if everything is working normally we strongly recommend that you use the web interface to perform these actions. But things happen, and if something gets broken (like everybody forgets their password, or someone deletes all the users), you can use the command line tools to get things working again.

7.7.1 Getting to the tools

First, you’ll need to get to the tools. You’ll need to log into your server’s command line prompt again, like you did when you first set up the server. If you used our DigitalOcean installation steps but can’t quite remember how to do this, we suggest reviewing the section Installing Central as a reminder, or if you can’t remember your password to start at the top of that section to reset your password.

Once you have a command line in front of you (it should say something like root@server-name:~#), you’ll want to enter the following commands:

```
    cd central
    docker-compose exec service odk-cmd
```

If you see instructions appear with the section headings Usage, Options, and Commands, you’ll know you’re in the right place. If you are familiar with command line tools in general, those instructions are probably all you need to get going. Otherwise, please see the sections below for a short guide and example on how to use each one.
7.7.2 Creating a Web User by command line

If you followed our DigitalOcean installation steps, then you’ve already done this once down in this section. You shouldn’t have to do this again unless you deleted all your users. But if you do, please start by performing the steps above in the Getting to the tools section. Then, this is what you would type, assuming your email address is example@getodk.org:

```bash
docker-compose exec service odk-cmd --email example@getodk.org user-create
```

You will be asked for a password for the account, and if everything worked correctly you should see some data printed out that among other things lists the email you entered a moment ago. The next thing you’ll need to do is to make the new account an administrator, which is normally automatically done by the web interface. If you don’t, the new user will not be able to do anything.

7.7.3 Setting a Web User password by command line

You can always reset any user’s password from the website login page, which will send them an email with a link to set their new password. However, if for instance that email address no longer works, or the email is getting lost somehow, you can directly set any user’s password with the command line tools.

Please start by performing the steps above in the Getting to the tools section. Once you do, here is what you would type, assuming the email address of the account you wish to set a password for is example@getodk.org:

```bash
docker-compose exec service odk-cmd --email example@getodk.org user-set-password
```

You will be prompted for a new password. Type it in and press Enter, and if you see text that says true, the action succeeded. You can then use the website to log into that user account.

7.7.4 Promoting a Web User to administrator by command line

In the current release of Central, all users created by the website interface are automatically administrators. If you create a user using the user-create tool shown above, however, you’ll have to perform that step manually. If you do not, the user will be unable to do much of anything at all once they log in.

Please start by performing the steps above in the Getting to the tools section. Once you do, here is what you would type, assuming the email address of the account you wish to make an administrator is example@getodk.org:
Chapter 7. Setting Up ODK Central

7.8 Troubleshooting Central

7.8.1 Users aren’t receiving emails

Central uses email as a way to verify user identity when setting or changing passwords. This helps ensure that only the intended user has access to their Central account.

Email sounds like a simple technology but in practice there are many things that can cause message delivery issues. By default, Central is installed with a mail server which can be used without configuration. However, it will not work in every environment. For example:

- Many cloud providers restrict the usage of simple mail servers such as Central’s as a spam-prevention strategy
- You can be assigned an IP address that was previously used for sending spam and is therefore blocked by many mail recipients
- Your domain may not be recognized by mail recipients and therefore messages from it may be discarded or marked as spam

To address delivery issues, consider using a dedicated email service such as Mailgun. Because Central doesn’t send very many emails, using such a service will generally be a cost-effective way of ensuring email delivery. Once you have an account set up, you will need to configure Central to use it.

If you want to directly send emails from your Central installation, the mail-tester service can help you identify what barriers to email delivery you might have. Create a Central account with the email address that it provides, retrieve your results, and then delete the user. Typically, the first thing you will need to do is configure DKIM which will provide email recipients confidence that emails were actually sent by your Central server rather than by a spammer pretending to be your server.

7.8.2 Preview could not connect with server

You may run into a “Could not connect with Server” error message when previewing forms. This error message is typically because your host machine has not made its DNS servers available to Central.

To resolve this problem, first identify your upstream DNS servers. Run `cat /run/systemd/resolve/resolv.conf` to see your current list of nameservers with their IP addresses. They will look like this:
7.8. Troubleshooting Central

Now, run `nano /etc/docker/daemon.json` to make those nameservers and, optionally, the Google DNS (8.8.8.8) as a fallback available to Docker. Put the following in the `daemon.json` file.

```
{
    "dns": ["<ip1 from above>", "<ip2 from above>", "8.8.8.8"]
}
```

Finally, stop the containers, restart Docker, and bring the containers back up with `docker-compose stop`, `systemctl restart docker` and `docker-compose up -d`.

### 7.8.3 Export produces corrupt zip

If you have installed Central on a 1GB server or your forms collect many large media files, you may encounter problems exporting submission .zip files. Usually, the .zip file will end up being empty, or much smaller than expected and possibly corrupt. If you are expecting to collect media files, we recommend having at least 2GB of memory. When collecting images, we recommend specifying a maximum size in form design.

The preferred approach to addressing this is to increase the amount of memory that your server has. Instructions for doing this on DigitalOcean can be found in [this support article](https://docs.getodk.org).

If you can’t increase the memory available, you can alternately add swap. This will result in slower performance than adding physical memory but can be acceptable if it is only needed for occasional exports.

### 7.8.4 File upload fails with 413

If you get an error `413` when trying to upload a submission or when trying to upload a form attachment, the file you are trying to upload is too large. By default, files up to 100MB are accepted. We typically recommend reducing the size of the files to upload if possible. For example, images can be scaled down in form design.

If you absolutely must upload files over 100MB, you can change the `client_max_body_size` `nginx` directive:

```
cd
cd central
docker-compose stop
nano files/nginx/odk.conf.template
<modify the nginx conf value for client_max_body_size>
docker-compose up -d
```
7.8.5 Database disappeared after running Docker commands

It is possible to accidentally reset the database by running `docker-compose down`. We are working on a way to prevent this error in the future. For now, if you have run this command and your data has disappeared, you can follow these steps to relocate the data and attach it back to your server:

1. Run the following command: `docker inspect --type container central_postgres_1 -f '{{(index .Mounts 0).Source}}'`. It should print out a long name starting with `/var/lib/docker/volumes/` and ending in a long string of letters and numbers. Copy those letters and numbers and set them aside. They correspond to the location of your current (reset) database.

2. Run `docker volume ls`. This will tell you all the locations that docker has stored information. We need to find the location that contains your old data.

3. For each long string of letters and numbers you just printed out, run `file /var/lib/docker/volumes/{letters and numbers}/_data/pg_hba.conf`. So for example, `file /var/lib/docker/volumes/cd597c21c7f0920fd46001df36d454/_data/pg_hba.conf`.

4. If it tells you **No such file or directory**, move onto the next row and try again with the `file` command.

5. If it says **ASCII text**, you have found database data. But if the string of letters and numbers you just pasted is the same as what you found in step 1, it’s not the data you’re looking for. Move onto the next set of letters and numbers and try again with step 3.

6. Hopefully you found the data before you got to the end of the list. We found two sets of important letters and numbers following these steps: one in step 1 and one is step 5. Call these FIRST and SECOND, respectively.

7. Now to restore the data, you’ll want to run the following commands:

```
cd
cd central
docker-compose stop
pushd /var/lib/docker/volumes
mv FIRST postgres.data.bak
mv SECOND FIRST
popd
docker-compose up -d
```

Go to your site in a browser and try to log in with an account that previously existed. If that doesn’t immediately work, try doing another `docker-compose stop` followed by
7.8. Troubleshooting Central

docker-compose up.
Chapter 7. Setting Up ODK Central

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
8.1 Managing Users in Central

There are two types of user accounts in ODK Central: Web Users and App Users.

- **Web Users** have accounts on the Central management website. These accounts are global across all projects on the server. They can log into the web interface and perform administrative actions like user management, form upload and management, and submission data viewing and download. They can also fill and submit Forms directly in their web browser.

- **App Users** can use mobile data collection apps like ODK Collect to connect to Central. App Users are limited to a single project at a time. Once connected through the app, they will be able to see the list of forms, download the ones they need, and upload completed submissions to those forms.

You will need both types of users in order to run a successful data collection project: a Web User must upload a valid form definition, an App User must upload submissions to it from their mobile device, and the Web User will then be able to see those submissions in the web interface and download them for analysis.

8.1.1 Web User Roles

Central features Role-based User permissioning. In the current release of Central, we provide four roles: Administrator, Project Manager, Project Viewer, and Data Collector. In a future release, you will be able to define your own roles as you see fit.

By default, Central roles are configured to allow the following:

<table>
<thead>
<tr>
<th>Action</th>
<th>Administrator</th>
<th>Project Manager</th>
<th>Project Viewer</th>
<th>Data Collector</th>
</tr>
</thead>
<tbody>
<tr>
<td>Projects</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Continued on next page
### Managing Web Users

You can make Users into Administrators from the *site-wide Users panel*, and you can assign them as Project Managers or Viewers on the *Settings tab* within the Project.

To manage web users, navigate to → *Users* → *Web Users* at the top of the Central management website. You should see a listing of users that looks like this:

<table>
<thead>
<tr>
<th>Action</th>
<th>Administrator</th>
<th>Project Manager</th>
<th>Project Viewer</th>
<th>Data Collector</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Edit Details</strong></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Archive</strong></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Project Forms</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>List All</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Edit Attachments</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Edit Details</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Set State</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delete</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Project Form Submissions</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Edit &amp; Review</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Comment</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>View &amp; Download</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>OData Access</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td><strong>Project App Users</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>List All</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revoke Access</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>See Code</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Web Users</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Create</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>List All</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Edit Details</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>View Email Addr.</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revoke Password</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delete</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Server Configuration</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Setup Backups</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stop Backups</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
8.1. Managing Users in Central

Creating a Web User

To create a new Web User, click on the Create web user button on the right side of the Web Users listing page. You will see a popup that looks like this:

To create a new Web User, input the email address of the person who should receive access. Press Create once you are satisfied with the email address.

That email account will shortly receive an email with the subject line "ODK Central account created". If you do not see the email, check your spam folder. In the email, there will be a link which will allow the recipient to set a password for their new account, after which they will be able to log in.

The link is only valid for 24 hours. If 24 hours pass and it has not been used, you should use the Reset Password tool to send them a new link.
Chapter 8. Using ODK Central

Newly created Web Users are only able to log in and edit their profile information. In order to give them access to do useful work on the server, please read the following section.

Assigning a site-wide Web User Role

As mentioned under Web User Roles above, there are four Roles you may assign to Web Users in the current release of ODK Central: Administrator, Project Manager, Project Viewer, and Data Collector. Administrators may perform any action on the system, while Project Managers may perform any action on their assigned Project(s). Project Viewers may only see created forms and submissions within the Project they are assigned to, and cannot edit anything. Data Collectors may see the list of all Forms in the Project, and retrieve basic information about them, but may only create new Submissions. They cannot see any Submissions.

To learn how to assign a Project Manager or Viewer role, please see the Managing Project Roles section in the Projects guide.

To assign an Administrator role, navigate to the Web Users administration panel. There, you should see a table like this one:

<table>
<thead>
<tr>
<th>Email Address</th>
<th>Sitewide Role</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="mailto:alice@.opendatkit.org">alice@.opendatkit.org</a></td>
<td>Administrator</td>
<td><img src="image" alt="Dropdown" /></td>
</tr>
<tr>
<td><a href="mailto:bob@.opendatkit.org">bob@.opendatkit.org</a></td>
<td>Administrator</td>
<td><img src="image" alt="Dropdown" /></td>
</tr>
<tr>
<td><a href="mailto:chelsea@.opendatkit.org">chelsea@.opendatkit.org</a></td>
<td>Administrator</td>
<td><img src="image" alt="Dropdown" /></td>
</tr>
</tbody>
</table>

Under the Sitewide Role column in the table, you will see dropdown inputs with the options Administrator and None. To make a Web User an Administrator, change the dropdown next to their name to Administrator. You will see the page think for a moment, and then it will inform you that the action is done. To take away Administrator rights from a Web User, change the dropdown to None.

You will not be able to change your own Role in the system. To change your own Role, you will need to get somebody else to log in and change it for you.

Resetting a Web User password

Any user may request a reset of their own password by using the link at the bottom of the login screen:
After submitting the reset form, the user should receive an email with the subject line ”ODK Central account password reset”. If they cannot find it, they should check their spam folder. When resetting a password this way, the user’s current password continues to function until they actually use the link in the email to set a new one.

We also provide a separate way for administrators to directly reset any Web User’s password in the administration panel for two reasons:

1. In case the user’s password has been stolen and needs to be disabled immediately.

2. In case the user does not know how to do this themselves.

With the administrative reset, the user’s password stops working immediately and they will be completely unable to log in until a new one is set. They will receive an email with instructions and a link on how to do this exactly as shown above. To perform the administrative reset, navigate to the Web Users listing page, and use the Actions menu at the right side of the table:
Retiring a Web User

When you retire a Web User, their login access will be revoked and they will be immediately signed out everywhere. They will disappear from the Web Users management list, but any records that trace their actions (submission uploader or form creator name, or audit log action initiator, for example) will still show their information.

If a retired Web User attempts to reset their password, they will receive a special email explaining that their account has been retired.

To retire a Web User, find them on the Web User administration panel, and open the Actions menu:

From here, select Retire User and follow the on-screen instructions.

8.1.2 Managing App Users

App Users never gain any access to the management website: they do not have email addresses or passwords associated with their account, only a nickname so you can tell which is which. Once a Web User creates an App User within some project, a configuration QR Code will be generated which will grant a mobile device access to that project as that App User. Access can be revoked at any time, and Web Users can see which App Users uploaded which submissions.

A newly created App User does not have access to any Forms. To give them access once they are created, use the Form Access tab on the Project. You will be able to allow access to particular Forms within the Project for each App User.

Tip: In version 0.6 and earlier of ODK Central, all App Users were granted download and submission rights on all Forms within their Project. These users retain their access when you first upgrade to version 0.7. Once you have version 0.7 installed, you can adjust these Users’ access per form.

To manage App Users, navigate to the project whose App Users you wish to manage, and then click on the App Users tab just below the project name. You should see a listing of
users that looks like this:

Creating an App User

To create a new App User, click on the *Create app user* button on the right side of the App Users listing page. You will see a popup that looks like this:

Once you provide a nickname for the user (usually the name of the data enumerator who will carry the mobile device works well), click *Create*. The user will be created, and you will see a screen that looks like this:
That App User has now been created and granted access to use their mobile device to list, download, and submit to all available forms within their project. To do so, however, their mobile device will have to get set up with this new account. That is what the QR Code you see on this screen is for. Read on to the next section to find out how to use it.

**Configuring an App User mobile device**

A mobile device will need to be configured to access your ODK Central server as a particular App User in order to gain access to the forms and upload submissions within their project. This is done by way of the Collect Settings QR Code.

The QR Code contains information about how to find your ODK Central server, and how to prove to the server that the mobile device belongs to a valid App User. In future versions of ODK Central, it will be possible to specify other settings to be imported to the device as well.

There are two ways to access the QR Code for an App User. The first is in the second step of the *App User creation wizard*. Please find the second screenshot in the previous section to see what this looks like. If you close out of this wizard, you can still access the QR Code by clicking on the *See code* link in the listings table:
8.1. Managing Users in Central

If instead of a See code link you see text that says Access revoked, that App User no longer has access to the server. Create a new App User if you need a new QR Code.

Once you have found the QR Code, you will be able to use it to configure ODK Collect. Please see the section on importing settings into Collect to learn how to do this.

Revoking an App User

You may wish to revoke an App User’s access, for instance if their QR Code has been stolen or if they have left the organization. To do so, navigate to the App Users listing page, and use the Actions menu at the right side of the table:

App Users whose access has been revoked will still appear in the App Users listing table, and will still be visible as the submitter of any submissions they uploaded. However, they no longer have a valid QR Code with which they can configure an ODK Collect installation, and any mobile devices already configured with their code will no longer have access to the project.
8.2 Managing Projects in Central

New as of version 0.4, almost everything in ODK Central is organized by Project. Forms, Managers, and App Users are all partitioned by project. Both on the administration website as well as on a mobile device (within ODK Collect, for example), access to each project and its forms can be managed person-by-person. Only the Central administrative staff can create and grant initial access to projects.

8.2.1 Projects Overview and Roadmap

Projects are a work in progress that we will add to and improve over the next few releases. If you are working on ODK Central version 0.4, projects work like this:

- All forms must belong to a project. It is not possible to create a form that is not part of a project. It is not possible to assign one form to multiple projects (though the same form may be uploaded to as many projects as desired).
- All App Users (who may access Project Forms through mobile device applications like Collect) must also belong to a particular project.
  - When a user connects to a project through Collect, they will only be able to see and submit new data to forms within that project.

If you have upgraded to version 0.5, the following changes have been made:

- Only Web Users (who are allowed to administer ODK Central through the management website) marked Administrator have administrative access to the main site settings and all projects.
  - However, all Web Users created before version 0.5 were already marked Administrator. You will have to go change their Site-wide Role to None if you wish to remove these privileges.
- Normal Web Users who are not Administrators can be made Managers on select projects. These users have the ability to manage anything about the project. They can:
  - Appoint other project managers.
  - Create, manage, and archive forms on the project.
  - Access all submitted data within the project.
  - Create, manage, and retire App Users for the project.
  - Manage and archive the project itself.

Since the major changes that occurred in version 0.5, we have additionally made the following improvements:
8.2. Managing Projects in Central

- Better, centralized form state and access management.
- More granular project access, in the form of a Project Viewer role which allows read-only access only to Forms and Submission data.

In future releases, we have a loose roadmap with at least the following goals:

- Change the relationship between Collect and Central so that with one button you can synchronize the mobile device to some centrally managed desired state.
  - So for instance, you can decide which forms should be available on Collect within Central once, and within Collect just press "synchronize" to update to that state.
  - Eventually, Collect app settings may be synchronized this way as well.
  - And eventually, different App Users of different roles may be assigned different device states.
- Download an entire project’s data at once.

If you have ideas on how projects might be made more useful for you, please do not hesitate to leave us feedback on the ODK Forum.

8.2.2 Migrating to Projects

If you have an ODK Central server installed which is version 0.3 or earlier, your existing data will be migrated into a project when you update.

All of your existing forms and App Users will be placed into a project entitled "Forms you created before projects existed". There are no longer site-wide App Users, so that panel will no longer be present.

**Important: Upgrading to version 0.5**

Mobile clients like Collect which were configured to access version 0.3 will continue to work without intervention for version 0.4, but they must be reconfigured with a new QR Code access token from Central before migration to 0.5, because for version 0.5 this backwards compatibility will be removed.

If you do not refresh the access tokens before upgrading to 0.5, you will have to reconfigure the device with a new QR Code from 0.5 in order to restore the connection.

8.2.3 Creating a Project

Only ODK Central administrators may create projects. To create a project, navigate to the Projects section of the Central management homepage, and click on the New button.
You will be asked for a name for your new project. When you provide one and press Create, your project will be created and you will be taken to its management page.

### 8.2.4 Managing a Project

You can find all the projects you have access to in the Projects section of the ODK Central management website homepage.

When you click on its name, you will be taken to its management page.
8.2. Managing Projects in Central

Here, you will find some basic details about the project, and a listing of all the forms in the project. You can click on any form name to manage that form and view its submission data.

**Editing Project Settings**

To edit Project Settings, first navigate to the Project, then click on the *Settings* tab underneath the Project name.

From here, you will be able to edit the Project Name. You will also see a section for Archiving a Project, which is described in more detail below.
Managing Project Roles

Any Web User may be assigned as a Project Manager or Project Viewer on a Project.

Project Managers may perform any action upon and within that Project, including changing its name, adding more Project Managers, and uploading and managing Forms and Submissions. Any Web Users that are site-wide Administrators will already be able to perform these actions on any Project without being explicitly named a Manager.

Project Viewers can view basic information about all Forms in the Project, and download Submission data or access it over OData for analysis. They cannot make any modifications to any data or settings.

Data Collectors may view the list of Forms, retrieve basic information about them, and submit new records to any Form in the Project.

You will find a detailed breakdown of user roles here.

To assign or remove Managers, Viewers, or Collectors for a Project, first go to the Project overview page, then click on the Project Managers tab under the Project name. You should see the following page:

If roles have not already been assigned to the Project, the table will be empty. This is normal: the table only shows Users with assigned roles on the Project at first. To find a Web User to assign them a role, search for them in the Search for a user field above the table. You can find users by their Display Name or their Email. Type part or all of either into the box, and press Enter. The search results will appear in the table.
8.2. Managing Projects in Central

To make a Web User into a Project Manager, Viewer, or a Data Collector, change the dropdown next to their name in the Project Role column from None to the desired role. You should see the page think for a moment, and then a confirmation of success. If you clear the search in the text box, the newly assigned user should remain.

To demote a Web User from any role, change the dropdown back to None.

Managing Project App Users

To manage App Users for a Project, you can navigate to the Project overview page, then click on the App Users tab under the Project name. For more information about creating, managing, and retiring Project App Users, please see this section.

Managing Form Access

Right now, Central offers two ways to control around Form Access within each Project:

- Each Form’s Lifecycle State controls whether any App User can download and/or submit to that Form. Near the end of a Form’s life, for example, it makes sense to disallow downloading the Form as a blank, but still receive any submissions that have already been created.

- Access to download and submit each Form can be customized per App User associated with the Project. When first creating a Form, for example, it makes sense to only allow a testing user access to the Form so that one can be sure that it works before rolling it out to all users.

We place these access controls for all Forms in a single place, on the Form Access tab at the Project level. To access it, navigate to the Project and select the tab at the top of the page labeled Form Access.
Chapter 8. Using ODK Central

On the left side of the Form Access page, you will find a list of all the Forms in the Project, along with a dropdown selection to set the Form Lifecycle state for each one. Along the top, you will see all active App Users in the Project. At each row/column intersection, there is a checkbox that governs whether each App User is allowed access to each Form.

**Tip:** You may see a pencil icon next to the Lifecycle state dropdown on some Forms. This means that those Forms are currently Drafts, with no published version. They will not be visible to data collection clients no matter the Lifecycle state setting, or the checkboxes on this page. Once a Draft Form is published, then the settings on this page will immediately take effect.

If you are having trouble recalling what each Form State means, the ? icon in the header will give you a quick recap:
8.2. Managing Projects in Central

As you make changes to Form States and App User access, they will be highlighted in yellow. You can make all the changes you’d like to apply at once (for example, marking an old version of a Form as Closing while granting Open access to the new one), and once you are satisfied with what you see you can click the Save button at the top-right of the screen to apply them all at once.

**Tip:** When you first create an App User, it will not have access to any Forms. When you first create a Form, no App Users will be allowed to access it.

**Archiving a Project**

When you Archive a Project, it will appear at the bottom of the Project List on the home-page, with *(archived)* added onto the end of its name.

In version 0.6 of Central, archiving a Project would disable certain features on it. We have eliminated this behavior, so all your archived Projects can still be used and manipulated freely.

To Archive a Project, first navigate to the Project, then click on the *Settings* tab underneath the Project name.
Chapter 8. Using ODK Central

8.3 Managing Forms in Central

Forms are at the heart of ODK. Once fetched onto a mobile device from ODK Central, they define the questions, validation, and logic to be presented during data collection. They also serve to organize the data coming back from mobile devices.

8.3.1 Forms in ODK Central

In ODK Central, there are four important tasks you’ll perform while managing a data collection project:

1. **Form drafting** is where you design the Form itself, laying out many details like the questions and acceptable responses. ODK Central itself does not help with form design: instead, please take a look at the many available tools and the introduction to form design for help creating your form. Central does, however, help with testing your design, with Form Drafts. When you first upload your Form, it will be a Draft to which you can connect a mobile device and submit test data. If you are not satisfied with the Form, you can adjust the design and upload a new version in place of the old Draft. Once you are satisfied, you can Publish the Form to make it ready for use.

2. **Form lifecycle** settings allow you to control who can access and submit to particular Forms. These tools can help manage user permissions, but they can also help phase out outdated Forms as they come to the end of their life. You can control these using the Form Access tab on the Project.

3. **Data extraction and analysis** can occur either at the end of the project, or continually as the project runs. Either way, ODK Central provides several different methods
8.3. Managing Forms in Central

for extracting and analyzing your submission data. This is covered in the *Form Submissions* article.

4. **Form updates** allow you to make changes to a Form while it is already in use. Central allows you to create a new Draft of any Form at any time. As with initial Form creation, this Draft allows you to test your changes in a staging environment where it will not affect the live Form or its data. When you are satisfied with the Draft, you can again **Publish** it, and it will replace the live version.

Drafting and version updates are new as of Central 0.8. We will cover all of these topics in the sections below.

8.3.2 Uploading a form to ODK Central

As mentioned, ODK Central does not feature a built-in form design utility. Please take a look at the many *available tools* and the *introduction to form design* for help creating your form.

As of version 0.7, Central will accept either XForms `.xml` or XLSForm `.xls/.xlsx` files. If you have written your own `.xml` file, we strongly recommend that you first double check that it is valid using *ODK Validate*. If you upload an XLSForm spreadsheet, Central will convert it to an XForm for you, and it will be automatically validated as part of that process.

Once you have your form file, the next step will be to upload it into ODK Central. To do this, navigate to the Project (click on its name from the ODK Central homepage) to which you would like to add the Form, and locate the Forms listing section at the bottom of that page:
From there, click on the *New* button next to the section header, and you should see a popup appear:

You can either click on the *choose one* button to browse for your `.xml`, `.xls`, or `.xlsx` file, or if you already have it handy somewhere, you can drag it over the gray box and drop it to choose it. Either way, once you have chosen your file (you will see the name of your file at the bottom of the gray box when you do), you can click on the *Create* button immediately below to upload the form.
8.3. Managing Forms in Central

Some errors you may see:

- If you are uploading an XLSForm, and the converter flags warnings with it, your form will not be immediately created. You’ll be shown the warnings, and given the option to either ignore them and create the form anyway, or else you can fix the issues and start over by uploading a new file.

- You may see a message that reads **A resource already exists with xmlFormId value(s) of xyz**. If you do, there already exists a form within this project with the same unique designation. If you are using XLSForm, try changing the name of the file or the form_id in the settings sheet. If you designed the form by hand, please check the id="..." attribute immediately inside the <instance> tag.

- You may see a message that says **A form previously existed which had the same formId and version as the one you are attempting to create now. To prevent confusion, please change one or both and try creating the form again.** This means there once was a form within this project that has since been deleted that has exactly the same formId (see the previous bullet point) and version designation as the one you are now trying to upload. Central won’t accept the new form, because this conflict could cause confusion with mobile devices that still have the old form sitting around. To upload this form, change either the formId (again, see the previous bullet point) or update the version and try again.

Once the form is successfully uploaded, you will be taken to the Form Draft page. It will not be accessible to data collection clients until you publish the Draft, which we will cover in the following section.

8.3.3 Working with Form Drafts

Form Drafts, available as of Central 0.8, provide a way to safely and easily verify the design of your Form before you make it available for use. Drafts are accessible only to privileged Project staff. Each Form Draft has a unique access token which allows configured data collection clients to submit test submissions to the Draft. These test submissions disappear automatically when the Draft is published. Once a Draft is published, it is available for use according to the access rules you have specified in the Form Access tab on the Project.
The Draft Status page gives insight into the current status of your Draft, and provides controls for managing it.

On the left, you will find the Draft Checklist, which suggests the steps you might take before publishing your Draft. On the right are details about the currently uploaded Draft version of the Form, including its current version string, and actions you may take on the Draft:

- The Upload new definition button will allow you to upload a new Form definition, which will replace the current Draft version. When this happens, all test submissions will be erased. If you have uploaded Media Files, Central will attempt to preserve any that match the new definition.

- The Publish Draft button will publish the Draft, making it available for use according to the access rules you have specified on the Form Access tab on the Project. Any test submissions you have made will be erased.

- The Abandon Draft button will delete the Draft. When there is not yet a published version, this will delete the entire Form. If the Form has been published, only the Draft will be deleted.

Tip: When a form is first created, none of the existing App Users on the project will be able to access it for download or submission, even once the Form is published. Once you are ready to allow App Users to access the form, use the Project Form Access tab.
When you first create a new Form, the navigation tabs on the left will not be accessible. They pertain to the published version of the Form, and will become available once you publish your Draft. The tabs on the right, within the gray Draft section, relate to the Draft.

If your Draft requires Media Files, there will be a checklist step asking you to upload them, and a Media Files tab at the top of the page. See the next section *Forms With Attachments* for more information about uploading and managing attachments.

The *Testing* Draft tab shows test submissions that have been made to the Draft, and instructions for doing so:

At the top of the page are instructions and a QR Code which will configure a mobile device to submit to the Draft Form. For help configuring a mobile device, please see *importing settings into Collect*. The table below these instructions contains any test submissions that have been made to the current Draft. For help with this table or exporting test data, please see *Form Submissions in Central*.

### 8.3.4 Forms With Attachments

If your Form Draft references any external files (images, audio, or video included as part of your question prompts, or data lookup files used to populate selection lists), Central will see this and open up some additional displays and controls you will need to provide those external files:
If you see this extra **Upload Form Media Files** checklist step and **Media Files** tab at the top of your Form Draft checklist, then Central believes you need to upload some files associated with this form. If the checklist step has been checked off, then you’ve already completed this task: great work! Otherwise, click on the → **Media Files** tab at the top to see what files you’ll need to provide.

This form design references three files that we’ll need to provide, one of which we’ve already uploaded. You can see the name and expected type of the file in the table, as well as when the file was last uploaded. You can click on the name of any uploaded file to download what Central has for it. To upload a new one, you’ll want to drag-and-drop one or more files onto
8.3. Managing Forms in Central

the table.

On File Types and Contents

While Central will detect the type of file the form design expects, and will verify that the name of any uploaded file matches one that is expected, Central will not double-check the type of the file, nor the contents of the file for you. So, just because Central accepts your file does not necessarily mean that it will work correctly.

Once you publish a Draft, you will not be able to modify the Attachments associated with it without creating a new Draft.

Bulk-uploading Many Attachments At Once

If you select and drag many files at once onto the table, Central will automatically try to match each file with a name in the table. You’ll have a chance to see what it came up with and confirm that things look okay before the upload begins. You will see a warning if one or more of your dragged files don’t match any of the expected names.
Uploading One Attachment

If you drag a single file onto the table, you’ll have the option of which table row you’d like to upload that file into. This way, if the file isn’t named exactly what Central expects, you can still upload a file into that slot without having to rename it on your own computer. But if the file does have the appropriate name, you can drop it somewhere other than a specific slot (for example, just below or just above the table) to have Central match it up with the correct slot automatically.

8.3.5 The Form Overview page

Here, you can get a brief summary of the status of your form, and recommended next steps. You are automatically taken here when you publish a Form Draft or click on the form name in the Form listing page. You can also get back here from other form-related pages by clicking the → Overview tab below the name of the form.
8.3. Managing Forms in Central

The documentation on this page is a more detailed introductory explanation of form management in ODK Central, but the checklist you find on the Overview page is tailored to the current status of your form and your project and is a great place to look when you aren’t sure what to do next.

In the future, look forward to seeing even more useful information at-a-glance on this page.

### 8.3.6 Seeing Form Submissions

To see the current submissions uploaded to Central for a form, you can click on the → Submissions tab below the name of the form. Here, you will see a summary table of all known submissions, and you will find multiple options for downloading and analyzing your submission data. This page and these options are covered in more detail in the Managing Form Submissions in Central article.

### 8.3.7 Managing Public Access Links

Public Access Links allow broad web-based distribution of a Form for direct responses from subjects. They can be used to conduct large remote self-administered data collection campaigns, or allow direct surveying of the public at large. Respondents will fill the Form directly in their web browser.
For more information, please see the *relevant section* in the Submission article.

### 8.3.8 Managing Form Lifecycle

Forms can be in one of three lifecycle stages: **Open**, **Closing**, and **Closed**. All forms start in **Open** state when they are first created. You can see what each of these means below:

<table>
<thead>
<tr>
<th>State</th>
<th>Available for download to apps</th>
<th>Accepts new submissions from apps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Closing</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Closed</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

As you can see, you can use the **Closing** state to prevent further distribution of a form while still allowing the final few submissions to come in, while the **Closed** state effectively turns the form off completely. You can always set the form lifecycle stage to whatever you want: you can always, for example, re-open a closed form.

To set the form lifecycle stage, go to the **Form Access** tab for the Project, under the name of the Project at the top of the page. You may have to navigate back out of the Form first by clicking on the **Back to Project Overview** link at the top of the page. Here, you will find the three possible stages in a dropdown for each Form on the left side of the page. Select the ones you want for each Form, then click **Save** at the top-right to save the changes.

You can find more information about the Form Access page [here](#).

### 8.3.9 Updating Forms to a New Version

As of Central 0.8, it is possible to update a published Form with a new definition, or new Media Files, and to test these changes before they are applied to the Form in use.

There is one primary restriction Central enforces on updated definitions: once defined in a published Form version, each field Data Name (in technical terms, the Instance XPath) cannot change its Data Type. Unused fields may be removed, and new fields may be added, but if any field reuses a previously existing Data Name, it must have the same Type as it did before. If you run into an error with this restriction, the easiest solution is usually to rename the changed field to a new name.

To begin the process of updating a published Form, click on the **Create a new Draft** button in the Draft navigation on the Form:
8.3. Managing Forms in Central

Initially, the new Draft will have the same definition as the published Form. If you only want to update attachment Media Files, this means you don’t have to bother uploading a definition at all: you can go straight to the Media Files tab and upload the changed files.

You can replace the Draft definition, Media Files, and make test submissions as with the initial Form Draft before the Form was first published. Test submissions will not interfere with published Form submissions.

Once you are satisfied that your updated Form is ready to be published for immediate use, you can click on the Publish Draft button on the Draft Status tab.

Form Version naming

If you did not change the definition, or your updated definition did not change the version of the Form, Central will not be able to publish the Form as-is. This is because the version must change in order for data collection clients to understand that they should update. You can upload a new definition with a changed version, or else Central will offer to change it in-place for you.

Once the Draft has been published, it becomes the version in use and there will no longer be a Draft associated with the Form.

What happens to my submissions?

When a new Form version is published in place of an old one, all the previous submissions continue to exist, and will export along with all your data over Zip download or OData. However, only the current Form definition will be used in that export: if, for example, you have deleted a field that used to exist, that field will not appear in the export.

Draft testing submissions will never export with your final data, and only exist as long as the Draft does. If you delete, publish, or replace your current Draft, all test submissions will be cleared away.
8.3.10 Accessing Older Form Versions

If you have published multiple version of a Form, you can see each of them under the `Versions` tab.

Each published version of the Form will be listed, along with actions to download the definition of each Form. In future versions of Central, the Media File attachments associated with each version of the Form will be downloadable as well.

8.3.11 Deleting a Form

Tip: If you only want to turn the form off so that it doesn’t appear to users of mobile data collection apps, we suggest using the `form lifecycle controls` explained above.

If you are certain you wish to delete a Form, you can find the option on the Form Settings page: click on the → `Settings` tab under the name of the form at the top of the page. On the right side of this page, you will find the `Delete this form` button.
Once a Form has been deleted, it will remain in the Trash for 30 days before being permanently deleted. You can find the Trash on the Project page, under the Forms list. Here, you can undelete a Form using the button on the right. Undeleting a Form will restore it exactly as it was when deleted.

After 30 days, when a Form is permanently deleted, the data will be removed from the system completely.

8.4 Managing Form Submissions in Central

The most common way to use ODK Central is in conjunction with a data collection client, typically on a mobile device, such as ODK Collect. To do this, you will need to connect to it from your mobile device, after which you will be able to uploaded submissions back to Central.

ODK Central also bundles Enketo, which enables preview and submission of forms directly from a web browser. Please note that as with all ODK clients, Enketo does not always behave quite the same as Collect, or support the same features. Any authorized Web User may fill out a Form directly from the browser, as will be described in more detail below.

Finally, ODK Central offers Public Access Links. A Public Access Link grants anybody in possession of the link the ability to submit to a Form on your server. You can control whether each respondent can submit more than once, and revoke access from any Link at any time.

Submissions sent to Central are available to browse in a preview table, to connect directly to data analysis tools, and for download.

8.4.1 Direct Web Browser Submissions

Web Users who are Administrators, Project Managers, or Data Collectors can directly fill Forms in the web browser from the Central administration website. This functionality is provided by Enketo, which does not always behave quite the same as Collect, or support the same features.
Administrators and Project Managers can begin a survey by going to the Submissions tab of the Form, and clicking on the New button next to the Submissions header. This will open a new tab which will load the Form in Enketo.

Data Collectors do not have access to the detailed Form management pages. Instead, they will find a Fill Form button next to the Form name in the list of Forms on the Project Overview page.

### 8.4.2 Public Access Links

ODK Central allows the distribution of surveys to a broad or open respondent group using Public Access Links. These Links take recipients directly to the Form in their web browser, administered by Enketo.

To create a Public Access Link, go to the Form’s Public Access tab. Click on Create Public Access Link... to begin.

In the window that appears, you’ll need to name the Link. This name is for your own identification purposes in the administration website, and is not displayed to respondents.
8.4. Managing Form Submissions in Central

You’ll also need to decide whether to allow multiple submissions per respondent. Normally, respondents filling a Form through a Public Link will be redirected to a thank you page after sending a Submission. Pressing the back button will not bring them back to the Form but they could send in another Submission by visiting the Link again. Checking the Single Submission checkbox enables basic protection against more than one Submission being made from the same browser.

Single submission enforcement

In Enketo, the enforcement limiting each respondent to a single response is done with in-browser tracking. This means that a user could submit multiple times using different devices or browsers, or distribute the link beyond the intended group.

Also because of this tracking method, respondents will only be able to respond once per Form, not once per single-submission Link. Future versions of Central may change how this works. Please leave feedback on the community forum if this is something you’d like to see.

Once a Link is created, it will appear in the table, along with a web address you can copy and paste to distribute the Link to respondents.

You cannot yet edit any of the details of a Public Link. This will come in a future version of Central.

Revoking a Link

You can revoke a Link at any time to prevent any further Submissions through it. Once a Link is revoked, all Submissions will be immediately denied, and new attempts to load the Form using the Link will result in an error instead.

To revoke a Link, click on the Revoke button in the Link’s row in the table. You will be asked to confirm the action. Once a Link is revoked, there is no way to restore it.
8.4.3 Accessing Submissions

To find the Form submissions page, first find the form in the Form listings page (→ Forms) and click on it. You will be taken to the Form Overview page for that form. Click on the → Submissions tab below the form name to find the submissions.

The table preview you see here will at first show you the first ten fields of your survey and their results, with the latest submissions shown closest to the top. Any downloadable files will appear with a green download link you can use to directly download that media attachment. The submission’s instance ID will always be shown at the right side of this preview table.

If your form has more than ten fields, you can show more columns by accessing the Columns shown dropdown and checking the columns you wish to see. While the Columns shown pane is open, you can use the search box along its top to search for a particular column name if you have many.

In the State and actions column, you will see the current review state of each submission and the number of edits that have been made, if any. If a submission is missing expected media uploads, you will see a warning here. When you hover over a row (or tab to it with your keyboard) you will see controls in this column to edit the submission, or see more details about it. You can read more about review states and the submission detail page below.

You can limit the rows that appear by the submitter, the date, and the review status. These filter controls are available just above the submission table.

Any filter you apply to the submission table also applies to the download button. To work with your data, you can download it from Central. Right now, you can do this in one of two ways:

1. The CSV Download option will get you a .zip file containing one or more .csv tables, along with any multimedia submitted to the form. This is a good option if you already have custom tools you wish to use, or you want to import it into an offline
8.4. Managing Form Submissions in Central

analysis tool like SPSS or Stata.

2. The **OData connector** allows you connect a live representation of the data to OData-capable tools like Microsoft Excel, Microsoft Power BI, Tableau, SAP, and others. This option has some advantages: the data is transferred more richly to maintain more data format information, and the feed is always live, meaning any analysis or reports you perform in your tool over an OData connection can be easily refreshed as more submissions come in.

When the rows of the table have been filtered in any way, that filter also applies to the downloads. This makes it possible to download partial sections of your data at once.

Learn more about these options below:

8.4.4 Downloading submissions as CSVs

To download all submission data as `.csv` tables, click on the *Download all # records* button on the right side of the listing page:

If you have any row filters applied to the submission table, those filters will be applied to your download as well. You can use this to, for example, download only submissions from a particular month, or only approved submissions.

Once the download dialog opens, you’ll be given some additional export options.
Some of the options may be disabled if they do not apply to your data, or if they are not available due to features that you have enabled (such as managed encryption).

- The option to split *select multiple* choices will create a new column in the export .csv for each unique known value in each select multiple field. These columns then indicate whether each submission checked each option.

- The remove group names option takes out the prefix usually added to groups in the header: so for example, `meta-instanceID` would become just `instanceID`.

- Finally, the option to include previously deleted fields will include every known previously deleted field in any version of the Form in the export, along with any data found for those fields.

Click on one of the format options on the right to start the download.

Once the .csv or .zip completes downloading, you will find one or more files when you extract it:

- A root table .csv named after your Form title.
- Join table .csv files representing any repeats you may have in your form, with join columns on the left of each table relating each row to its counterpart in the parent table. Each join table is named to reflect its relationship with the others. If there is only one .csv file, then your form has no repeats.
- A folder named `files` which contains subfolders, each named after an `instanceId` of a submission. Each subfolder then contains a set of file attachments relating to that submission. If no `files` folder exists, then no multimedia attachments have been submitted to this form.
- If you have enabled *Client Audit Logging* on your form, and log events have been submitted to the server, then you will find a file that ends with - audit.csv. This
8.4. Managing Form Submissions in Central

File combines all the logging data from all submissions to the form into a single table.

Tip: Excel will not import CSVs with Unicode characters like à, ß, and ç correctly if you double-click the file or open it from the File menu. You must use the Text Import Wizard and specify a file origin of Unicode (UTF-8, 65001) and the comma delimiter.

Rather than downloading CSVs manually, you can also connect Excel directly to Central via OData and get a live-updating spreadsheet with auto-detected data types and Unicode support.

8.4.5 Connecting to submission data over OData

To connect a third-party tool to Central over OData, you will need a link to paste into the tool. You can find this link by clicking on the Analyze via OData button on the right side of the listing page:

Once you click on it, you should see this popup appear:
To connect with Excel or Power BI, follow these steps.

1. Start the Get OData feed action.
   - In Excel, select Data menu item, then Get Data, From Other Sources, then From OData Feed.
   - In Power BI, select the Home menu item, then Get Data, then OData feed.
2. Copy and paste in the link from Central and then select OK.
3. Switch to Basic authentication, enter your Central credentials, and then Connect.
4. The Navigator window now appear. Select Submissions, then Load.
Tip: See Import external data into Excel and OData feeds in Power BI for more information.

If you want to use the free and popular R statistics and analysis tool, we recommend you use ruODK. A guide for getting started with it can be found here. ruODK is developed and supported by community members. If you wish to help improve it, you can find information on GitHub.

Tip: If you are having trouble getting Power BI or Excel to connect, and especially if you see error messages about permissions or authentication, clear your cached permissions and try again.

You can also access the OData feed yourself. The OData feed is an easily consumable JSON data format and offers a metadata schema, some filtering and paging options, and more. To learn more about the OData feed, click the API Access button or see the developer documentation directly.

Setting up Media Downloads

For a lot of reasons, it can be tricky to access submission media files while doing data analysis. Getting an analysis tool to fetch data from Central does not mean it can or knows how to get images, video, and other media.

In OData data responses, you will see media files given by their filename. If you want, you can construct a link within your analysis tool that will download any media file with your web browser. You can do this by gluing together pieces of text into a URL. Often this gluing operation is called concat or concatenate. You’ll need to make it look like this:

```
https://DOMAIN/#/dl/projects/PROJECTID/forms/FORMID/submissions/
<INSTANCEID>/attachments/FILENAME
```

Where the uppercase words need to be replaced with the appropriate values. The easiest way to get the DOMAIN, PROJECTID, and FORMID is to open the Form in your web browser in the Central administration website and just copy the values you see there. The two web addresses are quite similar. Then you have to add the INSTANCEID and the FILENAME, both of which you can find in the OData data itself.

Here is an example of a completed address:

```
https://my.odk.server/#/dl/projects/1/forms/forest_survey/submissions/
<uuid:20bcee82-4a22-4381-a6aa-f926fc85fb22>/attachments/my.file.mp3
```

This location is a web page that causes a web browser to download a file. It cannot be used directly to embed images or video on any website or application.
8.4.6 Submission Review States

As of version 1.2, Central allows Project Managers and Administrators to review submissions and assign them certain states. This feature lets you perform verification and follow-up data editing within Central itself, if you need this kind of a workflow. The available states are:

<table>
<thead>
<tr>
<th>State</th>
<th>Assigned by</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Received</td>
<td>System</td>
<td>The default state for all incoming submissions, assigned automatically by Central</td>
</tr>
<tr>
<td>Edited</td>
<td>System</td>
<td>Automatically assigned by Central whenever a submission is edited by any user</td>
</tr>
<tr>
<td>Has Issues</td>
<td>User</td>
<td>Can be assigned by project staff if a submission has problems</td>
</tr>
<tr>
<td>Approved</td>
<td>User</td>
<td>Can be assigned by project staff to approve a submission</td>
</tr>
<tr>
<td>Rejected</td>
<td>User</td>
<td>Can be assigned by project staff to reject a submission</td>
</tr>
</tbody>
</table>

The Received and Edited states are automatically set by Central any time a submission is uploaded or edited. The other states are assigned by project staff. We suggest some meanings for these states above, but they don’t cause anything to happen automatically. For example, rejected submissions will still be included in your data exports unless you filter them out yourself. So, you are free to use these states however you’d like.

Once submissions have been reviewed, the submission table download and the OData connection both allow submissions to be filtered by review state. This lets you, for example, download only all the approved submissions.

8.4.7 Submission Details

As of version 1.2, each submission has its own detail page which provides basic information about the submission, an activity history of action and discussion on that submission, and tools for updating the submission review state and data itself.
The title at the top is pulled from the instance_name if there is one, otherwise it will be the automatically assigned instanceID. We recommend you define an instance_name based on the data in each submission. This is especially important if you plan on using this page a lot, because it makes navigation much easier to have friendly names at the top of the page and in the web browser title and tab.

Basic detail can be found along the left. If there are expected media attachments for this submission, that status information will be provided.

The main activity feed on the right shows you the discussion and action history of the submission. Any review state changes, comments, and edits will appear here. At the top of the activity feed, you can Review a submission to assign a new review state, Edit the submission directly in your web browser, or type in the box to begin leaving a comment.

You can leave a note when you update the review state, to indicate why the decision is being made, or any other information you’d like saved.

8.4.8 Submission Editing

From the submission detail page you can press the Edit button to edit the submission in your web browser. When an edited submission is resubmitted, a new version of it is created, just like a form version. You will be able to see previous submission versions in a future version of Central.
Any time a user edits a submission, they will see a note when they are returned to the detail page suggesting that they leave a comment describing the edits they have made. This is optional but highly encouraged. In a future version of Central, greater detail will be automatically provided about the data values that were changed.

Finally, when edits are submitted, the submission review state will automatically be set to Edited, and (as of version 1.3) you will see the changes between versions in the Submission Detail activity feed. Please note that Central will show you the differences between versions, but it doesn’t know the exact actions you took to cause those changes. Sometimes the differences shown are not the same as the actions taken, but the resulting data will appear exactly as edited.

8.5 Encrypted Forms in Central

As of version 0.6, ODK Central supports the encryption of submitted data, similar to ODK Aggregate.

For Aggregate users already using encryption

If you already use encryption with ODK Aggregate, you can use the same encrypted forms with ODK Central without any change, and decrypt data using Briefcase with the same steps you already use today. However, you may still wish to read the rest of this document to understand what Central does and the easier to use encryption available to you in Central.

8.5.1 Encryption with Central

First, here is what enabling encryption on a project or form will do:

- Collect will encrypt finalized data on the mobile device.
- Submission data is sent encrypted, even if Central is not configured with HTTPS.
8.5. Encrypted Forms in Central

- Data (filled forms and their attachments) at rest on the Central server is encrypted, and can only be decrypted by using the appropriate secret key information.

However, here are some things that encryption will not do:

- Increase the security of user passwords or Collect QR codes.
- Encrypt the blank form contents or attachments.
- Prevent a hijacker from replacing forms or redirecting submissions to a malicious server.
- Prevent the creation of phony submissions.
- Substitute for HTTPS security on the web administration panel.

And, here are some limitations that will appear when encryption is enabled on a project or form:

- Encrypted submissions will appear only as metadata records over OData. Metadata like the submitter and the submission date will be available, but none of the actual data from the form will appear.
- The same restriction applies to the submission preview table on the Central administration panel.
- As of version 0.6, enabling encryption will not encrypt already-submitted submissions.
- As of version 0.6, it is not possible to disable project managed encryption (explained below) once enabled.
- As of version 0.6, submissions using self-supplied key encryption (Aggregate-style encryption) can be retrieved and decrypted only through Briefcase or direct API access.

8.5.2 Central Encryption Modes

We offer two methods of form encryption in ODK Central:

- **Self-supplied key encryption** is our term for the encryption process already supported by Aggregate and other ODK ecosystem servers. To use self-supplied key encryption, you must generate and securely store your own cryptographic keys which are used to perform the encryption and decryption of the data. The public key (which cannot decrypt the data) is embedded into the XForm itself and this is how it is distributed with the form to Collect and other mobile clients. Decryption is possible only with the private key, and only through Briefcase.

- **Project managed encryption** is new to ODK Central. It uses the same existing ODK encryption standard under the covers, and so it is compatible with existing mobile clients like Collect. However, Central will generate and securely store the cryptographic keys for you. The keys are themselves protected by a passphrase that you provide, similar to the Central backups encryption system. You do not need to use Briefcase: if you can supply the correct passphrase, the Central administration
panel will decrypt the data on the fly and provide you with exported data as usual. Without the passphrase, the data cannot be decrypted, and as Central does not store the passphrase it cannot decrypt the data without your input.

For most cases, we recommend using project managed encryption. It is easier to use, as you do not have to learn how to generate a cryptographic key, you do not have to manually configure each form with the correct key, and you do not need to use Briefcase to decrypt the data. It can also be more secure in many cases, because cryptographic keys are files that must be stored digitally and can be difficult to secure properly. Conversely, a passphrase can be memorized, saved in a password manager, or written down and physically secured.

If you already have familiarity with the encryption in Aggregate, or you cannot at all trust the security of Central itself due to where it has been installed, you may prefer to use self-supplied key encryption.

### 8.5.3 Using Self-Supplied Key Encryption

To use self-supplied key encryption, please see the Encrypted Forms document.

When using a self-supplied key with Central, you may and must apply encryption to each form in a project individually. The encrypted data will not be available over OData or for download as a ZIP file. You will need to use Briefcase to retrieve and decrypt encrypted submissions.

### 8.5.4 Using Project Managed Encryption

For Aggregate users already using encryption

Central will always respect any encryption settings already present in your Forms. This means that if you have already put `<submission base64RsaPublicKey="..."/>` configuration in your Forms, Central managed encryption will ignore those Forms and you will not be able to decrypt Submissions uploaded to those Forms using your managed encryption passphrase. Perhaps this is something you want to have happen, but if not you should remove encryption configuration from your Forms before turning managed encryption on.

Managed encryption can be enabled only at the Project level. To enable managed encryption, first navigate to the Project, then to the Settings tab underneath the Project name. On the right side, you will find the section managing your encryption settings.
8.5. Encrypted Forms in Central

To enable managed encryption for the whole project, first click on the Enable Encryption button. You will be presented with some warnings, which we have also described above in this document:

Once you review those warnings and press Next to proceed, you will be asked for your passphrase, and an optional passphrase hint:
The passphrase you provide is the encryption secret that will be used to secure your data. Anybody who has it will be able to decrypt your submission data. If you lose it, there is no way to recover it, and no way to decrypt your data. Central does not store your passphrase in any way.

The passphrase hint will be displayed whenever the passphrase is needed to decrypt data. It can be a useful way to store information like where in a shared password manager to look for the passphrase. It is optional.

Once you have provided a passphrase and ensured that it is correct, press Next to proceed. At this time, managed encryption will be turned on for the Project. All Forms within the Project will be updated to include encryption information, and mobile devices will have to fetch these new versions in order to submit successfully to Central.

Once encrypted data has been submitted, you will be asked for your encryption passphrase when you try to download your data:
Enter your passphrase and press *Download* to download the data. If the passphrase you provide is incorrect, an error message will be displayed after a moment.

### 8.6 Server Audit Logs in Central

As of version 0.6, ODK Central tracks and logs audit actions for most administrative actions performed on the server. The following actions are logged:

- **Web User Actions**
  - Create
  - Update Details (display name, email, password)
  - Assign Role
  - Revoke Role
  - Log in
  - Retire

- **Project Actions**
  - Create
  - Update Details (name, settings, archival)
  - Delete

- **Form Actions**
  - Create
  - Update Details (state, settings)
  - Create or Update Draft
  - Publish Draft
  - Abandon Draft
  - Update Attachments
  - Download Submissions
  - Delete

- **Submission Actions**
  - Create
  - Update Metadata (review state)
  - Update Submission Data

Our documentation is updated frequently. Get the latest version at [https://docs.getodk.org](https://docs.getodk.org).
– Update Attachments

• App User Actions
  – Create
  – Give Access
  – Remove Access
  – Revoke
  – Delete

• Public Link Actions
  – Create
  – Give Access
  – Remove Access
  – Revoke
  – Delete

• Server Configuration Actions
  – Set

To access the audit logs, navigate to System, then select Server Audit Logs from the navigation menu that appears:
8.6. Server Audit Logs in Central

Some details will sometimes appear in the Details column. The details view will be improved in future versions.

The table always defaults to only showing audit events from the current calendar day. To select a different date range, click on the date in the filter bar. A calendar will appear:

You can click on two separate days to select a range of those days, or you can select a single day by clicking on it twice. Dates and times are always filtered and displayed according to Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
the local time on the computer you are browsing the administration panel from.

You can also filter by the type of audit action instead. To do this, click on the dropdown to
the left of the date, which by default is labeled (All Actions).

You can select a single action to filter by, or you can filter by an entire category (select Form
Actions to see any action that pertains to Forms, for example.

### 8.7 Using the Central REST API

We provide a fully featured REST API for Central. In fact, everything the Central manage-
ment website does is over the REST API, so everything it can do you can, too: creating
users and forms, managing app users, and so on. We also provide compliant standard APIs
for OpenRosa and OData.

For more information on using the Central APIs, please visit our developer documentation.

### 8.8 Using Briefcase with Central

*ODK Briefcase* is a desktop application which provides access to the Central API from
graphical and command-line interfaces. It can make it easier for non-developers to automate
some aspects of data management. In particular, it is useful for migrating data from another
ODK-compatible server to Central.

Briefcase was created to provide features that the older ODK server, Aggregate, did not
have. With Central, we encourage users to either use data exports from the Central interface
or to write scripts to interact directly with the Central API. If you find yourself reaching for
Briefcase a lot with Central, we would like to hear about what features you use it for on the
forum.
Warning: Pulling and pushing media files is memory intensive. If you get a “Success with errors” status from Briefcase, we recommend increasing your Central server memory. If increasing memory is not an option, you can also add swap.

Submission date and submitter are not included in exports from Central.

8.8.1 Migrating from another ODK-compatible server to Central

In most contexts where an existing ODK-compatible server is in use, we recommend completing ongoing data collection projects before switching to Central. However, some long-running studies may want to make the switch before project completion. There may also be some contexts where storing data from past data collection projects in Central makes sense.

ODK Briefcase can be used to perform a migration of both blank forms and form submissions to Central. Please be aware of the following limitations:

- The submission date for all records will be the date that they were migrated to Central, not the date that they were submitted to the original server.

- The submitter for all records will be the user that performs the migration. You may want to create a special user for this purpose and name it something like “Aggregate Migration”.

- Briefcase only keeps track of the latest published form version. For forms that have multiple form versions on the source server, Briefcase will upload the same form definition with different version identifiers. Learn more.

- Migrating multi-version forms where one of the versions is blank is not currently supported automatically. If you need this, we recommend first manually uploading the form definition with a blank version and publishing it. Then you can proceed as normal.

8.8.2 Pushing forms with multiple versions to Central

Note: Briefcase v1.18+ must be used for both the initial pull and the push.

Warning: Pushing multi-version forms where one of the versions is blank is not currently supported. If you need this, we recommend first manually uploading the form definition with a blank version and publishing it. Then you can proceed as normal.

Briefcase only keeps track of the latest published form version. Central only accepts submissions if it knows about the form version that it was created with. To address this, Briefcase
v1.18+ inspects submissions as they are pulled and identifies all of the form versions that are referenced in submissions. On push, Briefcase first pushes and publishes the form definition with versions other than the published one. These are pushed in alphanumerical order. Then it pushes and publishes the form definition with the current version.

Because Briefcase only knows about the latest form version, it can’t push the real older versions. Instead, it pushes the same form definition with different declared versions. This will generally be acceptable because currently, the only thing that can be done with older form versions in Central is to download them. If you’d like to have access to historical edits of the form, you may want to manually upload all of the actual definitions before you push submissions. Be sure to upload the form versions from oldest to newest because the last published definition will be the active one.

If you have pulled submissions with a Briefcase version prior to v1.18, Briefcase will not know what versions submissions correspond to. It will push and publish the active form version and any submissions corresponding to that form version will succeed. However, any submissions corresponding to other form versions will fail. If you let the push complete, you can then retry and Briefcase will create the missing form versions and push the corresponding submissions.

If you end up in a state where the published version is not the correct one, re-upload the form definition that you would like to have published and give it a new version name.
INTRO TO FORMS IN ODK

Forms provide a structured way to collect and provide information. Typically forms are used by data collectors (also called enumerators) who ask questions of and share information with participants. *Web forms* can also be used for self-report.

ODK forms can be used to do things like:

- Conduct a socioeconomic survey of households
- Collect geo-tagged plant species data at different points in time
- Record video and audio of wild animals for later identification
- Guide health workers through a medical triage protocol in multiple languages
- Deliver video-based agricultural training based on a farmer’s responses to questions

Forms are made of fields of *different types* which can each collect or display one piece of information. Visible fields generally represent questions and are displayed differently based on their type.

Questions can be *grouped together* including *on one screen*. They can also be *repeated*. *Form logic, datasets, hidden fields, and external apps* make it possible to create easy-to-use, but powerful forms that represent complex processes.

### 9.1 Excel-based form creation

Most ODK users design their forms in Excel or Google Sheets using *XLSForm*. Examples in this documentation use XLSForm notation to show form features.

### 9.2 Drag-and-drop form creation

For simple forms, *ODK Build* is a drag-and-drop form designer that works both online and offline.
Note: The ODK ecosystems use XML form definitions following the ODK XForms specification, a subset of the W3C XForms specification. The form creation tools described above convert user-friendly form definitions into XML.
XLSForm is a standard for building forms in Excel. XLSForms are simple to get started with and can represent complex forms.

XLSForms can be created and edited by any application that works with .xlsx documents. This means that they are portable and can leverage many useful features commonly available in spreadsheet applications such as formulas, comments and document versioning.

Many users choose to use Google Sheets or Excel for the web so that they can collaborate on or publicly share their forms.

One way to get started is by downloading or making a copy of this form definition template. You can hover over columns to see guidance about how to use them. The All Widgets form definition shows examples of all of the different user-visible question types.

The ODK documentation shows all form design examples as XLSForms and describes how XLSForms are used by ODK tools. There is also a general tutorial for building XLSForms on the XLSForm website.

Once your form has been designed, you can upload the XLSForm directly to Central.

Tip: If your ODK server does not have the latest XLSForm features or you need to temporarily preview a form in a browser, try XLSForm Online.

If you need to design XLSForms offline or your form has sensitive data that you’d rather not upload into XLSForm Online, use the pyxform command line tool.

10.1 The survey sheet

At minimum, an XLSForm has a sheet named survey to describe the types and order of fields in a form. It must have these three columns:

- type: the type of the field represented by each row. Supported types and how they are displayed are described here.
• name: the name of the field represented by each row. This name will be used in your data results. It may not contain spaces and must start with a letter or underscore. Use a short and descriptive name. For example: date_of_birth.

• label: the user-visible question text for the field represented by each row. For example: When was ${first_name} born? This text can reference other fields or have translations.

The survey sheet can have many other columns to represent different question types and form logic. You can see the most commonly-used columns in this template.

10.2 The settings sheet

You should also include a settings sheet to uniquely identify your form definition and its current version. We recommend specifying at least the following columns:

• form_title: The title that will be displayed by tools that list this form.

• form_id: The unique ID that identifies this form to tools that use it. It may not contain spaces and must start with a letter or underscore. Use a short and descriptive name. For example: bench_inventory_2021.

• version: The unique version code that identifies the current state of the form. A common convention is to use a format like yyyymmddrr. For example, 2017021501 is the 1st revision from Feb 15th, 2017.

• instance_name: An expression that will be used to represent a specific filled form created from this form definition. For example, concat(${first_name}, ${age}). Learn more.

The settings sheet is also useful when using multi-language forms or when defining a form with encryption.

10.2.1 Naming filled forms

In an XLSForm’s settings sheet, you can add an instance_name column and specify an expression to use a specific filled form’s contents in its name. This name will be shown in several places to help guide data collection and analysis. You should pick a name that uniquely identifies the filled form and the data it had captured. For example:

• If a single filled form represents data about a real-world thing like a person or park bench, your instance_name expression should include some information to uniquely identify the thing like the person’s name or the park bench’s location and current status.

• If a single filled form represents data about an observation, consider including the date and time of the observation in the instance_name expression.
10.3. The choices sheet

- If your form definition includes a repeat, consider including the repeat count in the instance_name expression.

Filled form names in Collect

Each filled form is identified by its instance_name value in Collect’s Edit Saved Form, Send Finalized Form and View Sent Form lists.

In workflows where forms have to be be filled in multiple different steps, a useful instance_name expression will make it much easier to find which filled form to edit. If forms only have to be edited under certain conditions (e.g. not all household members were available), you can include this status in the instance_name.

In the View Sent Form list, instance_name can be helpful to identify which data collection tasks have been completed. For example, if a data collector needs to interview 25 specific people and the instance_name for each filled form identifies the respondent, they can go to View Sent Form to verify which subset of interviews they have already completed.

A sent form’s instance_name is maintained after it is deleted. This makes it possible to confirm what work has been completed even if submissions are configured to delete after send. However, it does mean sensitive data should be avoided in instance_name.

The instance_name is also used to identify filled forms in Collect’s filled form map.

Filled form names in Central

Each submission in Central has its own detail page which provides basic information about the submission, an activity history of action and discussion on that submission.

The title at the top is pulled from the instance_name and it makes navigation much easier to have friendly names at the top of the page and in the web browser title and tab.

10.3 The choices sheet

If you have multiple choice questions, you will also need a choices sheet to specify choices for those questions. It must have these three columns:

- list_name: The unique ID that identifies a group of choices. It may not contain spaces and must start with a letter or underscore. Use a short and descriptive name. For example: yes_no_maybe.

- name: the name of the field represented by each row. This name will be used in your data results. It may not contain spaces and must start with a letter or underscore. For example you might use 1 or y for Yes and -1 or n for No.
• label: the user-visible text for the choice represented by each row. For example: Yes, No, and Maybe. This text can reference other fields or have translations.

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).
CHAPTER

ELEVEN

QUESTION TYPES

*ODK Collect* supports forms with a wide variety of question types. The exact functionality and display style of each question are specified in your XLSForm definition using the type and appearance columns.

- Text widgets
- Number widgets
- Date and time widgets
- Select widgets
- Rank widget
- Location widgets
- Image widgets
- Audio widgets
- Video widgets
- File upload widget
- Barcode widget
- Range widgets
- Note widget
- URL widget
- Printer widget
- Trigger/acknowledge widget
- Signature widget
- Grouping multiple widgets on the same screen
• Grid of selects on the same screen

• Hidden questions

Tip: You can find an XLSForm with all available question types here.
For simpler forms you can use ODK Build, which provides a visual, drag-and-drop interface.

Helpful terminology
question A prompt to the user, usually requesting a response. Questions are written as a single line in an XLSForm, and usually appear on a single screen in Collect.

widget A rendered question screen in Collect. The type and appearance of a question determine the widget that is displayed.

11.1 Text widgets

All of the text widgets share the text type, and the inputs from them are saved as literal strings.

Warning: If you are using Aggregate and expect answers to be more than 255 characters, you should increase the database field length to over 255 characters.

• Default text widget

• Number text widget

• External app string widget

11.1.1 Default text widget

type text

appearance none

A simple text input.
11.1. Text widgets

The text entry field expands as the user types, and line breaks can be included. The keyboard displayed depends on the Android device and user settings.

XLSForm

Table 11.1: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>appearance</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>name</td>
<td>What is your name?</td>
<td></td>
</tr>
</tbody>
</table>

11.1.2 Number text widget

**type** text

**appearance** numbers

A numerical input that treats the input as a string, rather than a number.

The number input accepts numerals (0123456789), hyphens (−), and decimal points (.). These are the only characters available on the number keypad displayed with this widget.

This is useful for phone numbers, ID numbers, IP addresses, and similar data. It can also be used in place of the Integer widget or Decimal widget if large numbers are needed. (The
integer widget has a limit of nine digits, and the decimal widget has a limit of 15 characters.)

### XLSForm

#### Table 11.2: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>appearance</th>
<th>hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>string_number_widget</td>
<td>String number widget</td>
<td>numbers</td>
<td>text type with numbers appearance</td>
</tr>
</tbody>
</table>

**Note:** This appearance can be combined with the *thousands-sep* appearance.

#### 11.1.3 External app string widget

- **type**: text
- **appearance**: ex.*
11.1. Text widgets

Launches an external app and receives a string input back from the external app. If the specified external app is not available, a manual input is prompted.

The external app widget is displayed when the appearance attribute begins with `ex:`. The rest of the appearance string specifies the application to launch.

See also:

*Launching External Apps*
**XLSForm**

Table 11.3: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>appearance</th>
<th>hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>ex_string</td>
<td>string widget</td>
<td>ex:change.uw.android.BREATHCOUNT Type</td>
<td>with ex:change.uw.android.BREATHCOUNT appearance (can use other external apps)</td>
</tr>
</tbody>
</table>

### 11.2 Number widgets

Number widgets collect and store number inputs — either *integers* or *floating-point decimals*. Number values can also be captured by the *Range widgets*.

- *Integer widget*
- *Decimal widget*
- *Number widget appearance options*
11.2. Number widgets

- Thousands separator
- Number from an external app

11.2.1 Integer widget

type integer

appearance none

A whole number entry input.

Integer widgets will not accept decimal points, and the entry field has a limit of nine digits. If you need numbers larger than nine digits, see the Number text widget.

The integer widget supports:

- Thousands separators
- External apps
Table 11.4: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>integer</td>
<td>age</td>
<td>What is your age in years?</td>
</tr>
</tbody>
</table>

### 11.2.2 Decimal widget

**type**  decimal

**appearance**  none

A numeric input that will accept decimal points.

Decimal number entry is capped at 15 characters (14 digits and a decimal point). If you need numbers larger than 15 digits, see the Number text widget.

The decimal widget supports:

- *Thousands separators*
- *External apps*
11.2. Number widgets

XLSForm

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>decimal</td>
<td>weight</td>
<td>Weight in kilograms.</td>
</tr>
</tbody>
</table>

11.2.3 Number widget appearance options

Thousands separator

type integer, decimal, (text)

appearance thousands-sep, (numbers)

If thousands-sep is added to appearance, integer, decimal, and number text widgets will display their values using locale-specific thousands separators.

Note: For locales that use the point separator (.), a space is used instead.

Fig. 11.1: The comma separator, used in English/U.S. (and others).
Fig. 11.2: The space separator, used in Français/France (and others).
11.2. Number widgets

Number from an external app

type integer, decimal

appearance ex:*  

By specifying an external app in the appearance, your form can launch an external app and receive a number (integer or decimal) from the external app as input. If the specified external app is not available, a manual input is prompted.

See also:

Launching External Apps
Table 11.6: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>appearance</th>
<th>hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>integer</td>
<td>ex_integer_widget</td>
<td>integer type with ex:change.uw.android.BREATHCOUNT appearance (can use other external apps)</td>
<td>integer type with ex:change.uw.android.BREATHCOUNT with integer type</td>
<td></td>
</tr>
</tbody>
</table>

### 11.3 Date and time widgets

- **Default date widget**
  - Date widget with spinner input
  - Month and year only
  - Year only
- **Date widgets with non-Gregorian calendars**

Our documentation is updated frequently. Get the latest version at [https://docs.getodk.org](https://docs.getodk.org).
11.3. Date and time widgets

- Coptic calendar
- Ethiopian calendar
- Islamic calendar
- Bikram Sambat calendar
- Myanmar calendar
- Persian calendar

- Time widget
- Datetime widget

11.3.1 Default date widget

type date

appearance none
Chapter 11. Question Types
11.3. Date and time widgets

XLSForm

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>date_widget</td>
<td>Date widget</td>
<td>date type with no appearance</td>
</tr>
</tbody>
</table>

Date widget with spinner input

**type**  date

**appearance**  no-calendar

The no-calendar appearance displays a spinner-style date selection. This is especially appropriate for selecting dates more than one year in the past or future.
### XLSForm

Table 11.8: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>appearance</th>
<th>hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>date_widget_nocalendar</td>
<td>Date Widget</td>
<td>no-calendar</td>
<td>date type with no-calendar appearance</td>
</tr>
</tbody>
</table>

### Month and year only

type  date
appearance  month-year
Collects only a month and year.
11.3. Date and time widgets

XLSForm

Table 11.9: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>appearance</th>
<th>hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>date_widget_month_year</td>
<td>Date widget</td>
<td>month-year</td>
<td>date type with month-year appearance</td>
</tr>
</tbody>
</table>

**Year only**

type date

appearance year

Collects only a year.
XLSForm

Table 11.10: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>appearance</th>
<th>hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>date_widget_year</td>
<td>Date widget</td>
<td>year</td>
<td>date type with year appearance</td>
</tr>
</tbody>
</table>

11.3.2 Date widgets with non-Gregorian calendars

Collect supports several non-Gregorian calendars.

- Coptic calendar
- Ethiopian calendar
- Islamic calendar
- Bikram Sambat calendar
- Myanmar calendar
11.3. Date and time widgets

- *Persian calendar*

*Note:* The non-Gregorian calendar is used only on input. The dates are converted and stored as standard Gregorian dates.

**Coptic calendar**

- **type** date
- **appearance** coptic

**Ethiopian calendar**

- **type** date
- **appearance** ethiopian
Islamic calendar

type date
appearance islamic
11.3. Date and time widgets

Bikram Sambat calendar

type date
appearance bikram-sambat
Myanmar calendar

type date

appearance myanmar
11.3. Date and time widgets

**Persian calendar**

type  date

appearance  persian
11.3.3 Time widget

type time

appearance none

A time selector. Captures only a specific time-of-day, not a date and time. For date and time, see the Datetime widget.

The time widget does not accept any appearance attributes.

Note: The time widget stores the time along with a time zone. This can cause unexpected behavior around Daylight saving time.

For example, if you record a time before the clock change, and then view the time after the clock change, it will appear to be an hour off. This happens because the recorded time data is understood as a specific moment in time that is being "translated" into your current, local time zone.

A similar problem occurs when moving between geographic time zones.

This makes the time widget unsuitable for abstract time-of-day questions such as What time do you usually wake up? For questions like this, you may want to use a Minimal select widget. You can set the options at whatever level of accuracy you need — for example, 15
11.3. Date and time widgets

or 30 minute increments. Alternatively, you could use the select widget for hours, and an Integer widget for minutes.
Chapter 11. Question Types

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
11.3. Date and time widgets

XLSForm

Table 11.11: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>wakeup</td>
<td>What time do you usually wakeup?</td>
</tr>
</tbody>
</table>

11.3.4 Datetime widget

A date and time selector.

For date only, see Default date widget. For time only, see Time widget.
Chapter 11. Question Types

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
11.3. Date and time widgets
XLSForm

Table 11.12: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>dateTime</td>
<td>previous_meal</td>
<td>When was the last time you ate?</td>
</tr>
</tbody>
</table>

Note: The Datetime widget supports the no-calendar spinner-style appearance.

11.4 Select widgets

Select widgets offer the participant options to pick from. You can offer the participant a single choice, or the ability to choose multiple answers. The order of the choices can be randomized for any of the select types described below. The list of choices available can also be filtered based on answers to previous questions.

The options for a select question are listed on a sheet named choices, in your XLSForm file. The choices sheet has at least three columns:

- **list_name** A set of choices for a single question share a common list_name. The value of list_name is included in the type column on the survey sheet.
- **name** The canonical identifier for a specific choice. This value is what is stored on the completed form. If you refer to a select response using a variable, the name string is returned.
  
  As with the survey sheet, name must not include spaces.
- **label** The user-facing text displayed for the choice.

Select widgets can include images as choices.

- **Single select widget**
  - Minimal select widget
  - Select widget with autoadvance
  - Select widget with autocomplete
  - Select widget with columns-pack appearance
  - Select widget with columns appearance
  - Select widget with columns-n appearance
11.4. Select widgets

- Select widget with no-buttons appearance
- Likert widget
- Multi select widget
- Image map select widget
- Including media files in choices
- Randomizing choice order
- Including "other" as a choice

11.4.1 Single select widget

```
type select_one {list_name}
```

![Single select widget](image)
XLSForm

Table 11.13: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>select_one</td>
<td>select_one_widget</td>
<td>Select one</td>
<td>select_one type with no appearance, 4 text choices</td>
</tr>
<tr>
<td>opt_abcd</td>
<td>select_one_widget</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 11.14: choices

<table>
<thead>
<tr>
<th>list_name</th>
<th>name</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>opt_abcd</td>
<td>a</td>
<td>A</td>
</tr>
<tr>
<td>opt_abcd</td>
<td>b</td>
<td>B</td>
</tr>
<tr>
<td>opt_abcd</td>
<td>c</td>
<td>C</td>
</tr>
<tr>
<td>opt_abcd</td>
<td>d</td>
<td>D</td>
</tr>
</tbody>
</table>

Minimal select widget

type select_one {list_name}

appearance minimal

Adding the minimal appearance shows the choices in a compact way. This is particularly helpful when the list of choices is long and the select question is displayed on the same screen as other questions. It is often combined with the autocomplete appearance.
11.4. Select widgets

![Select one minimal widget](image)

Select one minimal widget

- A
- B
- C
- D

Our documentation is updated frequently. Get the latest version at [https://docs.getodk.org](https://docs.getodk.org).
XLSForm

Table 11.15: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>appearance</th>
<th>hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>select_one</td>
<td>opt_abcd</td>
<td>Select</td>
<td>minimal</td>
<td>select_one type with minimal appearance, 4 text choices</td>
</tr>
</tbody>
</table>

Table 11.16: choices

<table>
<thead>
<tr>
<th>list_name</th>
<th>name</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>opt_abcd</td>
<td>a</td>
<td>A</td>
</tr>
<tr>
<td>opt_abcd</td>
<td>b</td>
<td>B</td>
</tr>
<tr>
<td>opt_abcd</td>
<td>c</td>
<td>C</td>
</tr>
<tr>
<td>opt_abcd</td>
<td>d</td>
<td>D</td>
</tr>
</tbody>
</table>

Select widget with autoadvance

**type** select_one {list_name}

**appearance** quick

When the quick appearance is added, the form advances immediately to the next question once a selection is made.

Video showing auto-advance after the questions are answered.

XLSForm

Table 11.17: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>appearance</th>
<th>hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>select_one</td>
<td>opt_abcd</td>
<td>Select</td>
<td>autoadvance</td>
<td>select_one type with quick appearance, 4 text choices</td>
</tr>
</tbody>
</table>

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
11.4. Select widgets

Table 11.18: choices

<table>
<thead>
<tr>
<th>list_name</th>
<th>name</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>opt_abcd</td>
<td>a</td>
<td>A</td>
</tr>
<tr>
<td>opt_abcd</td>
<td>b</td>
<td>B</td>
</tr>
<tr>
<td>opt_abcd</td>
<td>c</td>
<td>C</td>
</tr>
<tr>
<td>opt_abcd</td>
<td>d</td>
<td>D</td>
</tr>
</tbody>
</table>

Select widget with autocomplete

type select_one {list_name}

appearance autocomplete

The autocomplete appearance allows the enumerator to filter the list of available choices. This is especially helpful for questions with a large number of choices.
Chapter 11. Question Types

**XLSForm**

Table 11.19: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>appearance</th>
<th>hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>select_one</td>
<td>select_one_autocomplete</td>
<td>Select one autocomplete widget</td>
<td>autocomplete appearance, 4 text choices</td>
<td></td>
</tr>
<tr>
<td>opt_abcd</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 11.20: choices

<table>
<thead>
<tr>
<th>list_name</th>
<th>name</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>opt_abcd</td>
<td>a</td>
<td>A</td>
</tr>
<tr>
<td>opt_abcd</td>
<td>b</td>
<td>B</td>
</tr>
<tr>
<td>opt_abcd</td>
<td>c</td>
<td>C</td>
</tr>
<tr>
<td>opt_abcd</td>
<td>d</td>
<td>D</td>
</tr>
</tbody>
</table>

**Select widget with columns-pack appearance**

type select_one {list_name}
11.4. Select widgets

appearance columns-pack

When the columns-pack appearance is added, the app tries to accommodate as many choices in a single line as possible. If the choice labels have different lengths, they will not be in even columns.

XLSForm

Table 11.21: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>appearance</th>
<th>hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>select_one</td>
<td>opt_abcd</td>
<td>select_widget</td>
<td>columns-pack</td>
<td>select_one type with columns-pack appearance, 4 text choices</td>
</tr>
</tbody>
</table>

Table 11.22: choices

<table>
<thead>
<tr>
<th>list_name</th>
<th>name</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>opt_abcd</td>
<td>a</td>
<td>A</td>
</tr>
<tr>
<td>opt_abcd</td>
<td>b</td>
<td>B</td>
</tr>
<tr>
<td>opt_abcd</td>
<td>c</td>
<td>C</td>
</tr>
<tr>
<td>opt_abcd</td>
<td>d</td>
<td>D</td>
</tr>
</tbody>
</table>
Select widget with columns appearance

type select_one {list_name}

appearance columns

When the columns appearance is added, the app puts choices in 2, 3, 4 or 5 columns depending on the screen size.

Select widgets support image choices. The images are referenced in the choices sheet, and the image files need to be included in the media folder.

See *Including media files in choices* to learn more about including images in surveys.

XLSForm

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>appearance</th>
<th>hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>select_one</td>
<td>select_widget</td>
<td>Select one</td>
<td>columns</td>
<td>select_one type with columns appearance, 4 text + image choices</td>
</tr>
<tr>
<td>abcd_icon</td>
<td>select_widget</td>
<td>Select one</td>
<td>columns</td>
<td>select_one type with columns appearance, 4 text + image choices</td>
</tr>
</tbody>
</table>

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
11.4. Select widgets

Table 11.24: choices

<table>
<thead>
<tr>
<th>list_name</th>
<th>name</th>
<th>label</th>
<th>media::image</th>
</tr>
</thead>
<tbody>
<tr>
<td>abcd_icon</td>
<td>a</td>
<td>A</td>
<td>a.jpg</td>
</tr>
<tr>
<td>abcd_icon</td>
<td>b</td>
<td>B</td>
<td>b.jpg</td>
</tr>
<tr>
<td>abcd_icon</td>
<td>c</td>
<td>C</td>
<td>c.jpg</td>
</tr>
<tr>
<td>abcd_icon</td>
<td>d</td>
<td>D</td>
<td>d.jpg</td>
</tr>
</tbody>
</table>

Select widget with columns-n appearance

type select_one {list_name}

appearance columns-n

When the columns-n appearance is added, the app puts choices in n columns.
XLSForm

Table 11.25: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>appearance</th>
<th>hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>select_one</td>
<td>select_widget</td>
<td>Select one widget</td>
<td>columns-2 appearance</td>
<td>select_one type with columns-2 appearance, 4 text + image choices</td>
</tr>
<tr>
<td>abcd_icon</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 11.26: choices

<table>
<thead>
<tr>
<th>list_name</th>
<th>name</th>
<th>label</th>
<th>media::image</th>
</tr>
</thead>
<tbody>
<tr>
<td>abcd_icon</td>
<td>a</td>
<td>A</td>
<td>a.jpg</td>
</tr>
<tr>
<td>abcd_icon</td>
<td>b</td>
<td>B</td>
<td>b.jpg</td>
</tr>
<tr>
<td>abcd_icon</td>
<td>c</td>
<td>C</td>
<td>c.jpg</td>
</tr>
<tr>
<td>abcd_icon</td>
<td>d</td>
<td>D</td>
<td>d.jpg</td>
</tr>
</tbody>
</table>

Select widget with no-buttons appearance

**type** select_one {list_name}

**appearance** no-buttons

When the no-buttons appearance is added, the app displays choices without the selection radio button. If images are specified for choices, only the images are displayed. This is particularly useful for building a grid of images.
11.4. Select widgets

XLSForm

Table 11.27: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>appearance</th>
<th>hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>select_one</td>
<td>select_widget</td>
<td>Select</td>
<td>columns-pack no-buttons</td>
<td>select_one type with columns-pack no-buttons appearance, 4 image choices</td>
</tr>
<tr>
<td>abcd_icon</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 11.28: choices

<table>
<thead>
<tr>
<th>list_name</th>
<th>name</th>
<th>label</th>
<th>media::image</th>
</tr>
</thead>
<tbody>
<tr>
<td>abcd_icon</td>
<td>a</td>
<td>A</td>
<td>a.jpg</td>
</tr>
<tr>
<td>abcd_icon</td>
<td>b</td>
<td>B</td>
<td>b.jpg</td>
</tr>
<tr>
<td>abcd_icon</td>
<td>c</td>
<td>C</td>
<td>c.jpg</td>
</tr>
<tr>
<td>abcd_icon</td>
<td>d</td>
<td>D</td>
<td>d.jpg</td>
</tr>
</tbody>
</table>

Likert widget

New in version 1.25: ODK Collect v1.25.0
type select_one {list_name}

appearance likert

A single-select question can be styled as a Likert scale. Options can include text, images or both. If both are provided, images appear above text.

If adding images, note that the images are referenced in the choices sheet, and the image files need to be included in the media folder. See Including media files in choices to learn more about including images in choices.

XLSForm

Table 11.29: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>appearance</th>
<th>hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>select_one</td>
<td>likert</td>
<td>Likert</td>
<td>likert</td>
<td>select_one type with Likert appearance, 5 image choices (strongly_disagree.jpg, disagree.jpg, neutral.jpg, agree.jpg, strongly_agree.jpg)</td>
</tr>
</tbody>
</table>
11.4. Select widgets

Table 11.30: choices

<table>
<thead>
<tr>
<th>list_name</th>
<th>name</th>
<th>label</th>
<th>media::image</th>
</tr>
</thead>
<tbody>
<tr>
<td>likert_widget</td>
<td>strongly_disagree</td>
<td>Strongly Disagree</td>
<td>strongly_disagree.jpg</td>
</tr>
<tr>
<td>likert_widget</td>
<td>disagree</td>
<td>Disagree</td>
<td>disagree.jpg</td>
</tr>
<tr>
<td>likert_widget</td>
<td>neutral</td>
<td>Neutral</td>
<td>neutral.jpg</td>
</tr>
<tr>
<td>likert_widget</td>
<td>agree</td>
<td>Agree</td>
<td>agree.jpg</td>
</tr>
<tr>
<td>likert_widget</td>
<td>strongly_agree</td>
<td>Strongly Agree</td>
<td>strongly_agree.jpg</td>
</tr>
</tbody>
</table>

11.4.2 Multi select widget

type select_multiple {list_name}

appearance none

Multi select questions support multiple answers.

Note: The multi select widget supports all of the same appearance attributes as the Single select widget excluding the quick appearance:
### XLSForm

Table 11.31: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>select_multiple</td>
<td>select_mult</td>
<td>Multi select</td>
<td>select_multiple type with no appearance, 4 text choices</td>
</tr>
<tr>
<td>opt_abcd</td>
<td>select_multi_widget</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 11.32: choices

<table>
<thead>
<tr>
<th>list_name</th>
<th>name</th>
<th>label</th>
<th>media::image</th>
</tr>
</thead>
<tbody>
<tr>
<td>opt_abcd</td>
<td>a</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>opt_abcd</td>
<td>b</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>opt_abcd</td>
<td>c</td>
<td>C</td>
<td></td>
</tr>
<tr>
<td>opt_abcd</td>
<td>d</td>
<td>D</td>
<td></td>
</tr>
</tbody>
</table>

**Warning:** If you are using Aggregate and expect users to select many options, you may need to increase the database field length to over 255 characters.

### 11.4.3 Image map select widget

New in version 1.13.

**type** select_one `{list_name}`, select_multiple `{list-name}`

**appearance** image-map

The image map widget displays an SVG image with selectable regions.

To make an image with selectable regions:

1. Create or edit an .svg source file. Include id attributes on any elements you want to be selectable.

2. In the choices tab of your XLSForm, put the value of the id attributes in the name column. Add an appropriate human-friendly label to each choice.

3. In the survey tab of your XLSForm, put the .svg file name in the image column.

4. Include the .svg file in your form’s media folder.

See also:

- **Inkscape** An open source vector graphics editor.
- **SVG Documentation** From Mozilla Developer Network.
11.4. Select widgets

Free SVG Files From Wikimedia Commons.
Chapter 11. Question Types

Choose multiple shapes

Selected: blob, rectangle
11.4. Select widgets

SVG

```
<svg width="640" height="480" xmlns="http://www.w3.org/2000/svg" xmlns:svg="http://www.w3.org/2000/svg">
  <title>shapes</title>
  <g>
    <title>Layer 1</title>
    <path id="path" fill="#000080" stroke="#000000" stroke-width="5" d="m125,382c33,56 -193,97 48,55c241,-42 279,-15 48,-62c-38,-47 -183,47 -183,47z"/>
    <rect id="rect" fill="#FF0000" stroke="#000000" stroke-width="5" x="52" y="53" width="176" height="149"/>
    <ellipse id="ellipse" fill="#41A317" stroke="#000000" stroke-width="5" cx="423" cy="143" rx="107" ry="78"/>
  </g>
</svg>
```

XLSForm

Table 11.33: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>appearance</th>
<th>image</th>
</tr>
</thead>
<tbody>
<tr>
<td>select_one shapes</td>
<td>choose-shape</td>
<td>Choose a shape</td>
<td>image-map</td>
<td>shapes.svg</td>
</tr>
<tr>
<td>select_multiple shapes</td>
<td>choose-shapes</td>
<td>Choose multiple shapes</td>
<td>image-map</td>
<td>shapes.svg</td>
</tr>
</tbody>
</table>

Table 11.34: choices

<table>
<thead>
<tr>
<th>list_name</th>
<th>name</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>shapes</td>
<td>path</td>
<td>blob</td>
</tr>
<tr>
<td>shapes</td>
<td>rect</td>
<td>rectangle</td>
</tr>
<tr>
<td>shapes</td>
<td>ellipse</td>
<td>ellipse</td>
</tr>
</tbody>
</table>

11.4.4 Including media files in choices

As with questions themselves, choices can include media (image, video, or audio files):
Table 11.35: choices

<table>
<thead>
<tr>
<th>list_name</th>
<th>name</th>
<th>label</th>
<th>media::image</th>
<th>media::video</th>
<th>media::audio</th>
</tr>
</thead>
<tbody>
<tr>
<td>opt_media</td>
<td>a</td>
<td>A</td>
<td>a.jpg</td>
<td></td>
<td></td>
</tr>
<tr>
<td>opt_media</td>
<td>b</td>
<td>B</td>
<td>b.mp4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>opt_media</td>
<td>c</td>
<td>C</td>
<td></td>
<td></td>
<td>c.mp3</td>
</tr>
</tbody>
</table>

Note: select_one and select_multiple questions using the compact appearances will not display media buttons next to choices. However, if a choice has audio, it will be played when the choice is selected.

11.4.5 Randomizing choice order

Note: Randomizing choice order support was added in Collect v1.18.2 and Central v1.0.0.

To reduce bias, choice order can be randomized for any of the select question types described above. To display the choices in a different order each time the question is displayed, set randomize to true in the parameters column of the XLSForm survey sheet:

XLSForm

Table 11.36: survey

<table>
<thead>
<tr>
<th>type</th>
<th>parameters</th>
<th>name</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>select_one</td>
<td>randomize=true</td>
<td>select_one</td>
<td>Select one with random choice order</td>
</tr>
<tr>
<td>opt_abcd</td>
<td></td>
<td>select_one_random_widget</td>
<td>on each display</td>
</tr>
</tbody>
</table>

Table 11.37: choices

<table>
<thead>
<tr>
<th>list_name</th>
<th>name</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>opt_abcd</td>
<td>a</td>
<td>A</td>
</tr>
<tr>
<td>opt_abcd</td>
<td>b</td>
<td>B</td>
</tr>
<tr>
<td>opt_abcd</td>
<td>c</td>
<td>C</td>
</tr>
<tr>
<td>opt_abcd</td>
<td>d</td>
<td>D</td>
</tr>
</tbody>
</table>

In the example above, each time the question is displayed, the choices will be in a different order. It is often preferable to pick one order that the choices will always be displayed in for a given filled form. This can be accomplished by setting an integer seed for the randomization.
11.4. Select widgets

XLSForm

Table 11.38: survey

<table>
<thead>
<tr>
<th>type</th>
<th>parameters</th>
<th>name</th>
<th>label</th>
<th>calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>calculate</td>
<td></td>
<td>my_seed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>select_one</td>
<td>randomize=true,seed=${my_seed}</td>
<td>select_one_widget</td>
<td>Select one with random choice order set once per filled form</td>
<td></td>
</tr>
</tbody>
</table>

Table 11.39: choices

<table>
<thead>
<tr>
<th>list_name</th>
<th>name</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>opt_abcd</td>
<td>a</td>
<td>A</td>
</tr>
<tr>
<td>opt_abcd</td>
<td>b</td>
<td>B</td>
</tr>
<tr>
<td>opt_abcd</td>
<td>c</td>
<td>C</td>
</tr>
<tr>
<td>opt_abcd</td>
<td>d</td>
<td>D</td>
</tr>
</tbody>
</table>

This seed can also be used to recreate the order choices were displayed in. See the XForms spec for a description of the randomization algorithm used.

Note: In the example above, the integer seed is created from the last 8 numbers of the \( \text{decimal-date-time()} \) which is unlikely to repeat across devices. In the seed expression, \( \text{once()} \) is important because it makes sure the seed is not changed if the same filled form is opened more than once.

11.4.6 Including "other" as a choice

Warning: We do not recommend using or_other because it does not support multiple languages or choice_filter. Instead, add your own "other" question and use form logic to have it appear as needed.

On the survey sheet, in the type column, after the type and the list_name, you can add or_other. This will add "Other" as an additional option to your choice list. The name value of the choice when selected will be other.
11.5 Rank widget

The rank widget allows the user to order options from a list. The value saved in the form and sent to the server is a space-separated ordered list of the options.

Like with Select widgets, the options are listed on a sheet named choices in an XLSForm.

To change the order of the options in the list, tap the Rank items button. In the resulting dialog, long press on an item and once it gets a border around it, drag it up or down to change the order. If no default is provided, the value for the question is blank until the user taps OK in the ranking dialog.

```
type rank {list_name}
```

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
11.5. Rank widget

Rank widget
rank type with no appearance, 4 text choices.
Long press on a choice and drag it to change its position.

Rank items

1. A
2. C
3. D
4. B
### 11.6 Location widgets

Location widgets capture one or more points representing locations on Earth. Each point is represented as four numbers separated by spaces: latitude, longitude, altitude in meters, and accuracy radius in meters.

For example, if a Collect user captured a point while at the coordinates 12°22’17.0”N 1°31’10.9”W, with a reported accuracy radius of 17.4 meters, and at 305 meters above sea level, the geopoint representation would be:

12.371400 -1.519700 305 17.4

Multiple points that form lines or shapes are separated by semicolons.

**Note:** The accuracy radius is an estimate of what Android calls the radius of 68% confidence: there is a 68% chance that the true location falls within this radius. This is an estimate reported by the Android system based on the available sensors (GPS, network, etc). The accuracy radius itself may be more or less reliable depending on the sensor(s) used and current conditions.

To get an accurate location quickly, ensure devices have a clear view of the sky. For even faster points, consider ”warming” the GPS with a start-geopoint question. See improving location performance for more.

**Note:** Since v1.30, when a mock location provider is detected, the accuracy is set to 0. Achieving such perfect accuracy is not possible using GPS so that indicates it comes from a
11.6. Location widgets

mock provider.

In v2021.3 and later, you can opt out of this behavior by setting `allow-mock-accuracy` to `true` in the `parameters` column of your question in your XLSForm `survey` sheet. This is useful for external GPS devices that require Android’s mock provider feature.

- **Geopoint widget**
  - Geopoint with map display
  - Geopoint with user-selected location
- **Geotrace widget**
- **Geoshape**
  - Calculating the area of a geoshape
- **Bearing widget**
- **OpenMapKit widget**

### 11.6.1 Geopoint widget

type geopoint

appearance none

Captures the current geolocation from the device.

For a geopoint with a user-selected location, see *Geopoint with user-selected location*

**Tip:** You can use `selected-at()` to require geopoints meet a particular threshold. For example, if you need points with an accuracy better than 10 meters, use this constraint:

`selected-at(${geopoint_widget}, 3) < 10`.

The 3 in the above constraint references accuracy, the third value in the `geopoint` data type. Use 1 to reference latitude, 2 for longitude, and 4 for altitude.
XLSForm

Table 11.42: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>geopoint</td>
<td>geopoint_widget</td>
<td>Geopoint widget</td>
<td>geopoint type with no appearance</td>
</tr>
</tbody>
</table>

Geopoint with map display

type geopoint

appearance maps

The default *Geopoint widget* does not display a map to the user. When the appearance attribute is maps, the widget displays a map to help the user get oriented and confirm that the selected point is correct and sufficiently accurate.

When the device’s geolocation is available, it is displayed on the map by a blue cross. A blue shaded circle around the cross represents the accuracy radius of the geolocation. The “add marker” button at the top right of the screen can be tapped to add a point at the location indicated by the middle of the blue cross. The selected point is represented by a small circle with a red outline.
11.6. Location widgets

When the map view is opened again with a selected point, the map is centered on that point. To change the selection, first tap the "trash" icon and then select a new point.

For a geopoint with a location that the user can manually select or adjust, see *Geopoint with user-selected location*.

**XLSForm**

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>appearance</th>
<th>hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>geopoint</td>
<td>geo-point_widget_maps</td>
<td>Geopoint widget</td>
<td>maps</td>
<td>geopoint type with maps appearance</td>
</tr>
</tbody>
</table>

**Geopoint with user-selected location**

- **type** geopoint
- **appearance** placement-map

The default *Geopoint widget* does not allow the user to place the point anywhere other than the device’s current geolocation.

A geopoint with the appearance attribute placement-map allows the user to select any point from a map. The user can either long press to place the point anywhere, or, if the device knows its geolocation, tap on the "add point" button at the top right of the screen. The selected point is represented by a small circle with a red outline (see arrow in screenshot).

The save button saves the selected point and returns to the question screen. If the point was selected by long pressing, the accuracy radius and altitude will both be 0. If the device’s geolocation was selected, the accuracy radius will be greater than 0.

When the map view is opened again with an existing point, the map is centered on the selected point. To change the selection, first tap the "trash" icon and then select a new point.
XLSForm

Table 11.44: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>appearance</th>
<th>hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>geo-point</td>
<td>geo-point_widget_placementmap</td>
<td>Geopoint</td>
<td>placement-map</td>
<td>geopoint type with placement-map appearance</td>
</tr>
</tbody>
</table>

11.6.2 Geotrace widget

type geotrace

appearance none

A series of points. Identical to geoshape except that the first and last point may be different and at least 2 points are required.

Points can be entered either by tapping the screen to place each point, or by taking readings of the device’s geolocation over time. On a map, each coordinate is represented by small circles with red outlines. These are connected by red lines.

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
11.6. Location widgets

To collect a geotrace, first select the location-recording mode by tapping the "add point" button in the upper right side of the screen. The selected mode will be displayed in the gray bar at the bottom of the screen. While point collection is ongoing, the "add marker" button changes to a "pause" button. The "back arrow" button can be used to remove the last-entered point either when actively collecting points or when paused. Any point can be manually moved at any time by tapping on it and dragging it. The mode can only be changed if an existing line is first cleared by tapping the "trash" button. Recording must be paused to clear the existing line.

Tip: Points that were entered by tapping or adjusted by dragging will always have an accuracy radius of 0. Points that were read from the device location will never have an accuracy radius of 0.

Once the trace has been saved, the coordinates of its points will be displayed on the question screen. The trace can be opened for manual editing by tapping to add more points, moving existing points or deleting the last-added point. After a trace has been saved once, it cannot be added to in manual or automatic location recording modes.

The three location recording modes are:

**Placement by tapping** The user taps the device to place points.

**Manual location recording** The user chooses when to tap the "record a point" button at the top of the screen to capture the device geolocation at that moment.

**Automatic location recording** The user is prompted to select a recording interval and accuracy requirement. If the accuracy requirement is set to None, points are always collected at the recording interval. If the accuracy requirement is set to any other value, a point will only be captured if it meets the requirement. For example, given a recording interval of 20s and an accuracy requirement of 10m, the app places a point at the device location every 20s if the location is accurate to 10m or better.

**Warning:** If you are using Aggregate and you would like to collect more than 5 points at a time, you should increase the database field length to over 255 characters. Otherwise, additional points will be lost.
Chapter 11. Question Types

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
11.6. Location widgets

XLSForm

Table 11.45: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>geotrace</td>
<td>trace_example</td>
<td>Where have you been?</td>
</tr>
</tbody>
</table>

11.6.3 Geoshape

type geoshape

appearance none

A series of points that form a closed polygon. Identical to geotrace except that the first and last point are always the same and at least 3 points are required.

Points can be entered either by tapping the screen to place each point, or by taking readings of the device’s geolocation over time. On a map, each coordinate is represented by small circles with red outlines. These are connected by red lines.

To collect a geoshape, first select the location-recording mode by tapping the ”add point” button in the upper right side of the screen. The selected mode will be displayed in the gray bar at the bottom of the screen. While point collection is ongoing, the ”add marker” button changes to a ”pause” button. The ”back arrow” button can be used to remove the last-entered point either when actively collecting points or when paused. Any point can be manually moved at any time by tapping on it and dragging it. The mode can only be changed if an existing line is first cleared by tapping the ”trash” button. Recording must be paused to clear the existing line.

Tip: Points that were entered by tapping or adjusted by dragging will always have an accuracy radius of 0. Points that were read from the device location will never have an accuracy radius of 0.

Once the shape has been saved, the coordinates of its points will be displayed on the question screen. The shape can be opened for manual editing by tapping to add more points, moving existing points or deleting the last-added point. After a shape has been saved once, it cannot be added to in manual or automatic location recording modes.

The three location recording modes are:

Placement by tapping The user taps the device to place points.

Manual location recording The user chooses when to tap the ”record a point” button at the top of the screen to capture the device geolocation at that moment.
Automatic location recording  The user is prompted to select a recording interval and accuracy requirement. If the accuracy requirement is set to None, points are always collected at the recording interval. If the accuracy requirement is set to any other value, a point will only be captured if it meets the requirement. For example, given a recording interval of 20s and an accuracy requirement of 10m, the app places a point at the device location every 20s if the location is accurate to 10m or better.

**Warning:** If you are using Aggregate and you would like to collect more than 5 points at a time, you should increase the database field length to over 255 characters. Otherwise, additional points will be lost.
11.6. Location widgets

XLSForm

Table 11.46: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>geoshape</td>
<td>shape_example</td>
<td>Select an area</td>
</tr>
</tbody>
</table>

Calculating the area of a geoshape

type calculate

calculation area(${geoshape})

The area() function calculates the land area, in square meters, of a polygon defined in a Geoshape. The value will be included in your completed survey data, and can also be used in later widgets in the form.
Chapter 11. Question Types

Record a Geoshape

Start GeoShape

Long press to place marks.
11.6. Location widgets

Record a Geoshape

View or Change GeoShape

The area of the recorded geoshape is:

19322.2 m²
**Chapter 11. Question Types**

**XLSForm**

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>geoshape</td>
<td>shape</td>
<td>Record a Geoshape</td>
<td></td>
</tr>
<tr>
<td>calculate</td>
<td>shape_area</td>
<td></td>
<td>area(${shape})</td>
</tr>
<tr>
<td>calculate</td>
<td>rounded_shape_area</td>
<td></td>
<td>round(${shape_area}, 2)</td>
</tr>
<tr>
<td>note</td>
<td>shape_area_note</td>
<td>The area of the recorded geoshape is:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>${rounded_shape_area} m²</td>
<td></td>
</tr>
</tbody>
</table>

**11.6.4 Bearing widget**

**type** decimal

**appearance** bearing

Captures a compass reading, which is stored as a decimal.
Table 11.47: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>appearance</th>
<th>hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>decimal</td>
<td>bearing_widget</td>
<td>Bearing widget</td>
<td>bearing</td>
<td>decimal type with bearing appearance</td>
</tr>
</tbody>
</table>

### 11.6.5 OpenMapKit widget

OpenMapKit allows you to add questions about OpenStreetMap features in a Collect-rendered form.

For more details, see the OpenMapKit documentation.

### 11.7 Image widgets
11.7. **Image widgets**

**Tip:** Image files can be very large. We recommend always including a maximum image size in form design. Also, consider making test submissions to your server with the Internet conditions you expect when gathering data to make sure that you can send files of the size you expect.

- **Default image widget**
  - *Image widget with annotation*
  - *Image widget with required new image*
- **Self portrait (selfie) image widget**
- **External app image widget**
- **Draw widget**
- **Scaling down images**

11.7.1 **Default image widget**

**type** image  

**appearance** none

Captures an image from the device. The user can choose to take a new picture with the device camera, or select an image from the device photo gallery.
Chapter 11. Question Types

XLSForm

Table 11.48: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>image</td>
<td>image_widget</td>
<td>Image widget</td>
<td>image type with no appearance</td>
</tr>
</tbody>
</table>

Image widget with annotation

type image

appearance annotate

Adding the annotate appearance allows the user to draw on the image before submitting it.
11.7. Image widgets

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
Chapter 11. Question Types

312

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
11.7. Image widgets
XLSForm

Table 11.49: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>appearance</th>
<th>hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>image</td>
<td>annotate_image_widget</td>
<td>Annotate widget</td>
<td>annotate</td>
<td>image type with annotate appearance</td>
</tr>
</tbody>
</table>

**Image widget with required new image**

type image

appearance new

An image widget that does not include a *Choose Image* button. This requires the user to take a new picture.

---

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
11.7. Image widgets

XLSForm

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>appearance</th>
<th>hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>image</td>
<td>image_widget</td>
<td>Image widget not choose</td>
<td>new</td>
<td>image type with new appearance (can also be added with annotate appearance and on audio and video types)</td>
</tr>
</tbody>
</table>

11.7.2 Self portrait (selfie) image widget

type image

appearance new-front

Takes a picture using the front-facing ("selfie") camera. The Choose image button is not displayed.

Changed in version 1.15: Prior to v1.15, the appearance attribute for this was selfie. The old appearance attribute will continue to work on existing forms, but new forms should use the new-front appearance.
Chapter 11. Question Types
11.7. Image widgets

XLSForm

Table 11.50: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>hint</th>
<th>appearance</th>
</tr>
</thead>
<tbody>
<tr>
<td>image</td>
<td>self-portrait</td>
<td>Self portrait (selfie) widget</td>
<td>image type with new-front appearance</td>
<td>new-front</td>
</tr>
</tbody>
</table>

11.7.3 External app image widget

New in version 1.30.

Launches an external app and receives an image back from the external app. If the specified external app is not available, it is not possible to use the widget.

The external app image widget is displayed when the appearance attribute begins with ex:. The rest of the appearance string specifies the application to launch.

See also:

Launching External Apps
XLSForm

Table 11.51: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>appearance</th>
<th>hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>image</td>
<td>ex_image_widget</td>
<td>ex:com.example.collectanswersprovider(questionImage=&quot;&quot;) with appearance (can use other external apps)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

11.7.4 Draw widget

type  image

appearance  draw

Provides the user a drawing pad and collects the drawn image.
11.7. Image widgets

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
XLSForm

Table 11.52: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>appearance</th>
<th>hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>image</td>
<td>draw_image_widget</td>
<td>Draw widget</td>
<td>draw</td>
<td>image type with draw appear-</td>
</tr>
</tbody>
</table>

11.7.5 Scaling down images

Images created with any of the image widgets described above can be automatically scaled down on save by using the `max-pixels` parameter. If the long edge of the image is larger than the maximum size specified, the image is resized proportionally so that the long edge matches the provided pixel value. This is useful to reduce the upload size when bandwidth is limited.

Available in Collect since v1.10.0 and in XLSForm since 7/2018.
11.8. Audio widgets

XLSForm

In the parameters column, write `max-pixels=` followed by the desired maximum length of the long edge in pixels.

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>parameters</th>
<th>hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>image</td>
<td>my_scaled_image</td>
<td>Scaled image</td>
<td>max-pixels=1024</td>
<td>image scaled to a max long edge of 1024 pixels</td>
</tr>
</tbody>
</table>

11.8 Audio widgets

- Default audio widget
- Using the built-in audio recorder
  - Customizing audio quality
  - Changing audio quality during form entry
- Recording with an external app
- Getting audio from a custom external app

11.8.1 Default audio widget

type audio

appearance none

Records audio using the device’s microphone or a connected external microphone. By default, an external application is used.

Tip: Instead of the default audio widget, we recommend you use the built-in audio recorder because you can customize audio quality and record while filling out other questions. Built-in recording is available in Collect v1.29 or later.
XLSForm

Table 11.54: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>audio</td>
<td>bird_recording</td>
<td>What does it sound like?</td>
</tr>
</tbody>
</table>

Tip: Audio files can be very large so if you record audio in your form, make sure that you consider your audio quality settings. Also, consider making test submissions to your server with the Internet conditions you expect when gathering data to make sure that you can send files of the size you expect.

Android devices can make many sounds during use and these will be included in recordings. We recommend turning off sounds from button presses, camera shutters and notifications before recording.

11.8.2 Using the built-in audio recorder

type audio
11.8. Audio widgets

appearance none

parameters quality=normal

New in version 1.29: ODK Collect v1.29.0

The built-in audio recorder makes it possible to capture audio without having to install an external app.

It also enables recording while filling out other questions and is designed to continue recording even if the user switches to another app or if the phone screen is locked. To use the built-in audio recorder, specify the desired audio quality for each audio question in your form.

You can also configure Collect to always use the built-in recorder by disabling external app recording.

When built-in audio recording is enabled and recording is initiated, a recording control bar appears at the top of the screen. At the top left of this bar is an icon to represent whether recording is currently ongoing or paused (1). To the right of this icon is the current length of the recording (2).

Warning: Pause is only available on Android 7.0 and above. On lower Android versions, the pause button is hidden.
Chapter 11. Question Types

At the right of the control bar are a pause button (3) and a stop button (4). When the pause button is tapped, recording is temporarily suspended and the button icon changes to a microphone. When the microphone is tapped, recording is resumed. Recording can be paused and resumed as many times as desired. When the stop button is tapped, the recording is ended and can no longer be modified.

Recording status is also displayed below the audio question text. There is a time representing the current length of the recording (5) and a diagram (6) representing the volume of the recording over time. The diagram provides confirmation that the microphone is working and can help a user ensure an even, sufficient volume.

Other questions can be included on the same screen as a built-in recording question. As shown in the screenshot above, this makes it possible to capture quantitative content while recording. To achieve this, put the questions in a field list.

During recording, the user is prevented from leaving the current question screen. However, it is safe to use other applications or to lock the device screen.

Once recording is stopped, the control bar disappears. The recording is made available for playback below the question text.

To replace the audio captured, first delete the current file and then record again.

In some rare cases such as the device running out of space, the recording may complete successfully but not be attached to the form. If this happens, a dialog will be displayed explaining that the file is available but needs to be accessed manually. You can find these files in the recordings folder of the Collect directory. This folder is never cleared so consider emptying it yourself once you have retrieved its files.

Customizing audio quality

ODK Collect v1.29.0, Central v1.1.0.

The quality of audio recordings can be customized using the quality parameter. If a quality is specified, the built-in recorder is always used, regardless of Collect settings. If no quality is specified and external app recording has been disabled, normal is used. The available quality values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Extension</th>
<th>Encoding</th>
<th>Bit rate</th>
<th>Sample rate</th>
<th>File size</th>
</tr>
</thead>
<tbody>
<tr>
<td>normal</td>
<td>.m4a</td>
<td>AAC</td>
<td>64kbps</td>
<td>32kHz</td>
<td>~30MB/hour</td>
</tr>
<tr>
<td>low</td>
<td>.m4a</td>
<td>AAC</td>
<td>24kbps</td>
<td>32kHz</td>
<td>~11MB/hour</td>
</tr>
<tr>
<td>voice-only</td>
<td>.amr</td>
<td>AMR</td>
<td>12.2kbps</td>
<td>8kHz</td>
<td>~5MB/hour</td>
</tr>
</tbody>
</table>

Tip: We’d recommend only using voice-only for one-on-one interviews in a quiet place as otherwise there might be too much detail loss. low will sound compressed but speech is
11.8. Audio widgets

generally intelligible, even if multiple people are talking at once. normal is similar to typical podcast settings and will sound good on most devices.

It’s a good idea to test the different qualities out with the device (and any other equipment) you’ll be using in the field to see which one fits your use case and setup best.

**XLSForm**

In the parameters column, write quality= followed by the desired value.

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>audio</td>
<td>voice_only_audio</td>
<td>Voice audio</td>
<td>quality=voice-only</td>
</tr>
<tr>
<td>audio</td>
<td>normal_audio</td>
<td>Normal audio</td>
<td>quality=normal</td>
</tr>
</tbody>
</table>

**Changing audio quality during form entry**

If it’s a possibility that an individual question could need different qualities depending on context you can use relevance to switch between them:

**XLSForm**

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>select_one</td>
<td>is_quiet</td>
<td>Are you currently in a quiet location with only one person speaking at a time?</td>
<td>relevance=${is_quiet} = 'yes'</td>
</tr>
<tr>
<td>audio</td>
<td>recording_voice_only</td>
<td>Please record quality=voice-only</td>
<td></td>
</tr>
<tr>
<td>audio</td>
<td>recording_normal</td>
<td>Please record quality=normal</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>list_name</th>
<th>name</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes_no</td>
<td>yes</td>
<td>Yes</td>
</tr>
<tr>
<td>yes_no</td>
<td>no</td>
<td>No</td>
</tr>
</tbody>
</table>
11.8.3 Recording with an external app

type audio
appearance none
parameters quality=external

Setting quality to external will cause Collect to use an external app to record audio rather than the built-in recorder. You can also configure Collect to always use an external app for recording (this is the default behavior) and set no quality parameter.

Some Android devices provide a default application for audio recording. Others do not, and the user will need to install an audio recording app.

Any app that responds to android.provider.MediaStore.Audio.Media.RECORD_SOUND_ACTION should be compatible. We recommend Axet Audio Recorder.

11.8.4 Getting audio from a custom external app

New in version 1.30.

Launches an external app and receives an audio file back from the external app. If the specified external app is not available, it is not possible to use the widget.

The external app audio widget is displayed when the appearance attribute begins with ex:. The rest of the appearance string specifies the application to launch.

See also:

Launching External Apps
11.9. Video widgets

**Tip:** Video files can be very large. We recommend configuring video options for every device you intend to use for data collection. Also make submissions to your server with the Internet conditions you expect when gathering data to make sure that you can send files of the size you expect. Note that Central has a 100MB file upload size limit by default.

### XLSForm

**Table 11.58: survey**

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>appearance</th>
<th>hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>audio</td>
<td>ex_audio_widget</td>
<td>External audio widget</td>
<td>ex:com.example.collectanswersprovider(questionAudio=&quot;&quot;) with ex:com.example.collectanswersprovider(questionAudio=&quot;&quot;)</td>
<td>appearance (can use other external apps)</td>
</tr>
</tbody>
</table>

Our documentation is updated frequently. Get the latest version at [https://docs.getodk.org](https://docs.getodk.org).
• Default video widget
• Self portrait (selfie) video widget
• External app video widget

11.9.1 Default video widget

Records video, using the device camera.
11.9. Video widgets

Please record a video of yourself blinking.

Three times is probably sufficient.

- Record Video
- Choose Video
- Play Video
XLSForm

Table 11.59: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>video</td>
<td>blinking</td>
<td>Please record a video of yourself blinking.</td>
<td>Three times is probably sufficient.</td>
</tr>
</tbody>
</table>

11.9.2 Self portrait (selfie) video widget

Records video, using the front-facing ("selfie") camera. The Choose Video button is not displayed.
11.9. Video widgets

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
XLSForm

Table 11.60: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>appearance</th>
<th>hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>video</td>
<td>selfie-video</td>
<td>Self portrait <em>(selfie)</em> video widget</td>
<td>new-front</td>
<td>video type with new-front appearance</td>
</tr>
</tbody>
</table>

11.9.3 External app video widget

New in version 1.30.

Launches an external app and receives a video file back from the external app. If the specified external app is not available, it is not possible to use the widget.

The external app video widget is displayed when the appearance attribute begins with ex:. The rest of the appearance string specifies the application to launch.

See also:

*Launching External Apps*
11.10. File upload widget

XLSForm

Table 11.61: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>appearance</th>
<th>hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>video</td>
<td>ex_video</td>
<td>External video widget</td>
<td>ex:com.example.collectanswersprovider(questionVideo=&quot;&quot;) with ex:com.example.collectanswersprovider(questionVideo=&quot;&quot;) appearance (can use other external apps)</td>
<td></td>
</tr>
</tbody>
</table>

11.10 File upload widget

- Default file upload widget
- External app file widget

11.10.1 Default file upload widget

New in version 1.15: ODK Collect v1.15.0

Uploads any file from the device to the form.

**Warning:** Users can upload any file type, which includes potentially malicious files. You should not include this widget unless you trust the people using the form.

Even then, you should take precautions before downloading or opening files.

- Run an antimalware scan.
- Verify the file is a type you expect (such as a .pdf document), and not a potentially dangerous file (such as .exe or .ini).
Chapter 11. Question Types
11.10. File upload widget

XLSForm

Table 11.62: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>file</td>
<td>some-file</td>
<td>Select a file to upload.</td>
</tr>
</tbody>
</table>

11.10.2 External app file widget

New in version 1.30.

Launches an external app and receives an arbitrary file back from the external app. If the specified external app is not available, it is not possible to use the widget.

The external app file widget is displayed when the appearance attribute begins with ex:.

The rest of the appearance string specifies the application to launch.

See also:

Launching External Apps

**Warning:** This widget accepts files of any type. Learn more about the risk *above*. You should only specify an external application that you trust.
**Chapter 11. Question Types**

**XLSForm**

Table 11.63: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>appearance</th>
<th>hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>file</td>
<td>ex_file</td>
<td>external file widget</td>
<td>ex:com.example.collectanswersprovider(questionFile=”) with ex:com.example.collectanswersprovider(questionFile=”) appearance (can use other external apps)</td>
<td></td>
</tr>
</tbody>
</table>

**11.11 Barcode widget**

Scans, decodes, and captures the content of a barcode, using the device camera.

The following barcode formats are supported:

- UPC-A
- UPC-E
- EAN-8
- EAN-13

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
11.11. Barcode widget

- Code 39
- Code 93
- Code 128
- Codabar
- ITF
- RSS-14
- RSS-Expanded
- QR Code
- Data Matrix
- Aztec (beta)
- PDF 417 (beta)
- MaxiCode

Note: Barcode scanning is built into Collect versions 1.7.0 and greater. Versions of Collect prior to 1.7.0 require the Barcode Scanner app to be installed.

11.11.1 Default barcode widget

The flash can be used as a light source when scanning barcodes in a poorly lit environment.
Chapter 11. Question Types

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
11.11. Barcode widget

XLSForm

Table 11.64: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>barcode</td>
<td>barcode_example</td>
<td>Scan any barcode.</td>
</tr>
</tbody>
</table>

11.11.2 Self portrait (selfie) barcode widget

In some cases a front camera may work better. The flash can’t be used in this case.

XLSForm

Table 11.65: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>appearance</th>
</tr>
</thead>
<tbody>
<tr>
<td>barcode</td>
<td>barcode_example</td>
<td>Scan any barcode.</td>
<td>front</td>
</tr>
</tbody>
</table>
11.12 Range widgets

Range widgets allow the user to select numbers from within a range that is visually represented as a number line. The parameters of the range widget are defined by start, end, and step values defined in the parameters column of your XLSForm. The parameter values can be integers or decimals.

- Default range widget with integers
- Default range widget with decimals
- Vertical range widget
- Range widget with picker
- Range widget with rating

11.12.1 Default range widget with integers

type range
appearance none

If all three parameter values are integers, the input will be stored as an integer.
11.12. Range widgets

Table 11.66: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>appearance</th>
<th>hint</th>
<th>parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>range</td>
<td>range_integer_widget</td>
<td>range integer widget</td>
<td></td>
<td>range integer widget with no appearance</td>
<td>start=1; end=10; step=1</td>
</tr>
</tbody>
</table>

11.12.2 Default range widget with decimals

type range
appearance none

If any of the parameter values are decimals, the input will be stored as a decimal.
### XLSForm

#### Table 11.67: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>appearance</th>
<th>hint</th>
<th>parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>range</td>
<td>range_decimal</td>
<td>Widget decimal</td>
<td>range decimal widget</td>
<td>with no appearance</td>
<td>start=1.5;end=5.5;step=0.5</td>
</tr>
</tbody>
</table>

#### 11.12.3 Vertical range widget

**type** range

**appearance** vertical

To display the range widget’s number line vertically, use the vertical appearance. Both integers and decimals are supported.
11.12. Range widgets

Table 11.68: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>appearance</th>
<th>hint</th>
<th>parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>range</td>
<td>range_integer_widget_vertical</td>
<td>Range vertical integer widget</td>
<td>vertical</td>
<td>range integer widget with vertical appearance</td>
<td>start=1;end=10;step=1</td>
</tr>
</tbody>
</table>

11.12.4 Range widget with picker

type range

appearance picker

When the picker appearance is added, the range widget is displayed with a spinner-style select menu in a dialog. The value between horizontal lines is the selected value. Users can scroll the spinner up and down or can tap on the value above to go up by one and on the value below to go down by one.
Chapter 11. Question Types

Range picker integer widget

range integer widget with picker appearance. This appearance can also be applied to a decimal range.

Select value

No value selected
11.12. Range widgets

XLSForm

Table 11.69: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>appearance</th>
<th>hint</th>
<th>parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>range</td>
<td>range_integer_widget_picker</td>
<td>picker</td>
<td>range integer widget with picker appearance</td>
<td>start=1;end=10;step=1</td>
<td></td>
</tr>
</tbody>
</table>

11.12.5 Range widget with rating

type range

appearance rating

When the rating appearance is added, the range widget is displayed with stars having equal spacing. Number of stars is calculated using the end parameter. When the user taps on an empty star, the stars up to and including that star will be filled. If the stars don’t fit in the device width, they will wrap onto additional lines.
XLSForm

Table 11.70: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>appearance</th>
<th>hint</th>
<th>parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>range</td>
<td>range_integer_widget</td>
<td>Rating rating</td>
<td>rating</td>
<td>range integer widget with</td>
<td>end=9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rating appearance</td>
<td></td>
</tr>
</tbody>
</table>

11.13 Note widget

type  note

appearance  none

A note to the user, accepting no input. This example includes hint text.
### 11.14 URL widget

**XLSForm**

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>note</td>
<td>note_1</td>
<td>This is an example note.</td>
<td>The text displays, but there is no input.</td>
</tr>
</tbody>
</table>

#### 11.14 URL widget

**type** text  

**appearance** url  

Provides a link which the user can open from the survey. Takes no input.

The URL to open is specified with default.
**Table 11.71: survey**

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>appearance</th>
<th>hint</th>
<th>default</th>
</tr>
</thead>
</table>

### 11.15 Printer widget

**type** text

**appearance** printer:org.opendatakit.sensors.ZebraPrinter

Connects to an external label printer, and prints labels that can contain a barcode, a QR code, or text.

See printer-widget for complete details.
11.16. Trigger/acknowledge widget

**XLSForm**

Table 11.72: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>appearance</th>
<th>calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>ex_printer</td>
<td>printer printer widget</td>
<td>printer:org.opendatakit.sensors.ZebraPrinter</td>
<td>concat('123456789','&lt;br&gt;','QR CODE','&lt;br&gt;','Text')</td>
</tr>
</tbody>
</table>

### 11.16 Trigger/acknowledge widget

type trigger, acknowledge

appearance none

The trigger widget, also known as the acknowledge widget, presents a single checkbox. A completed trigger response is stored as the string OK. The example shown here includes the required attribute.

![Trigger widget](image-url)

Prompts for confirmation. Useful to combine with required or relevant. (type=trigger) OK. Please continue.
Chapter 11. Question Types

Trigger widget
Prompts for confirmation. Useful to combine with required or relevant.
(type=trigger)
☐ OK. Please continue.

Sorry, this response is required!
11.17 Signature widget

**XLSForm**

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>hint</th>
<th>required</th>
</tr>
</thead>
<tbody>
<tr>
<td>trigger</td>
<td>my_trigger</td>
<td>Trigger widget</td>
<td>Prompts for confirmation. Useful to combine with required or relevant. (type=trigger)</td>
<td>true()</td>
</tr>
</tbody>
</table>

### 11.17 Signature widget

type image

appearance signature

Collects a signature from the user.
### 11.18 Grouping multiple widgets on the same screen

**XLSForm**

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>appearance</th>
<th>hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>image</td>
<td>signature_widget</td>
<td>Signature widget</td>
<td>signature</td>
<td>image type with signature appearance</td>
</tr>
</tbody>
</table>

**Table 11.74: table**

#### 11.18 Grouping multiple widgets on the same screen

- **type** `begin_group`
- **appearance** `field-list`

The field-list appearance attribute, applied to a group of widgets, displays them all on a single screen.

**Warning:** Relevance, constraint and calculation evaluation within the same screen is supported in Collect v1.22 and later.

**Warning:** Displaying *Repeating questions* on the same screen (inside a field-list group) is not supported.

**See also:**

Groups of questions and Repeating questions.

#### 11.19 Grid of selects on the same screen

If you have multiple select questions with the same choices, it can be helpful to group them on one screen.
To do this, put your select questions in a field-list group and use the following appearance attributes:

**label** Only the option labels are displayed, without checkboxes. This is used for the top row with the 'Yes' and 'No' options in the example above.

**list-nolabel** Only checkboxes or radio buttons are displayed, without their labels. This is used for the question rows in the example above.

**list** The labels are displayed along with checkboxes for multi-select questions and radio buttons for single-select questions. You could use this instead of having a label row to keep the option labels closer to the checkboxes or radio buttons.

**XLSForm**

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>appearance</th>
</tr>
</thead>
<tbody>
<tr>
<td>begin_group</td>
<td>underlying_conditions</td>
<td>Underlying conditions</td>
<td>field-list</td>
</tr>
<tr>
<td>select_one yes_no</td>
<td>condition_labels</td>
<td>Conditions</td>
<td>label</td>
</tr>
<tr>
<td>select_one yes_no</td>
<td>Comcond_preg</td>
<td>Pregnancy</td>
<td>list-nolabel</td>
</tr>
<tr>
<td>select_one yes_no</td>
<td>Comcond_partum</td>
<td>Post-partum (&lt; 6 weeks)</td>
<td>list-nolabel</td>
</tr>
<tr>
<td>end_group</td>
<td>underlying_conditions</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## 11.20 Hidden questions

Not all question types render as visible widgets in Collect. Hidden fields collect and store values which are accessible as *variables* and available in *Central* and other data analysis tools.

<table>
<thead>
<tr>
<th>list_name</th>
<th>name</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes_no</td>
<td>yes</td>
<td>Yes</td>
</tr>
<tr>
<td>yes_no</td>
<td>no</td>
<td>No</td>
</tr>
</tbody>
</table>

### 11.20.1 Metadata

Metadata questions capture information about the device or a survey collection event and are not visible to the user.

A more complete record of user behavior within a form can be generated by enabling *form audit logging*.

These items are dependent on the survey collection event:

- **start** — The datetime the survey was started in ISO 8601 format (e.g., 2019-09-27T09:45:10.854-07:00).
- **end** — The last datetime the survey was saved in ISO 8601 format.
- **today** — The date the survey was started in ISO 8601 format (e.g., 2019-09-27).
- **start-geopoint** — The geolocation when the survey was started. *Read more.*
This item is defined at installation time and cannot be changed:

- `deviceid`

These items are defined in Collect, and can be edited in Settings:

- `username`
- `phonenumber`

### XLSForm

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>start</code></td>
<td><code>start</code></td>
</tr>
<tr>
<td><code>end</code></td>
<td><code>end</code></td>
</tr>
<tr>
<td><code>today</code></td>
<td><code>today</code></td>
</tr>
<tr>
<td><code>deviceid</code></td>
<td><code>deviceid</code></td>
</tr>
<tr>
<td><code>username</code></td>
<td><code>username</code></td>
</tr>
<tr>
<td><code>phonenumber</code></td>
<td><code>phonenumber</code></td>
</tr>
<tr>
<td><code>start-geopoint</code></td>
<td><code>start-geopoint</code></td>
</tr>
</tbody>
</table>

#### 11.20.2 Geolocation at survey start

See also:

*Audit log geolocation tracking*

**Note:** Geolocation at survey start was added in Collect v1.23 and Central v1.0.0.

The start-geopoint question type is used to capture a single geolocation in *geopoint format* when the survey is first started. Questions of type start-geopoint may be given any allowable name. Although it is possible to have more than one start-geopoint question in a form, all will have the same value.

Any time a survey with a `start-geopoint` question is opened in Collect, the enumerator will see a warning that the form tracks device location. If the device battery is low, or if location tracking needs to be turned off for any reason, you can tap → *Track location* or turn off location providers in Android.

The first time that a survey with a `start-geopoint` question is opened, Collect will attempt to read the device’s geolocation. The geolocation reading with the highest accuracy received in a 20-second window will be recorded. A location icon will be displayed in the Android status bar while the geolocation is being requested by Collect.
11.20. Hidden questions

Geolocation is read using data from GPS, WiFi and possibly other signals so this feature should work in most environments.

If geolocation information is unavailable, the question will be left blank. Reasons for a blank value may include the enumerator turning off location providers, Collect not having location permissions, Google Play Services not being installed, the GPS not having satellite lock and more. No troubleshooting information is provided in the form submission.

Including a *start-geopoint* question may make it faster to get high-accuracy geolocation readings for other location question types by ”warming” the GPS.

11.20.3 Calculate

type calculate

Calculate questions let you evaluate complex *expressions*, storing the values for later use. For more details, see *Calculations*.

11.20.4 Background audio recording

type background-audio

New in version 1.30: ODK Collect v1.30.0, Central v1.2.0

See also:

*Logging enumerator behavior*, *audio questions*

**XLSForm**

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>background-audio</td>
<td>my_recording</td>
</tr>
</tbody>
</table>

When a form includes a question of type background-audio, audio is recorded while the form is open and attached to the form submission as a single audio file. These recordings can be used for quality assurance, training, transcription, and more. Use background recording instead of an *audio question* when you want to make sure to record everything that happens during form filling.

By default, audio files will be saved in the *amr* format with a bitrate of 12.2kbps and a sample rate of 8kHz, resulting in a file size of about 5MB per hour. These settings correspond to the voice-only quality *for audio questions* and minimize file size while maintaining reasonable
quality for a conversation between two people. You can override that default quality by specifying a value in the parameters column as described for audio questions.

XLSForm

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>background-audio</td>
<td>my_recording</td>
<td>quality=low</td>
</tr>
</tbody>
</table>

Planning for background audio recording

Before adding background audio recording to your form, make sure that you have a plan for following local laws around audio recording. We generally recommend including a note at the beginning of the form to remind data collectors and participants that they are being recorded and to describe the purpose of the recording. Depending on the context, you may be required to ask for the consent of every person in speaking range of the microphone.

Additionally, you should make a plan for using the resulting audio files. Do you have someone who can listen to and make sense of many audio files? Could you get access to speech-to-text capabilities? Also consider whether your data collectors will be able to send audio files. Will they have access to a fast enough Internet connection? Will their Internet plan or number of credits allow them to send all the audio files you expect?

It can be helpful to combine background audio recording with audit logging to have more context while listening to the recording. Recording starts at the form start and form resume events and stops at the form exit event. You can use the difference between the form start time and the start time of an event to identify how far into the background recording a certain event happened.

Background audio recording user interface

While recording is ongoing, an audio status bar is shown at the top of the screen. This bar helps remind data collectors that they are being recorded and provides visual feedback about audio volume.

If a data collector exits a form and then re-opens it for editing, audio recording is resumed. Audio recording continues as long as the form is open, even if another application is in the foreground or the screen is locked. Note that Collect settings can’t be accessed directly from a form that has background recording. The audio file sent to the server will include audio from every time that the form was opened for editing.

Tip: Android devices can make many sounds during use and these will be included in...
11.20. Hidden questions

recordings. We recommend turning off sounds from button presses, camera shutters and
notifications before recording.

There is an override available to data collectors when recording might compromise safety
or when consent to record can’t be obtained but it’s still important to capture some data.
When the audio recording option is unchecked, audio recording ends immediately and any
previously-recorded audio is deleted. Recording can’t be resumed during the current form-
filling session. Toggling the audio recording option back on indicates that the next form filling
session should be recorded. If audit logging is enabled, a background audio disabled event
will be logged if a data collector toggles off recording and a background audio enabled
event will be logged if a data collector toggles it back on.

Background audio recording troubleshooting

In some rare cases such as the device running out of space, the recording may complete
successfully but not be attached to the form. If this happens, the recording may be available
in the recordings folder of the Collect directory. This folder is never cleared so consider
emptying it yourself once you have retrieved its files.
ODK Collect supports a wide range of dynamic form behavior. This document covers how to specify this behavior in your XLSForm definition.

See also:

XLSForm

Warning: Relevance, constraint and calculation evaluation within the same screen is supported in Collect v1.22 and later. In earlier versions of Collect, questions tied by logic must be displayed on different screens.

- Form logic building blocks
  - Variables
  - Expressions
  - Calculations
  - Values from the last saved record
  - Form logic gotchas
- Requiring responses
- Setting default responses
  - Static defaults
  - Dynamic defaults
  - Dynamic defaults in repeats
  - Dynamic defaults from form data
- Triggering calculations on value change
Chapter 12. Form Logic

12.1 Form logic building blocks

12.1.1 Variables

Variables reference the value of previously answered questions. To use a variable in XLSForm, put the question’s name in curly brackets preceded by a dollar sign:

\${question-name}

Variables can be used in label, hint, and repeat_count columns, as well as any column that accepts an expression.
12.1. Form logic building blocks

What is your name?

Adam

Hello, Adam.
XLSForm

Table 12.1: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>your_name</td>
<td>What is your name?</td>
</tr>
<tr>
<td>note</td>
<td>hello_name</td>
<td>Hello, ${your_name}.</td>
</tr>
</tbody>
</table>

You can also refer to the current question or to the current question’s parent group or repeat:

<table>
<thead>
<tr>
<th>Explanation</th>
<th>Example</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>current question’s value</td>
<td>.</td>
<td>&gt;= 18 Used in constraints.</td>
</tr>
<tr>
<td>current question’s parent group</td>
<td>position(...)</td>
<td>Used with position() to get a parent repeat instance’s index.</td>
</tr>
</tbody>
</table>

Advanced: XPath paths

The ${} notation in XLSForm is a convenient shortcut to refer to a specific field. When an XLSForm is converted, ${} references are expanded to XPath paths which describe where the field is located in the form.

Some tools like ODK Build do not support ${} notation so XPath notation must be used. Even in XLSForm, it can be advantageous to use XPath notation, especially in the context of repeats or datasets. The ${} and XPath notations can be mixed freely.

One way to think about XPath is that it sees a form or dataset like a series of folders and files on your computer. Questions are like files while groups and repeats are like folders because they can contain other elements. Path elements are separated by /.

The absolute path to q1 is /data/outer/inner/q1.

The data in the example above is the name of the form root. This root is named data by default but can be modified by adding a name column in the XLSForm settings sheet and specifying a value below it. This is rarely needed. The / at the start of the path indicates that the path is absolute.

XPath paths can also be relative. For example, let’s say there’s a relevance expression for q1 in the example above and that this expression refers to a question with name age in the outer group. We could refer to it using an absolute expression: /data/outer/age. We could also write a relative expression: ../../age.

The ../.. part of the relative expression says to go up two levels from the current position of /data/outer/inner/q1. The first .. goes up one level to /data/outer/inner and then the second .. goes up another level to /data/outer/. We want to access a question in the outer group so we add that question’s name to get ../../age.
ODK tools support a subset of XPath described in the ODK XForms specification.

**XPath predicates for filtering**

In *repeats* and *datasets*, an XPath path can refer to multiple nodes. This is called a nodeset. XPath predicates are True/False (boolean) expressions in square brackets that filter the nodeset they come after. When you define a *choice filter* for a select, that expression is used as an XPath predicate to filter the choice items.

You can also write your own expressions with predicates. For example, consider a form with a repeat with name *people* and a question inside with name *age* (see *XPath paths for repeats* for the form definition). The expression `/data/people[age < 18]` evaluates to a nodeset that includes all instances of the *people* instance for which the value of the *age* question is less than 18. *age* in the predicate is a relative expression evaluated in the context of each node in the nodeset. In this case, the relative expression *age* is evaluated in the context of `/data/people`, giving the path expression `/data/people/age`. This means that `/data/people/age` is compared to 18 for every *people* repeat instance.

You can add more path steps after a predicate. For example, `/data/people[age < 18]/pet_count` evaluates to a nodeset that includes all the pet counts for instances of the *people* repeat that have *age* values under 18. Nodesets can be passed in to functions like `sum()` or other functions that take nodeset arguments.

Sometimes forms may use *groups* to organize question sections within repeats. Those groups must be accounted for in predicates. If the *age* question were nested in a group called *inner*, the predicate expression would need to be *inner/age < 18*. Additionally, if the *pet_count* question were nested in a group called *details*, the full expression would be `/data/people[inner/age < 18]/details/pet_count`.

XPath predicates are also the way to reference specific values in a *dataset*. Learn more in the section on *referencing values in datasets*.

**XPath paths for repeats**

When a form definition includes a *repeat*, corresponding filled forms will have 0 or more instances of that repeat. Using the file and folder analogy described above, each repeat instance is like a folder and all of these folders have the name of the *repeat*. Repeat instances are differentiated by their index (first, second, ...).

When writing expressions within a repeat, it can be helpful to use the position of the repeat instance an enumerator is currently filling out. This can be done by using the *position()* function. One context in which this is useful is if you want to first collect a roster of people or things and then ask additional questions about each of those. As shown in the example in the *position()*, you can use a first repeat for the roster and then a second repeat that references items in the first repeat based on their position.
Another use of the `position` function is to access a preceding repeat instance. See an example of this in the section on `dynamic defaults in repeats`.

XPath paths can be useful to reference some or all repeat instances from outside the repeat. XPath notation is particularly helpful for filtering repeat instances, for example to provide a summary from data collected in repeats:

### XLSForm

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>begin repeat</td>
<td>people</td>
<td>Person</td>
<td></td>
</tr>
<tr>
<td>int</td>
<td>age</td>
<td>Age</td>
<td></td>
</tr>
<tr>
<td>int</td>
<td>pet_count</td>
<td>How many pets does this person have?</td>
<td></td>
</tr>
<tr>
<td>end repeat</td>
<td>people</td>
<td></td>
<td></td>
</tr>
<tr>
<td>int</td>
<td>total_pets</td>
<td></td>
<td>sum(${people}[age &lt; 18]/pet_count)</td>
</tr>
<tr>
<td>note</td>
<td>total_note</td>
<td>Total pets owned by children:</td>
<td>${total_pets}</td>
</tr>
</tbody>
</table>

In the path expression ${people}[age < 18]/pet_count, ${people} uses ${} notation to refer to all of the instances of the repeat. You could also expand this to the XPath path of `/data/people`. See the section on `XPath predicate` for more details. In this example, the `total_pets` value is displayed to the user. It could be used in many different contexts such as to define the `relevance` of a group if there’s a section of questions that only need to be filled out if there are more than one child-owned pets in the community.

#### 12.1.2 Expressions

An `expression` is evaluated dynamically as a form is filled out. It can include XPath functions, operators, `values from previous responses`, and (in some cases) `the value of the current response`.

**Example expressions**

${bill_amount} * 0.18 Multiplies the previous value `bill_amount` by 18%, to calculate a suitable tip.
12.1. Form logic building blocks

concat(${first_name}, ' ', ${last_name})  Concatenates two previous responses with
a space between them into a single string.

${age} >= 18  Evaluates to True or False, depending on the value of age.

round(${bill_amount} * ${tip_percent} * 0.01, 2)  Calculates a tip amount based
on two previously entered values, and then rounds the result to two decimal places.

Expressions are used in:

- Calculations
- Validating and restricting responses
- Conditionally showing questions

12.1.3 Calculations

To evaluate complex expressions, use a calculate row. Put the expression to be evaluated in
the calculation column. Then, you can refer to the calculated value using the calculate row’s
name.

Expressions cannot be used in label and hint columns, so if you want to display calculated
values to the user, you must first use a calculate row and then a variable.
Table 12.3: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>decimal</td>
<td>bill_amount</td>
<td>Bill amount:</td>
<td>round((${bill_amount} * 0.18),2)</td>
</tr>
<tr>
<td>calculate</td>
<td>tip_18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>calculate</td>
<td>tip_18_total</td>
<td></td>
<td>${bill_amount} + ${tip_18}</td>
</tr>
<tr>
<td>note</td>
<td>tip_18_note</td>
<td></td>
<td>Bill: ${bill_amount}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Tip (18%): ${tip_18}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Total: ${tip_18_total}</td>
</tr>
</tbody>
</table>
12.1.4 Values from the last saved record

**Warning:** Support for last-saved was added in Collect v1.21.0 and Central v1.3.0. Using older versions or encrypted forms will have unpredictable results.

The last-saved feature does not work with encrypted forms.

You can refer to values from the last saved record of this form definition:

\[ ${\text{last}-\text{saved}\#\text{question-name}} \]

This can be very useful when an enumerator has to enter the same value for multiple consecutive records. An example of this would be entering in the same district for a series of households.

**XLSForm that shows using a last-saved value as a dynamic default**

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>default</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>street</td>
<td>Street</td>
<td>[${\text{last}-\text{saved}#street}]</td>
</tr>
</tbody>
</table>

The value is pulled from the last saved record. This is often the most recently created record but it could also be a previously-existing record that was edited and saved. For the first record ever saved for a form definition, the last saved value for any field will be blank.

Questions of any type can have their defaults set based on the last saved record. References to the last saved record can be used as part of any expression wherever expressions are allowed.

12.1.5 Form logic gotchas

**When expressions are evaluated**

Every expression is constantly re-evaluated as an enumerator progresses through a form. This is an important mental model to have and can explain sometimes unexpected behavior. More specifically, expressions are re-evaluated when:

- a form is opened
- the value of any question in the form changes
- a repeat group is added or deleted
• a form is saved or finalized

This is particularly important to remember when using functions that are not connected to fields in the form such as \texttt{random()} or \texttt{now()}. The value they represent may change as the conditions listed above take place.

To control when an expression is evaluated, use \textit{dynamic defaults} or \textit{trigger calculations on value change}. Dynamic defaults are evaluated exactly once on form load or repeat creation.

\section*{Empty values in math}

Unanswered \textit{number questions} are nil. That is, they have no value. When a \textit{variable} referencing an empty value is used in a math \textit{operator} or \textit{function}, it is treated as Not a Number (NaN). The empty value \textbf{will not} be converted to zero. The result of a calculation including NaN will also be NaN, which may not be the behavior you want or expect.

To convert empty values to zero, use either the \texttt{coalesce()} function or the \texttt{if()} function.

\begin{verbatim}
coalesce($\{potentially_empty_value\}, 0)
\end{verbatim}

\begin{verbatim}
if($\{potentially_empty_value\}==\"\", 0, $\{potentially_empty_value\})
\end{verbatim}

\section*{12.2 Requiring responses}

By default, users are able to skip questions in a form. To make a question required, put yes in the required column.

Required questions are marked with a small asterisk to the left of the question label. You can optionally include a \texttt{required\_message} which will be displayed to the user who tries to advance the form without answering the question.
12.2. Requiring responses

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
12.3 Setting default responses

To provide a default response to a question, put a value in the default column. Defaults are set when a record is first created from a form definition. Defaults can either be fixed values (static defaults) or the result of some expression (dynamic defaults).

12.3.1 Static defaults

The text in the default column for a question is taken literally as the default value. Quotes should not be used to wrap values, unless you actually want those quote marks to appear in the default response value.

In the example below, the "Phone call" option with underlying value phone_call will be selected when the question is first displayed. The enumerator can either keep that selection or change it.

**XLSForm to select "Phone call" as the default contact method**

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>default</th>
</tr>
</thead>
<tbody>
<tr>
<td>select_one</td>
<td>contact_method</td>
<td>How should we contact you?</td>
<td>phone_call</td>
</tr>
</tbody>
</table>

---

Our documentation is updated frequently. Get the latest version at [https://docs.getodk.org](https://docs.getodk.org).
12.3. Setting default responses

12.3.2 Dynamic defaults

**Warning:** Support for *dynamic defaults* was added in Collect v1.24.0 and Central v1.0.0. Using older versions will have unpredictable results.

If you put an expression in the default column for a question, that expression will be evaluated once when a record is first created from a form definition. This allows you to use values from outside the form like the current date or the *server username*. Dynamic defaults as described in this section are evaluated once on record creation. See below for using *dynamic defaults in repeats* or setting the default value of one field to the value of another field in the form.

**XLSForm to set the current date as default**

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>default</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>fever_onset</td>
<td>When did the fever start?</td>
<td>now()</td>
</tr>
</tbody>
</table>

In the example below, if a username is set either in the *server configuration* or the *metadata settings*, that username will be used as the default for the question asked to the enumerator.

**XLSForm to confirm metadata like username**

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>default</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
<td>username</td>
<td></td>
<td></td>
</tr>
<tr>
<td>text</td>
<td>confirmed_username</td>
<td>What is your username?</td>
<td>${username}</td>
</tr>
</tbody>
</table>

**Tip:** If enumerators will need to enter the same value for multiple consecutive records, dynamic defaults can be combined with *last saved*. For example, if enumerators are collecting data about trees and trees of the same kind grow together, you can use the last saved tree species as the default for new records.

12.3.3 Dynamic defaults in repeats

Dynamic defaults in repeats are evaluated when a new repeat instance is added.
One powerful technique is to use a value from a previous repeat instance as a default for the current repeat instance. For example, you could use the tree species specified for the last visited tree as the default species for the next tree.

**Note:** If you are collecting data about multiple entities such as trees, you can choose to use repeats or to use one form record per entity. See the repeats section for more information on making that decision. If you use one form record per entity, you can use last saved to get the same behavior as described in this section.

### XLSForm to set a default value based on the last repeat instance

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>default</th>
</tr>
</thead>
<tbody>
<tr>
<td>begin_repeat</td>
<td>tree</td>
<td>Tree</td>
<td></td>
</tr>
<tr>
<td>text</td>
<td>species</td>
<td>Species</td>
<td>${tree}[position() = position(current()/..) - 1]/species</td>
</tr>
</tbody>
</table>

In the default expression above, \${tree} is a reference to the repeat. The expression [position() = position(current()/. . ) - 1] in brackets says to filter the list of possible tree repeat instances to only include the one with a position that is one less than the current repeat’s position. Finally, /species specifies that the species question from the repeat should be used. This is a mix of XLSForm’s ${} shortcut syntax for specifying question names and raw XPath syntax.

### 12.3.4 Dynamic defaults from form data

**Warning:** Support for dynamic defaults from form data was added in Collect v1.24.0 and Central v1.1.0. Using older versions will have unpredictable results.

It can be helpful to use a value filled out by the enumerator as a default for another question that the enumerator will later fill in. Dynamic defaults as described above can’t be used for this because they are evaluated on form or repeat creation, before any data is filled in.

You also can’t use the calculation column on its own for this because the expression in the calculation would be evaluated on form save and replace any changes the enumerator has made. Instead, use a combination of calculation and trigger. The question reference in the trigger column will ensure that your calculation is only evaluated when that reference changes.
12.3. Setting default responses

**XLSForm that uses current age as the default for diagnosis age**

Table 12.11: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>calculation</th>
<th>trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>name</td>
<td>Child’s name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>integer</td>
<td>current_age</td>
<td>Child’s age</td>
<td></td>
<td></td>
</tr>
<tr>
<td>select_one</td>
<td>gender</td>
<td>Gender</td>
<td></td>
<td></td>
</tr>
<tr>
<td>integer</td>
<td>diagnosis_age</td>
<td>Age at malaria diagnosis</td>
<td>${current_age}</td>
<td>${current_age}</td>
</tr>
</tbody>
</table>

In the example above, `${current_age}` in the trigger column means that when the value of the `current_age` question is changed by the enumerator, the calculation for the `diagnosis_age` question will be evaluated. In this case, this means the new value for `current_age` will replace the current value for `diagnosis_age`. If the enumerator then changes the value for `diagnosis_age`, this value will be retained unless the value for `current_age` is changed again.

Another option for the scenario above is to clear out the value for `diagnosis_age` when `current_age` changes. Making `diagnosis_age` a required question will force the enumerator to update `diagnosis_age` if `current_age` is corrected.

**XLSForm that clears diagnosis age if current age is updated**

Table 12.12: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>required</th>
<th>calculation</th>
<th>trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>name</td>
<td>Child’s name</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>integer</td>
<td>current_age</td>
<td>Child’s age</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>select_one</td>
<td>gender</td>
<td>Gender</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>integer</td>
<td>diagnosis_age</td>
<td>Age at malaria diagnosis</td>
<td>true()</td>
<td>&quot;</td>
<td>${current_age}</td>
</tr>
</tbody>
</table>

In the example above, `diagnosis_age` is cleared any time the value of the `current_age` question is changed.

This kind of default is particularly useful if a form is being filled in about entities that there is already some knowledge about. For example, if you have a list of people to interview and you know their phone numbers, you may want to use the known phone number as a default and allow the enumerator to update it as needed.
**Chapter 12. Form Logic**

**XLSForm that looks up default values based on a selection**

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>calculation</th>
<th>trigger</th>
<th>choice_filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>select_one</td>
<td>participant</td>
<td>Participant</td>
<td>instance('participants')/root/item[name=${participant}]/phone_number</td>
<td></td>
<td>true()</td>
</tr>
<tr>
<td>text</td>
<td>phone_number</td>
<td>Phone number</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 12.14: choices

<table>
<thead>
<tr>
<th>list_name</th>
<th>name</th>
<th>label</th>
<th>phone_number</th>
</tr>
</thead>
<tbody>
<tr>
<td>participants</td>
<td>kwame_onwuachi</td>
<td>Kwame Onwuachi</td>
<td>+1-850-555-0168</td>
</tr>
<tr>
<td>participants</td>
<td>sophia_roe</td>
<td>Sophia Roe</td>
<td>+36 55 562 079</td>
</tr>
</tbody>
</table>

In the example above, when a participant is selected, his or her phone number is populated as a default and can be updated as needed. If the selected participant changes, the phone number is replaced.

**Note:** The `true()` in the choice_filter column for the `select_one` in the example above is necessary to be able to look up participants’ phone numbers. This is currently needed to overcome a **pyxform** bug.

**12.4 Triggering calculations on value change**

**Warning:** Support for triggering calculations on value change was added in Collect v1.24.0 and Central v1.1.0. Using older versions will have unpredictable results.

Calculations are recomputed any time one of the values in its expression changes. For example, if your form includes the calculation `$\{q1\} + \{q2\}$, it will be recomputed any time either of the values for `$\{q1\}$ or `$\{q2\}$ changes.

Calculations can also be triggered when a value not involved in the calculation changes. This uses the same **trigger** column as **defaults from form data**. It is particularly useful for triggering calculations that involve values not in the form like random numbers or time.
12.4.1 Lightweight timestamping

Knowing how long an enumerator spent answering a question can help with quality control and training. ODK Collect provides an *audit log* that contains rich information about how an enumerator navigated a form. This log is captured as a separate file and can be complex to analyze. A simpler alternative is to capture timestamps when specific questions’ values change. This is similar to the *start* and *end* timestamp *metadata types*.

Capturing last update timestamps

Table 12.15: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>calculation</th>
<th>trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>name</td>
<td>Name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dateTime</td>
<td>name_last_update_time</td>
<td>Name</td>
<td>now()</td>
<td>${name}</td>
</tr>
<tr>
<td>string</td>
<td>life_story</td>
<td>Life story</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dateTime</td>
<td>life_story_last_update_time</td>
<td>Life story</td>
<td>now()</td>
<td>${life_story}</td>
</tr>
</tbody>
</table>

In the example above, the `name_last_update_time` field will be populated with the current time whenever the enumerator changes the value of the `name` question.

You can also capture the first time a question’s value is changed using an *if* in the *calculation*:

Capturing first update timestamps

Table 12.16: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>calculation</th>
<th>trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>name</td>
<td>Name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dateTime</td>
<td>name_first_update_time</td>
<td>Name</td>
<td><code>if(${name_first_update_time}=&quot;&quot;, now(), ${name_first_update_time})</code></td>
<td>${name}</td>
</tr>
<tr>
<td>string</td>
<td>life_story</td>
<td>Life story</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dateTime</td>
<td>life_story_first_update_time</td>
<td>Life story</td>
<td><code>if(${life_story_first_update_time}=&quot;&quot;, now(), ${life_story_first_update_time})</code></td>
<td>${life_story}</td>
</tr>
</tbody>
</table>

12.5 Validating and restricting responses

To validate or restrict response values, use the constraint column. The constraint expression will be evaluated when the user advances to the next screen. If the expression evaluates
to True, the form advances as usual. If False, the form does not advance and the constraint_message is displayed.

The entered value of the response is represented in the expression with a single dot ( . ).

Constraint expressions often use comparison operators and regular expressions. For example:

- \( . \geq 18 \) True if response is greater than or equal to 18.
- \( . > 20 \text{ and } . < 200 \) True if the response is between 20 and 200.
- \( \text{regex}(.,'p\{L\}+') \) True if the response only contains letters, without spaces, separators, or numbers.
- \( \text{not(contains(., 'prohibited'))} \) True if the substring prohibited does not appear in the response.

**Note:** Constraints are not evaluated if the response is left blank. To restrict empty responses, make the question required.

**See also:**

*Using regular expressions*
### 12.6. Conditionally showing questions

**XLSForm**

Table 12.17: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>constraint</th>
<th>constraint_message</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>middle_initial</td>
<td>What is your middle initial?</td>
<td>regex(, 'p{L}')</td>
<td>Just the first letter.</td>
</tr>
</tbody>
</table>

#### 12.5.1 Read-only questions

To completely restrict user-entry, use the read_only column with a value of yes. This is usually combined with a *default response*, which is often *calculated* based on *previous responses*.

**XLSForm**

Table 12.18: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>read_only</th>
<th>default</th>
<th>calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>decimal</td>
<td>salary_income</td>
<td>Income from salary</td>
<td></td>
<td></td>
<td>${salary_income}</td>
</tr>
<tr>
<td>decimal</td>
<td>self_income</td>
<td>Income from self-employment</td>
<td></td>
<td></td>
<td>${self_income}</td>
</tr>
<tr>
<td>decimal</td>
<td>other_income</td>
<td>Other income</td>
<td></td>
<td></td>
<td>${other_income}</td>
</tr>
<tr>
<td></td>
<td>income_sum</td>
<td></td>
<td></td>
<td></td>
<td>${income_sum}</td>
</tr>
<tr>
<td></td>
<td>total_income</td>
<td>Total income</td>
<td>yes</td>
<td></td>
<td>${income_sum}</td>
</tr>
</tbody>
</table>

#### 12.6 Conditionally showing questions

The relevant column can be used to show or hide questions and groups of questions based on previous responses.

If the expression in the relevant column evaluates to `True`, the question or group is shown. If `False`, the question is skipped.

Often, comparison *operators* are used in relevance expressions. For example:

\[ \${age} \leq 5 \quad \text{True if age is five or less.} \]
${has_children} = 'yes'  True if the answer to has_children was yes.

Relevance expressions can also use functions. For example:

selected(${allergies}, 'peanut')  True if peanut was selected in the Multi select widget named allergies.

contains(${haystack}, 'needle')  True if the exact string needle is contained anywhere inside the response to haystack.

count-selected(${toppings}) > 5  True if more than five options were selected in the Multi select widget named toppings.

12.6.1 Simple example

XLSForm

Table 12.19: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>relevant</th>
</tr>
</thead>
<tbody>
<tr>
<td>select_one</td>
<td>watch_sports</td>
<td>Do you watch sports?</td>
<td>${watch_sports} = 'yes'</td>
</tr>
<tr>
<td>yes_no</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>text</td>
<td>favorite_team</td>
<td>What is your favorite team?</td>
<td></td>
</tr>
</tbody>
</table>

Table 12.20: choices

<table>
<thead>
<tr>
<th>list_name</th>
<th>name</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes_no</td>
<td>yes</td>
<td>Yes</td>
</tr>
<tr>
<td>yes_no</td>
<td>no</td>
<td>No</td>
</tr>
</tbody>
</table>

12.6.2 Complex example

Video showing a series of select questions. The questions displayed change depending on what choices are selected in the first questions.
## 12.6. Conditionally showing questions

### XLSForm

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>hint</th>
<th>relevant</th>
<th>constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>select_multiple</td>
<td>what_issues</td>
<td>Have you experienced any of the following?</td>
<td>Select all that apply.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>select_multiple</td>
<td>what_cancer</td>
<td>What type of cancer have you experienced?</td>
<td>Select all that apply.</td>
<td></td>
<td>selected(${what_issues}, 'cancer')</td>
</tr>
<tr>
<td>select_multiple</td>
<td>what_diabetes</td>
<td>What type of diabetes do you have?</td>
<td>Select all that apply.</td>
<td></td>
<td>selected(${what_issues}, 'diabetes')</td>
</tr>
<tr>
<td>begin_group</td>
<td>blood_pressure</td>
<td>Blood pressure reading</td>
<td>selected(${what_issues}, 'hypertension')</td>
<td></td>
<td></td>
</tr>
<tr>
<td>integer</td>
<td>systolic_bp</td>
<td>Systolic</td>
<td></td>
<td>. &gt; 40 and . &lt; 400</td>
<td></td>
</tr>
<tr>
<td>integer</td>
<td>diastolic_bp</td>
<td>Diastolic</td>
<td></td>
<td>. &gt;= 20 and . &lt;= 200</td>
<td></td>
</tr>
<tr>
<td>end_group</td>
<td>text</td>
<td>other_health</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>note</td>
<td>after_health</td>
<td>This note is after all health questions.</td>
<td></td>
<td></td>
<td>selected(${what_issues}, 'other')</td>
</tr>
</tbody>
</table>
Table 12.22: choices

<table>
<thead>
<tr>
<th>list_name</th>
<th>name</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>medical_issues</td>
<td>cancer</td>
<td>Cancer</td>
</tr>
<tr>
<td>medical_issues</td>
<td>diabetes</td>
<td>Diabetes</td>
</tr>
<tr>
<td>medical_issues</td>
<td>hypertension</td>
<td>Hypertension</td>
</tr>
<tr>
<td>medical_issues</td>
<td>other</td>
<td>Other</td>
</tr>
<tr>
<td>cancer_types</td>
<td>lung</td>
<td>Lung cancer</td>
</tr>
<tr>
<td>cancer_types</td>
<td>skin</td>
<td>Skin cancer</td>
</tr>
<tr>
<td>cancer_types</td>
<td>prostate</td>
<td>Prostate cancer</td>
</tr>
<tr>
<td>cancer_types</td>
<td>breast</td>
<td>Breast cancer</td>
</tr>
<tr>
<td>cancer_types</td>
<td>other</td>
<td>Other</td>
</tr>
<tr>
<td>diabetes_types</td>
<td>type_1</td>
<td>Type 1 (Insulin dependent)</td>
</tr>
<tr>
<td>diabetes_types</td>
<td>type_2</td>
<td>Type 2 (Insulin resistant)</td>
</tr>
</tbody>
</table>

Warning: Calculations are evaluated regardless of their relevance.

For example, if you have a calculate widget that adds together two previous responses, you cannot use relevant to skip in the case of missing values. (Missing values will cause an error.)

Instead, use the if() function to check for the existence of a value, and put your calculation inside the then argument.

For example, when adding together fields a and b:

```plaintext
if(${a} != '' and ${b} != '', ${a} + ${b}, '')
```

In context:

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>integer</td>
<td>a</td>
<td>a =</td>
<td></td>
</tr>
<tr>
<td>integer</td>
<td>b</td>
<td>b =</td>
<td></td>
</tr>
<tr>
<td>calculate</td>
<td>a_plus_b</td>
<td></td>
<td>if(${a} != &quot; and ${b} != &quot;, ${a} + ${b},&quot;)</td>
</tr>
<tr>
<td>note</td>
<td>display_sum</td>
<td>a + b =</td>
<td>${a_plus_b}</td>
</tr>
</tbody>
</table>

12.7 Groups of questions

To group questions, use the begin_group...end_group syntax.
12.7. Groups of questions

XLSForm — Question group

Table 12.23: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>begin_group</td>
<td>my_group</td>
<td>My text widgets</td>
</tr>
<tr>
<td>text</td>
<td>question_1</td>
<td>Text widget 1</td>
</tr>
<tr>
<td>text</td>
<td>question_2</td>
<td>These questions will both be grouped together</td>
</tr>
<tr>
<td>end_group</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

You can use the field-list appearance on a group to display multiple questions on the same screen.

If given a label, groups will be visible in the form path to help orient the user (e.g. My text widgets > Text widget 1). Labeled groups will also be visible as clickable items in the jump menu:

Warning: If you use ODK Build v0.3.4 or earlier, your groups will not be visible in the jump menu. The items inside the groups will display as if they weren’t grouped at all.

Groups without labels can be helpful for organizing questions in a way that’s invisible to the user. This technique can be helpful for internal organization of the form. These groups can
also be a convenient way to conditionally show certain questions.

## 12.8 Repeating questions

You can ask the same question or questions multiple times by wrapping them in `begin_repeat...end_repeat`. By default, enumerators are asked before each repetition whether they would like to add another repeat. It is also possible to determine the number of repetitions ahead of time which can make the user interaction more intuitive. You can also add repeats as long as a condition is met.

### XLSForm — Repeating one or more questions

Table 12.24: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>begin_repeat</td>
<td>my_repeat</td>
<td>repeat group label</td>
</tr>
<tr>
<td>note</td>
<td>repeated_note</td>
<td>All of these questions will be repeated.</td>
</tr>
<tr>
<td>text</td>
<td>name</td>
<td>What is your name?</td>
</tr>
<tr>
<td>text</td>
<td>quest</td>
<td>What is your quest?</td>
</tr>
<tr>
<td>text</td>
<td>fave_color</td>
<td>What is your favorite color?</td>
</tr>
<tr>
<td>end_repeat</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Warning:** Displaying repeating questions on the same screen (inside a field-list group) is not supported.

**See also:**

*Repeat Recipes and Tips* describes strategies to address common repetition scenarios.

**Tip:** Using repetition in a form is very powerful but can also make training and data analysis more time-consuming. Repeats exported from Central or Briefcase will be in their own files and will need to be joined with their parent records for analysis.

Before adding repeats to your form, consider other options:

- if the number of repetitions is small and known ahead of time, consider “unrolling” the repeat by copying the same questions several times.

- if the number of repetitions is large and includes many questions, consider building a separate form that enumerators fill out multiple times and link the forms with some parent key (e.g., a household ID).
If repeats are needed, consider adding some summary calculations at the end so that analysis will not require joining the repeats with their parent records. For example, if you are gathering household information and would like to compute the total number of households visited across all enumerators, add a calculation after the repeats that counts the repetitions in each submission.

### 12.8.1 Controlling the number of repetitions

**User-controlled repeats**

By default, the enumerator controls how many times the questions are repeated. Before each repetition, the user is asked if they want to add another.

**Note:** The label in the begin_repeat row is shown in quotes in the Add ...? message. A meaningful label will help enumerators and participants navigate the form as intended. We generally recommend using a singular noun or noun phrase such as "observation" or "household member". This interaction may be confusing to users the first time they see it. If enumerators know the number of repetitions ahead of time, consider using a dynamically defined repeat count.

**Tip:** The jump menu also provides shortcuts to add or remove repeat instances.

**Fixed repeat count**

Use the repeat_count column to define the number of times that questions will repeat.
Fig. 12.1: The user is given the option to add each repetition.
12.8. Repeating questions

XLSForm

Table 12.25: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>repeat_count</th>
</tr>
</thead>
<tbody>
<tr>
<td>begin_repeat</td>
<td>my_repeat</td>
<td>Repeat group label</td>
<td>3</td>
</tr>
<tr>
<td>note</td>
<td>repeated_note</td>
<td>These questions will be repeated exactly three times.</td>
<td></td>
</tr>
<tr>
<td>text</td>
<td>name</td>
<td>What is your name?</td>
<td></td>
</tr>
<tr>
<td>text</td>
<td>quest</td>
<td>What is your quest?</td>
<td></td>
</tr>
<tr>
<td>text</td>
<td>fave_color</td>
<td>What is your favorite color?</td>
<td></td>
</tr>
<tr>
<td>end_repeat</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Dynamically defined repeat count

The repeat_count column can reference previous responses and calculations.

XLSForm

Table 12.26: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>repeat_count</th>
</tr>
</thead>
<tbody>
<tr>
<td>integer</td>
<td>number_of_children</td>
<td>How many children do you have?</td>
<td></td>
</tr>
<tr>
<td>begin_repeat</td>
<td>child_questions</td>
<td>Questions about child</td>
<td>${number_of_children}</td>
</tr>
<tr>
<td>text</td>
<td>child_name</td>
<td>Child’s name</td>
<td></td>
</tr>
<tr>
<td>integer</td>
<td>child_age</td>
<td>Child’s age</td>
<td></td>
</tr>
<tr>
<td>end_repeat</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Repeating as long as a condition is met

If the enumerator won’t know how many repetitions are needed ahead of time, you can still avoid the ”Add ...?” dialog by using the answer to a question to decide whether another repeat instance should be added. In the example below, repeated questions about plants will be asked as long as the user answers “yes” to the last question.
Table 12.27: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>calculation</th>
<th>repeat_count</th>
</tr>
</thead>
<tbody>
<tr>
<td>calculate</td>
<td>count</td>
<td></td>
<td>count(${plant})</td>
<td></td>
</tr>
<tr>
<td>begin_repeat</td>
<td>plant</td>
<td>Plant</td>
<td>if(${count} = 0 or</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>${plant}[position()=${count}]/more_plants = 'yes', ${count} + 1, ${count})</td>
<td></td>
</tr>
<tr>
<td>text</td>
<td>species</td>
<td>Species</td>
<td></td>
<td></td>
</tr>
<tr>
<td>integer</td>
<td>estimated</td>
<td>Estimated size</td>
<td></td>
<td></td>
</tr>
<tr>
<td>select_one</td>
<td>more_plants</td>
<td>there more</td>
<td></td>
<td></td>
</tr>
<tr>
<td>yes_no</td>
<td>plants</td>
<td>plants in this area?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>end_repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 12.28: choices

<table>
<thead>
<tr>
<th>list_name</th>
<th>name</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes_no</td>
<td>yes</td>
<td>Yes</td>
</tr>
<tr>
<td>yes_no</td>
<td>no</td>
<td>No</td>
</tr>
</tbody>
</table>

This works by maintaining a \texttt{count()} of the existing repetitions and either making \texttt{repeat\_count} one more than that if the continuing condition is met or keeping the \texttt{repeat\_count} the same if the ending condition is met.

In the \texttt{repeat\_count} expression, \texttt{${count} = 0} ensures that there is always at least one repeat instance created. The continuing condition is \texttt{${plant}[position()=${count}]/more\_plants = 'yes'} which means "the answer to \texttt{more\_plants} was \texttt{yes} the last time it was asked." The expression \texttt{position()=${count}} uses the \texttt{position()} function to select the last plant that was added. Adding \texttt{/more\_plants} to the end of that selects the \texttt{more\_plants} question.

### Repeating zero or more times

Sometimes it only makes sense to collect information represented by the questions in a repeat under certain conditions. If the number of total repetitions is known ahead of time, use \textit{Dynamically defined repeat count} and allow a count of 0. If the count is not known ahead of time, \textit{Conditionally showing questions} can be used to represent 0 or more repetitions. In the example below, questions about trees will only be asked if the user indicates that there are trees to survey.


### 12.9 Filtering options in select questions

To limit the options in a select question based on the answer to a previous question, specify an expression in the choice_filter column of the survey sheet. This expression will refer to one or more column in the choices sheet that the dataset should be filtered by.

For example, you might ask your enumerators to select a state first, and then only display cities within that state. This is referred to as a "cascading select" and can be extended to any depth. The example below has two levels: job category and job title.

The choice_filter expression for the second select in the example is `category=${job_category}`. category is the name of a column in the choices sheet and `${job_category}` refers to the first select question in the form. The filter expression says to only include rows whose category column value exactly matches the value selected by the enumerator as `${job_category}`.

Any expression that evaluates to True or False can be used as a choice_filter. For example, you could add a location column to the choices sheet and also ask the user to enter a location they want to consider jobs in. If the new location question on the survey sheet is named `${job_location}`, the choice filter would be `category=${job_category} and location=${job_location}`. Another example of a complex choice filter is one that uses text comparison functions to match labels that start with a certain value. Consider, for example, `starts-with(label, ${search_value})` where search_value is the name of a text question defined on the survey sheet.
12.10 Generating select ones from repeats

If you use a repeat, you can generate a follow-up `select_one` question using values from the repeat. For example, if you collect information about several household members in a repeat, you can then show a select one with all household members’ names. To do this, add a question of type `select_one` followed by the name of the question in the repeat that you want to use for the select options.
12.10. Generating select ones from repeats

Table 12.33: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>required</th>
<th>choice_filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>begin_repeat</td>
<td>person</td>
<td>Person</td>
<td></td>
<td></td>
</tr>
<tr>
<td>text</td>
<td>person_name</td>
<td>Person’s name?</td>
<td>true()</td>
<td></td>
</tr>
<tr>
<td>integer</td>
<td>person_age</td>
<td>${person_name}’s age?</td>
<td>true()</td>
<td></td>
</tr>
<tr>
<td>end_repeat</td>
<td>person</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>select_one</td>
<td>${person_name}</td>
<td>tallest</td>
<td></td>
<td></td>
</tr>
<tr>
<td>select_one</td>
<td>${person_name}</td>
<td>tallest_child under 18.</td>
<td></td>
<td>${person_age} &lt; 18</td>
</tr>
</tbody>
</table>

As shown in the example above, you can combine this with other select features such as filtering. Note that in the example above, the question used as select option text is required. If a question used to generate a select_one is not required and it is left blank for some repeat instances, those repeat instances will not be included in the select.
Expressions in calculations, constraints, and relevants can contain operators and functions.

- **Operators**
  - Math operators
  - Comparison operators
  - Boolean operators
  - Path operators

- **Functions**
  - Control flow
  - Accessing response values
  - Strings
  - Math
  - Date and time
  - Geography
  - Utility
13.1 Operators

13.1.1 Math operators

<table>
<thead>
<tr>
<th>Operation</th>
<th>Explanation</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>addition</td>
<td>${salary_income} + ${self_employed_income}</td>
</tr>
<tr>
<td>-</td>
<td>subtraction</td>
<td>${income} - ${expenses}</td>
</tr>
<tr>
<td>*</td>
<td>multiplication</td>
<td>${bill} * 1.18</td>
</tr>
<tr>
<td>div</td>
<td>division</td>
<td>${percent_int} div 100</td>
</tr>
<tr>
<td>mod</td>
<td>modulo (division remainder)</td>
<td>(${even_number} mod 2) = 0</td>
</tr>
</tbody>
</table>

**Warning:** Math operators only work with numbers.
- The addition operator cannot be used to concatenate strings. Use the `concat()` function instead.
- Empty values (that is, variables referencing unanswered questions) are actually empty strings, and will not be automatically converted to zero (0).

13.1.2 Comparison operators

Comparison operators are used to compare values. The result of a comparison is always True or False.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Explanation</th>
<th>Example</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>equal to</td>
<td>${enrolled} = 'yes'</td>
<td>Can compare numbers or strings.</td>
</tr>
<tr>
<td>!=</td>
<td>not equal to</td>
<td>${enrolled} != 'yes'</td>
<td>Can compare numbers or strings.</td>
</tr>
<tr>
<td>&gt;</td>
<td>greater than</td>
<td>${age} &gt; 17</td>
<td></td>
</tr>
<tr>
<td>&gt;=</td>
<td>greater than or equal to</td>
<td>${age} &gt;= 18</td>
<td></td>
</tr>
<tr>
<td>&lt;</td>
<td>less than</td>
<td>${age} &lt; 65</td>
<td></td>
</tr>
<tr>
<td>&lt;=</td>
<td>less than or equal to</td>
<td>${age} &lt;= 64</td>
<td></td>
</tr>
</tbody>
</table>

**Warning:**
- The relational operators (>, >=, <, <=) only work with numbers.
- Empty response values are not automatically converted to zero (0).
13.2. Functions

13.1.3 Boolean operators

Boolean operators combine two True or False values into a single True or False value.

<table>
<thead>
<tr>
<th>Boolean operator</th>
<th>Explanation</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>and</td>
<td>True if the expressions before and after are True</td>
<td>( {\text{age}} &gt; -1 ) and ( {\text{age}} &lt; 120 )</td>
</tr>
<tr>
<td>or</td>
<td>True if either of the expressions before or after are True</td>
<td>( {\text{age}} &lt; 19 ) or ( {\text{age}} &gt; 64 )</td>
</tr>
</tbody>
</table>

13.1.4 Path operators

<table>
<thead>
<tr>
<th>Path operator</th>
<th>Explanation</th>
<th>Example</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>current question’s value</td>
<td>( {\text{age}} \geq 18 )</td>
<td>Used in constraints.</td>
</tr>
<tr>
<td>..</td>
<td>current question’s parent group</td>
<td>position(..)</td>
<td>Used with position() to get a parent repeat instance’s index.</td>
</tr>
</tbody>
</table>

Note: Formally, these are not operators but rather XPath references to the current node (.) and the parent node (..). XPath paths can be used to reference nodes of a form.

13.2 Functions

- Control flow
- Accessing response values
  - Select questions
  - Repeat groups
- Strings
  - Searching and matching strings
  - Combining strings
  - Converting to and from strings
- Math
  - Number handling
Chapter 13. Form Operators and Functions

- Calculation
  - Date and time
    - Converting dates and time
    - Formatting dates and times for display
  - Geography
  - Utility

See also:
Functions in the ODK XForm Specification

13.2.1 Control flow

```
if(expression, then, else)
    Returns then if expression evaluates to True. Otherwise, returns else.
```

```
position(xpath)
    Returns an integer equal to the 1-indexed position of the current node within the node defined by xpath.
    Most often this is used in the form position(..) to identify the current iteration index within a repeat group.
```
### 13.2. Functions

#### XLSForm

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>repeat_count</th>
<th>calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>note</td>
<td>person_list</td>
<td>Please list the names of your household.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>begin_repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>text</td>
<td>name</td>
<td>Name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>end_repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>begin_repeat</td>
<td>person_details</td>
<td>Details</td>
<td></td>
<td>count(${person})</td>
</tr>
<tr>
<td>calculate</td>
<td>current_name</td>
<td></td>
<td></td>
<td>indexed-repeate(${name}, ${person}, position(..))</td>
</tr>
<tr>
<td>date</td>
<td>member_bday</td>
<td>Birthday of ${current_name}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>end_repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**once**(expression)

Returns the value expression if the question’s value is empty. Otherwise, returns the current value of the question.

This can be used to ensure that a random number is only generated once, or to store the first value entered for a question in a way that is retrievable even if the response is changed later.

**Warning:** This function is often misunderstood. Read *when expressions are evaluated* to learn more.

### 13.2.2 Accessing response values

**Note:** The response from most question types can be accessed using *variables*. Functions are needed for accessing responses to *multi select questions* and questions inside *repeat groups*.
Select questions

\texttt{selected}($\text{space\_delimited\_array, string}$)

Returns \texttt{True} if string is a member of \texttt{space\_delimited\_array}, otherwise returns \texttt{False}.

Commonly used to determine if a specific choice was selected in a \textit{select question}. (This is possible because a \textit{reference} to a select question returns a space-delimited array of choice names.)

Video showing a series of select questions. The questions displayed change depending on what choices are selected in the first questions.
### XLSForm

**Table 13.1: survey**

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>hint</th>
<th>relevant</th>
<th>constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>select_multiple medical_issues</td>
<td>what_issues</td>
<td>Have you experienced any of the following?</td>
<td>Select all that apply.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>select_multiple cancer_types</td>
<td>what_cancer</td>
<td>What type of cancer have you experienced?</td>
<td>Select all that apply.</td>
<td></td>
<td>selected(${what_issues}, 'cancer')</td>
</tr>
<tr>
<td>select_multiple diabetes_types</td>
<td>what_diabetes</td>
<td>What type of diabetes do you have?</td>
<td>Select all that apply.</td>
<td></td>
<td>selected(${what_issues}, 'diabetes')</td>
</tr>
<tr>
<td>begin_group</td>
<td>blood_pressure_reading</td>
<td>Blood pressure reading</td>
<td>selected(${what_issues}, 'hypertension')</td>
<td></td>
<td></td>
</tr>
<tr>
<td>integer</td>
<td>systolic_bp</td>
<td>Systolic</td>
<td></td>
<td></td>
<td>. &gt; 40 and . &lt; 400</td>
</tr>
<tr>
<td>integer</td>
<td>diastolic_bp</td>
<td>Diastolic</td>
<td></td>
<td></td>
<td>. &gt;= 20 and . &lt;= 200</td>
</tr>
<tr>
<td>end_group</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>text</td>
<td>other_health</td>
<td>other issues.</td>
<td>selected(${what_issues}, 'other')</td>
<td></td>
<td></td>
</tr>
<tr>
<td>note</td>
<td>after_health</td>
<td>This note is after not health questions.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 13.2: choices

<table>
<thead>
<tr>
<th>list_name</th>
<th>name</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>medical_issues</td>
<td>cancer</td>
<td>Cancer</td>
</tr>
<tr>
<td>medical_issues</td>
<td>diabetes</td>
<td>Diabetes</td>
</tr>
<tr>
<td>medical_issues</td>
<td>hypertension</td>
<td>Hypertension</td>
</tr>
<tr>
<td>medical_issues</td>
<td>other</td>
<td>Other</td>
</tr>
<tr>
<td>cancer_types</td>
<td>lung</td>
<td>Lung cancer</td>
</tr>
<tr>
<td>cancer_types</td>
<td>skin</td>
<td>Skin cancer</td>
</tr>
<tr>
<td>cancer_types</td>
<td>prostate</td>
<td>Prostate cancer</td>
</tr>
<tr>
<td>cancer_types</td>
<td>breast</td>
<td>Breast cancer</td>
</tr>
<tr>
<td>cancer_types</td>
<td>other</td>
<td>Other</td>
</tr>
<tr>
<td>diabetes_types</td>
<td>type_1</td>
<td>Type 1 (Insulin dependent)</td>
</tr>
<tr>
<td>diabetes_types</td>
<td>type_2</td>
<td>Type 2 (Insulin resistant)</td>
</tr>
</tbody>
</table>

\[ \text{selected-at} (\text{space\_delimited\_array}, \text{n}) \]

Returns the string at the \(n\)th position of the \text{space\_delimited\_array}. (The array is zero-indexed.) Returns an empty string if the index does not exist.

This can be used to get the name of a selected choice from a multi-select question. (This is possible because a reference to a select question returns a space-delimited array of choice names.)

\[ \text{Note:} \quad \text{If used to get a choice name from a select question, this function returns the name, not the label, of the selected choice. To get the label in the current language, use} \ j_{r}:\text{choice-name}(). \]
13.2. Functions

What colors do you like?
Select three.
- Red
- Blue
- Yellow
- Green
- Orange
- Purple

Selected colors:
- red
- green
- purple
Chapter 13. Form Operators and Functions

XLSForm

Table 13.3: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>hint</th>
<th>calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>select_multiple</td>
<td>color_prefs</td>
<td>what colors do you like?</td>
<td>Select three.</td>
<td></td>
</tr>
<tr>
<td>calculate</td>
<td>color_0</td>
<td></td>
<td></td>
<td>selected-at(${color_prefs}, 0)</td>
</tr>
<tr>
<td>calculate</td>
<td>color_1</td>
<td></td>
<td></td>
<td>selected-at(${color_prefs}, 1)</td>
</tr>
<tr>
<td>calculate</td>
<td>color_2</td>
<td></td>
<td></td>
<td>selected-at(${color_prefs}, 2)</td>
</tr>
<tr>
<td>note</td>
<td>color_note</td>
<td>selected colors:</td>
<td>${color_0} &lt;br&gt; ${color_1} &lt;br&gt; ${color_2}</td>
<td>&lt;br&gt;</td>
</tr>
</tbody>
</table>

Table 13.4: choices

<table>
<thead>
<tr>
<th>list_name</th>
<th>name</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>colors</td>
<td>red</td>
<td>Red</td>
</tr>
<tr>
<td>colors</td>
<td>blue</td>
<td>Blue</td>
</tr>
<tr>
<td>colors</td>
<td>yellow</td>
<td>Yellow</td>
</tr>
<tr>
<td>colors</td>
<td>green</td>
<td>Green</td>
</tr>
<tr>
<td>colors</td>
<td>orange</td>
<td>Orange</td>
</tr>
<tr>
<td>colors</td>
<td>purple</td>
<td>Purple</td>
</tr>
</tbody>
</table>

\texttt{count-selected} (\texttt{multi_select_question})

Returns the number of choices selected in \texttt{multi_select_question}.
XLSForm

Table 13.5: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>hint</th>
<th>constraint</th>
<th>constraint_message</th>
</tr>
</thead>
<tbody>
<tr>
<td>select_multiple colors</td>
<td>color_prefs</td>
<td>What colors do you like?</td>
<td>Select three.</td>
<td>count-selected(.)=3</td>
<td>Select exactly three.</td>
</tr>
</tbody>
</table>

Table 13.6: choices

<table>
<thead>
<tr>
<th>list_name</th>
<th>name</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>colors</td>
<td>red</td>
<td>Red</td>
</tr>
<tr>
<td>colors</td>
<td>blue</td>
<td>Blue</td>
</tr>
<tr>
<td>colors</td>
<td>yellow</td>
<td>Yellow</td>
</tr>
<tr>
<td>colors</td>
<td>green</td>
<td>Green</td>
</tr>
<tr>
<td>colors</td>
<td>orange</td>
<td>Orange</td>
</tr>
<tr>
<td>colors</td>
<td>purple</td>
<td>Purple</td>
</tr>
</tbody>
</table>

jr:choice-name(choice_name, 'select_question')
Chapter 13. Form Operators and Functions

Returns the label value, in the active language, associated with the choice_name in
the list of choices for the select_question.

**Note:** You have to wrap the select_question reference in quotes.

'${question_name}'

---

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
13.2. Functions

Selected colors:
Red
Green
Purple

¿Qué colores te gustan?
Selección tres.
☑ Rojo
☐ Azul
☐ Amarillo
☑ Verde
☐ Anaranjado
☑ Púrpura
### 13.2. Functions

**XLSForm**

Table 13.7: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label::English</th>
<th>label::Español</th>
<th>hint::English</th>
<th>hint::Español</th>
<th>calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>select_multiple</td>
<td>color_0</td>
<td>¿Qué colores te gustan?</td>
<td>Select three.</td>
<td></td>
<td></td>
<td>jr:choice-name( selected-at(${color_prefs}, 0), '${color_prefs}')</td>
</tr>
<tr>
<td>select_multiple</td>
<td>color_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>jr:choice-name( selected-at(${color_prefs}, 1), '${color_prefs}')</td>
</tr>
<tr>
<td>select_multiple</td>
<td>color_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>jr:choice-name( selected-at(${color_prefs}, 2), '${color_prefs}')</td>
</tr>
<tr>
<td>note</td>
<td>color_0</td>
<td>Colores seleccionados:</td>
<td></td>
<td></td>
<td></td>
<td>${color_0}&lt;br&gt;${color_1}&lt;br&gt;${color_2}</td>
</tr>
</tbody>
</table>

Table 13.8: choices

<table>
<thead>
<tr>
<th>list_name</th>
<th>name</th>
<th>label::English</th>
<th>label::Español</th>
</tr>
</thead>
<tbody>
<tr>
<td>colors</td>
<td>red</td>
<td>Red</td>
<td>Rojo</td>
</tr>
<tr>
<td>colors</td>
<td>blue</td>
<td>Blue</td>
<td>Azul</td>
</tr>
<tr>
<td>colors</td>
<td>yellow</td>
<td>Yellow</td>
<td>Amarillo</td>
</tr>
<tr>
<td>colors</td>
<td>green</td>
<td>Green</td>
<td>Verde</td>
</tr>
<tr>
<td>colors</td>
<td>orange</td>
<td>Orange</td>
<td>Anaranjado</td>
</tr>
<tr>
<td>colors</td>
<td>purple</td>
<td>Purple</td>
<td>Púrpura</td>
</tr>
</tbody>
</table>

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
Repeat groups

Helpful terms

**nodeset**  A collection of XML nodes. In XLSForms, this is typically a collection of response values.

Outside a *repeat group, referring to a question by name* will return a nodeset containing all the responses to that question.

Nodesets can also be created by joining two or more nodes with pipes: `/data/age | /data/name`.

**indexed-repeat** $(name, group, i\[, subGRP, sub_i\[, subSubGRP, subSub_i\]\])$

Returns the response value of question name from the repeat-group group, in iteration i.

Nested repeat groups can be accessed using the sub and sub_sub parameters.

**See also:**

*Referencing repeated questions from inside the repeat*

**XLSForm**

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>re-repeat_count</th>
<th>calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>note</td>
<td>person_list</td>
<td>Please list the names of</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>people in your household.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>begin_repeat</td>
<td>person</td>
<td>Member of household</td>
<td></td>
<td></td>
</tr>
<tr>
<td>text</td>
<td>name</td>
<td>Name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>end_repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>begin_repeat</td>
<td>person_details</td>
<td>Details</td>
<td></td>
<td><strong>count(${person}</strong>)**</td>
</tr>
<tr>
<td>calculate</td>
<td>current_name</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>date</td>
<td>member_bday</td>
<td>Birthday of ${current_name}$</td>
<td></td>
<td><strong>indexed-repeat(${name}, ${person}, position(..))</strong></td>
</tr>
<tr>
<td>end_repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
13.2. Functions

`count(nodeset)`

Returns the number of items in nodeset. This can be used to count the number of repetitions in a *repeat group*.

### XLSForm

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>note</td>
<td>person_list</td>
<td>Please list the names of the people in your household.</td>
<td></td>
</tr>
<tr>
<td>begin_repeat</td>
<td>person</td>
<td>Member of household</td>
<td></td>
</tr>
<tr>
<td>text</td>
<td>name</td>
<td>Name</td>
<td></td>
</tr>
<tr>
<td>end_repeat</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>begin_repeat</td>
<td>person_details</td>
<td>Details</td>
<td>count(${person})</td>
</tr>
<tr>
<td>calculate</td>
<td>current_name</td>
<td></td>
<td>indexed-repeat(${name}, ${person}, position(..))</td>
</tr>
<tr>
<td>date</td>
<td>member_bday</td>
<td>Birthday of ${current_name}</td>
<td></td>
</tr>
<tr>
<td>end_repeat</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

`count-non-empty(nodeset)`

Returns the number of non-empty members of nodeset.

`sum(nodeset)`

Returns the sum of the members of nodeset.

Can be used to *tally responses to a repeated select question*. 
Table 13.9: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>begin_repeat</td>
<td>guest_details</td>
<td>Guest details</td>
<td></td>
</tr>
<tr>
<td>text</td>
<td>guest_name</td>
<td>Guest name</td>
<td></td>
</tr>
<tr>
<td>select_one</td>
<td>meal_options</td>
<td>Meal preference</td>
<td></td>
</tr>
<tr>
<td>calculate</td>
<td>chkn</td>
<td>if(${meal_preference} = 'chicken', 1, 0 )</td>
<td></td>
</tr>
<tr>
<td>calculate</td>
<td>fsh</td>
<td>if(${meal_preference} = 'fish', 1, 0 )</td>
<td></td>
</tr>
<tr>
<td>calculate</td>
<td>veg</td>
<td>if(${meal_preference} = 'vegetarian', 1, 0 )</td>
<td></td>
</tr>
<tr>
<td>end_repeat</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>calculate</td>
<td>chkn_count</td>
<td>sum(${chkn})</td>
<td></td>
</tr>
<tr>
<td>calculate</td>
<td>fsh_count</td>
<td>sum(${fsh})</td>
<td></td>
</tr>
<tr>
<td>calculate</td>
<td>veg_count</td>
<td>sum(${veg})</td>
<td></td>
</tr>
</tbody>
</table>

Table 13.10: choices

<table>
<thead>
<tr>
<th>list_name</th>
<th>name</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>meal_options</td>
<td>chicken</td>
<td>Chicken</td>
</tr>
<tr>
<td>meal_options</td>
<td>fish</td>
<td>Fish</td>
</tr>
<tr>
<td>meal_options</td>
<td>vegetarian</td>
<td>Vegetarian</td>
</tr>
</tbody>
</table>

\textbf{max}(nodeset)

Returns the largest member of nodeset.

Table 13.11: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>begin_repeat</td>
<td>child_questions</td>
<td>Questions about child</td>
<td></td>
</tr>
<tr>
<td>text</td>
<td>child_name</td>
<td>Child’s name</td>
<td></td>
</tr>
<tr>
<td>integer</td>
<td>child_age</td>
<td>Child’s age</td>
<td></td>
</tr>
<tr>
<td>end_repeat</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>calculate</td>
<td>age_of_oldest_child</td>
<td></td>
<td>max(${child_age})</td>
</tr>
</tbody>
</table>
13.2. Functions

\texttt{min(noderset)}

Returns the smallest member of \texttt{noderset}.

**XLSForm**

Table 13.12: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>begin_repeat</td>
<td>child_questions</td>
<td>Questions about child</td>
<td></td>
</tr>
<tr>
<td>text</td>
<td>child_name</td>
<td>Child’s name</td>
<td></td>
</tr>
<tr>
<td>integer</td>
<td>child_age</td>
<td>Child’s age</td>
<td></td>
</tr>
<tr>
<td>end_repeat</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>calculate</td>
<td>age_of_youngest_child</td>
<td></td>
<td>\texttt{min(${child_age}$)}</td>
</tr>
</tbody>
</table>

**Warning:** The \texttt{min()} and \texttt{max()} functions only work sets of numbers. Empty values (that is, \texttt{variables} referencing unanswered questions) are actually empty strings, and will not be automatically converted to zero (0).

13.2.3 Strings

**Searching and matching strings**

\texttt{regex(string, expression)}

Returns True if string is an exact and complete match for expression.

See also:

*Using regular expressions*
## XLSForm

### Table 13.13: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>constraint</th>
<th>constraint_message</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>middle_initial</td>
<td>What is your middle initial?</td>
<td>regex(., '[pL]')</td>
<td>Just the first letter.</td>
</tr>
</tbody>
</table>

**contains**(*string, substring*)

Returns True if the string contains the substring.

**starts-with**(*string, substring*)

Returns True if string begins with substring.

**ends-with**(*string, substring*)

Returns True if the string ends with substring.

**substr**(*string, start[, end]*)

Returns the substring of string beginning at the index start and extending to (but not including) index end (or to the termination of string, if end is not provided). Members of string are zero-indexed.
13.2. Functions

**substring-before**(string, target)

Returns the substring of string *before* the first occurrence of the target substring. If the target is not found, or string begins with the target substring, then this will return an empty string.

**substring-after**(string, target)

Returns the substring of string *after* the first occurrence of the target substring. If the target is not found this will return an empty string.

**translate**(string, fromchars, tochars)

Returns a copy of string, where every occurrence of a character in fromchars is replaced by the corresponding character in tochars. If fromchars is longer than tochars then every occurrence of a character in fromchars that does not have a corresponding character in tochars will be removed.

**string-length**(string)

Returns the number of characters in string. If no value is passed in, returns the number of characters in the value of the question that this function call is tied to which can be useful in a constraint expression.

**normalize-space**(string)

Returns a string with normalized whitespace by stripping leading and trailing whitespace of string and replacing sequences of whitespace characters with a single space. If no value is passed in, normalizes whitespace of the value of the question that this function call is tied to.

**Combining strings**

**concat**(arg [, arg [, arg [, arg [...]]]])

Concatenates one or more arguments into a single string. If any arg is a *nodeset*, the values within the set are concatenated into a string.

**join**(separator, nodeset)

Joins the members of nodeset, using the string separator.

**Converting to and from strings**

**boolean-from-string**(string)

Returns True if string is "true" or "1". Otherwise, False.

**string**(arg)

Converts arg to a string.
13.2.4 Math

**Warning:** Math functions (except `number()`) only work with number values. You can use `number()` to convert a string of digits to a number, but it is usually better to get a number value directly.

Empty values (that is, variables referencing unanswered questions) are actually empty strings, and will not be automatically converted to zero (0).

**Number handling**

`round(number, places)`
Rounds a decimal number to some number of decimal places.

`int(number)`
Truncates the fractional portion of a decimal number to return an integer.

`number(arg)`
Converts arg to number value.

If arg is a string of digits, returns the number value.

If arg is True, returns 1. If arg is False, returns 0.

If arg cannot be converted, returns NaN (not a number).

`digest(data, algorithm, encoding method (optional))`
Computes and returns the hash value of the data string using the indicated hash algorithm string, and encoding this hash value using the optional encoding string.

Options for the algorithm are MD5, SHA-1, SHA-256, SHA-384, SHA-512.

If the third parameter is not specified, the encoding is base64. Valid options for the encoding are base64 and hex.

This function can be useful if, for example, someone wants to build a unique identifier from sensitive data like a national ID number without compromising that data.

See also:
`count()`, `max()`, `min()`, `number()`

**Calculation**

`pow(number, power)`
Raises a number to a power.
13.2. Functions

\texttt{log(number)}
\begin{itemize}
  \item Returns the natural log of number.
\end{itemize}

\texttt{log10(number)}
\begin{itemize}
  \item Returns the base-10 log of number.
\end{itemize}

\texttt{abs(number)}
\begin{itemize}
  \item Returns the absolute value of number.
\end{itemize}

\texttt{sin(number)}
\begin{itemize}
  \item Returns the sine of number.
\end{itemize}

\texttt{cos(number)}
\begin{itemize}
  \item Returns the cosine of number.
\end{itemize}

\texttt{tan(number)}
\begin{itemize}
  \item Returns the tangent of number.
\end{itemize}

\texttt{asin(number)}
\begin{itemize}
  \item Returns the arc sine of number.
\end{itemize}

\texttt{acos(number)}
\begin{itemize}
  \item Returns the arc cosine of number.
\end{itemize}

\texttt{atan(number)}
\begin{itemize}
  \item Returns the arctan of number.
\end{itemize}

\texttt{atan2(y, x)}
\begin{itemize}
  \item Returns the multi-valued inverse tangent of y, x.
\end{itemize}

\texttt{sqrt(number)}
\begin{itemize}
  \item Returns the square root of number.
\end{itemize}

\texttt{exp(x)}
\begin{itemize}
  \item Returns $e^x$.
\end{itemize}

\texttt{exp10(x)}
\begin{itemize}
  \item Returns $10^x$.
\end{itemize}

\texttt{pi()}
\begin{itemize}
  \item Returns an approximation of the mathematical constant $\pi$.
\end{itemize}

\subsection{13.2.5 Date and time}

\texttt{today()}
\begin{itemize}
  \item Returns the current date without a time component.
\end{itemize}

\texttt{now()}
\begin{itemize}
  \item Returns the current datetime in ISO 8601 format, including the timezone.
\end{itemize}
Warning: This function is often misused. Read *when expressions are evaluated* to learn more.

## Converting dates and time

### `decimal-date-time(dateTime)`
Converts dateTime value to the number of days since January 1, 1970 (the Unix Epoch).

This is the inverse of `date()`.

### `date(days)`
Converts an integer representing a number of days from January 1, 1970 (the Unix Epoch) to a standard date value.

This is the inverse of `decimal-date-time()`.

### `decimal-time(time)`
Converts time to a number representing a fractional day. For example, noon is 0.5 and 6:00 PM is 0.75.

### Formatting dates and times for display

### `format-date(date, format)`
Returns date as a string formatted as defined by format.

The following identifiers are used in the format string:

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%Y</td>
<td>4-digit year</td>
</tr>
<tr>
<td>%y</td>
<td>2-digit year</td>
</tr>
<tr>
<td>%m</td>
<td>0-padded month</td>
</tr>
<tr>
<td>%n</td>
<td>numeric month</td>
</tr>
<tr>
<td>%b</td>
<td>short text month (Jan, Feb, Mar...)</td>
</tr>
<tr>
<td>%d</td>
<td>0-padded day of month</td>
</tr>
<tr>
<td>%e</td>
<td>day of month</td>
</tr>
<tr>
<td>%a</td>
<td>short text day (Sun, Mon, Tue...)</td>
</tr>
</tbody>
</table>

Note: Month and day abbreviations are language and locale specific. If form locale can be determined, that locale will be used. Otherwise, the device locale will be used.

### `format-date-time(dateTime, format)`
Returns dateTime as a string formatted as defined by format.

Our documentation is updated frequently. Get the latest version at [https://docs.getodk.org](https://docs.getodk.org).
13.2. Functions

The identifiers list in `format-date()` are available, plus the following:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>%H</code></td>
<td>0-padded hour (24-hr time)</td>
</tr>
<tr>
<td><code>%h</code></td>
<td>hour (24-hr time)</td>
</tr>
<tr>
<td><code>%M</code></td>
<td>0-padded minute</td>
</tr>
<tr>
<td><code>%S</code></td>
<td>0-padded second</td>
</tr>
<tr>
<td><code>%3</code></td>
<td>0-padded millisecond ticks.</td>
</tr>
</tbody>
</table>

13.2.6 Geography

`area(nodeset | geoshape)`

Returns the area, in square meters, of either a nodeset of geopoints or a geoshape value.

It takes into account the circumference of the Earth around the Equator but does not take altitude into account.

`distance(nodeset | geoshape | geotrace)`

Returns the distance, in meters, of either:

- a nodeset of geopoints
- the perimeter of a geoshape
- the length of a geotrace value

It takes into account the circumference of the Earth around the Equator and does not take altitude into account.

13.2.7 Utility

`random()`

Returns a random number between 0.0 (inclusive) and 1.0 (exclusive).

**Warning:** This function is often misused. Read *when expressions are evaluated* to learn more.

`randomize(nodeset, seed)`

Returns a shuffled nodeset.

A shuffle with a numeric seed is deterministic and reproducible.

The primary use for this function is to randomize the order of choices for a select question. The *documentation on select widgets* describes how this is done in XLSForm.
randomize() can only be used in a context where a nodeset is accepted. Note that questions of type calculate cannot reference a nodeset.

uuid([length])
Without argument, returns a random RFC 4122 version 4 compliant UUID.
With an argument it returns a random GUID of specified length.

boolean(arg)
Returns True if arg is:
- a number other than zero
- a non-empty string
- a non-empty collection
- a comparison or expressions that evaluates to True.

Returns False if arg is:
- the number 0
- an empty string
- an empty collection
- a comparison or expression that evaluates to False.

not(arg)
Returns the opposite of boolean(arg).

coalesce(arg, arg)
Returns first non-empty value of the two args. Returns an empty string if both are empty or non-existent.

checklist(min, max, response[, response[, response[, ...]]])
Returns True if the number of responses that are exactly the string "yes" is between min and max, inclusive.

Set min or max to -1 to make the argument not applicable.

weighted-checklist(min, max, response, weight[, response, weight[, response, weight[, ... ]]])
Returns True if the sum of the weights of each response that is exactly the string "yes" is between min and max, inclusive.

Set min or max to -1 to make the argument not

ture()
Evaluates to True.

false()
Evaluates to False.
ODK forms can use datasets in a variety of ways. These datasets can be either internal or external to the form.

Internal datasets are defined in the `choices` sheet of an XLSForm and are typically used as choices for `selects`. You can also define a dataset in the `choices` sheet to `look up values` based on user input. Learn how to define internal datasets in the section on `selects`.

External datasets are useful when:

- data comes from another system. Using data files attached to the form generally requires fewer steps than adding the data to a form definition.
- data changes frequently. One or more data file attached to the form can be updated without modifying the form definition.
- data is reused between forms. It may be easier to attach the same data file to multiple forms instead of copying the data into all the form definitions.
- the same forms are used in different contexts. For example, the exact same form definition could be used in multiple countries with different data files listing regions, local products, etc.

Note: Most mobile devices released in 2019 or later can handle lists of 50,000 or more without slowdowns. If you experience slowdowns, please share the size of the dataset, the device you are using, and any expressions that reference the dataset on the community forum or to support@getodk.org.

14.1 Building selects from CSV files

CSV files can be used as datasets for select questions using `select_one_from_file` or `select_multiple_from_file`. CSV files used this way must have name and label columns. For each row in the dataset, the text in the name column will be used as the value saved.
when that option is selected and the text in the label column will be used to display the option. For select multiples, name must not contain spaces.

These files may also have any number of additional columns used in choice filters or other expressions. The example below uses one select from internal choices followed by selects from two different external CSV files.

Table 14.1: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>choice_filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>select_one states</td>
<td>state</td>
<td>State</td>
<td></td>
</tr>
<tr>
<td>select_one_from_file lgas.csv</td>
<td>local_gov_area</td>
<td>Local Government Area</td>
<td>state=${state}</td>
</tr>
<tr>
<td>select_multiple_from_file wards.csv</td>
<td>wards</td>
<td>Wards</td>
<td>lga=${local_gov_area}</td>
</tr>
</tbody>
</table>

Table 14.2: choices

<table>
<thead>
<tr>
<th>list_name</th>
<th>name</th>
<th>label</th>
<th>population</th>
</tr>
</thead>
<tbody>
<tr>
<td>states</td>
<td>abia</td>
<td>Abia</td>
<td>4112230</td>
</tr>
<tr>
<td>states</td>
<td>ebonyi</td>
<td>Ebonyi</td>
<td>2176947</td>
</tr>
</tbody>
</table>

Table 14.3: lgas.csv

<table>
<thead>
<tr>
<th>name</th>
<th>label</th>
<th>state</th>
</tr>
</thead>
<tbody>
<tr>
<td>aba_n</td>
<td>Aba North</td>
<td>abia</td>
</tr>
<tr>
<td>aba_s</td>
<td>Aba South</td>
<td>abia</td>
</tr>
<tr>
<td>afikpo_n</td>
<td>Afikpo North</td>
<td>ebonyi</td>
</tr>
</tbody>
</table>

Table 14.4: wards.csv

<table>
<thead>
<tr>
<th>name</th>
<th>label</th>
<th>lga</th>
</tr>
</thead>
<tbody>
<tr>
<td>eziama</td>
<td>Eziana</td>
<td>aba_n</td>
</tr>
<tr>
<td>umuogor</td>
<td>Umuogor</td>
<td>aba_n</td>
</tr>
<tr>
<td>ezeke_amasiri</td>
<td>Ezeke amasiri</td>
<td>afikpo_n</td>
</tr>
<tr>
<td>poperi_amasiri</td>
<td>Poperi amasiri</td>
<td>afikpo_n</td>
</tr>
</tbody>
</table>

14.2 Building selects from XML files

XML files can be used as datasets that populate select questions using select_one_from_file or select_multiple_from_file. This is typically less convenient than using CSV files. However, knowing about the XML representation is helpful for understanding how to reference values in both CSV and XML files.
14.3 Referencing values in datasets

XML files used for selects must have the following structure and can have any number of item blocks:

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

The item blocks are analogous to rows in the CSV representation. Each item must have at least name and label nested nodes and can have any number of additional nodes. These nodes correspond to columns in the CSV representation.

### 14.3 Referencing values in datasets

**XPath paths** can be used to reference values in internal or external datasets. These paths will start with the `instance(<instance name>)` function to identify which dataset is being accessed. The next part of the path is generally `/root/item` because of the XML structure used to represent datasets for selects. The only exception is when using custom XML files which may have arbitrary schemas if not used for selects.

For internal datasets, the instance name is the list_name specified on the choices sheet. For example, to reference the population of the selected state given the form above, the instance name to use is states. The expression would be `instance("states")/root/item[name = ${state}]/population`. To understand this expression better, read the section on XPath paths and especially the subsection about XPath paths for filtering. You could also do things like count the number of states with a population above a certain threshold using an expression like `count(instance("states")/root/item[population > ${pop_threshold}])`.

**Note:** Due to a pyxform bug, it is necessary for there to be some value in the choice_filter column (for at least one question) when referencing internal datasets. If none of the questions in your form need filtering, simply put `true()` as the choice_filter value.

For external datasets, the instance name is the filename specified in the select_one_from_file or select_multiple_from_file declaration without the file extension. For example, to look up a ward’s label given the form above, the instance name to use is wards because the filename referenced is wards.csv. The expression would be `instance("wards")/root/item[name = ${ward}]/label`.

---

Our documentation is updated frequently. Get the latest version at [https://docs.getodk.org](https://docs.getodk.org).
FORM STYLING

Questions can include *Media* such as images, sound or video. Additionally, labels, hints, and choices in an *XLSForm* can all be styled using *Markdown, fonts and colors*, and *Emoji*.

### 15.1 Media

You can include questions in your form that display images or that play video or audio files by including a media column in your *XLSForm*. Files referenced should be included *in your form’s media folder*.

#### 15.1.1 Images

Adding an image to a question displays the image as part of the question.
XLSForm

Table 15.1: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>media::image</th>
</tr>
</thead>
<tbody>
<tr>
<td>select_one yesnodk</td>
<td>coffee</td>
<td>Do you want coffee?</td>
<td>mug.jpg</td>
</tr>
</tbody>
</table>

Table 15.2: choices

<table>
<thead>
<tr>
<th>list_name</th>
<th>name</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>yesnodk</td>
<td>y</td>
<td>yes</td>
</tr>
<tr>
<td>yesnodk</td>
<td>n</td>
<td>no</td>
</tr>
<tr>
<td>yesnodk</td>
<td>dk</td>
<td>I don’t know</td>
</tr>
</tbody>
</table>

15.1.2 Audio

Adding audio to a question adds a play/stop button that controls the audio clip.
15.1. Media

XLSForm

Table 15.3: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>media::audio</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>feel</td>
<td>How does this song make you feel?</td>
<td>amazing.mp3</td>
</tr>
</tbody>
</table>

15.1.3 Video

Adding video to a question adds a button that will play the video clip full screen when clicked.
### 15.1.4 Autoplaying Media

Audio and video on questions can also be played automatically when a question is viewed by adding an autoplay column specifying either audio or video.

**XLSForm**

Table 15.5: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>media::audio</th>
<th>autoplay</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>feel</td>
<td>How does this song make you feel?</td>
<td>amazing.mp3</td>
<td>audio</td>
</tr>
</tbody>
</table>
• Audio/video included in select choices will be autoplayed after the question’s media in display order
• If using a field-list appearance for a group no media will be autoplayed
• Appearances for selects that hide buttons will disable autopl ay for media

15.2 Markdown

*XLSForm* supports limited used of *Markdown*.

15.2.1 Headers

Labels and hints can be styled with one of six header levels.

```
# Header H1
## Header H2
### Header H3
#### Header H4
##### Header H5
###### Header H6
```
**Warning:** If a Markdown header is used, the label or hint can only be one line of text. Line breaks in the XLSForm cell will break the header styling.
Fig. 15.1: A comparison of headline sizes. This exact effect *cannot be produced using Markdown.*
15.2.2 Emphasis

Collect’s Markdown support also includes **bold** and *italic* styling.
15.2. Markdown

/_italic_/  
*_italic_*  
__/bold__/  
**bold**

Note: The label of a form widget is already bold, so bolding text within the label has no effect. Similarly, the hint text of a form widget is already in italics, so italicizing text within the hint has no effect.

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>hint</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>note</td>
<td>emphasis</td>
<td>This label has <strong>bold</strong> and <em>italic</em> text.</td>
<td>This hint has <strong>bold</strong> and <em>italic</em> text.</td>
</tr>
</tbody>
</table>
15.2.3 Hyperlinks

Collect’s Markdown support include hyperlinks, which will open in the device’s default browser.

[Link anchor text](link.url)

XLSForm

Table 15.7: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>note</td>
<td>hyperlink</td>
<td>This label <a href="http://example.com">contains a link</a>.</td>
<td>This hint <a href="http://example.com">contains a link</a>.</td>
</tr>
</tbody>
</table>

15.2.4 Escaping Markdown

New in version 1.15.

If you want to include literal asterisks or underscores, escape them with a back-slash (\). If you want to include a literal back-slash, you’ll need to escape that too.
15.3. Fonts and colors

XLSForm

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>note</td>
<td>es-cape_md</td>
<td># This headline is normal</td>
<td><em>Asterisks</em> and <em>underscores</em> and one slash: \</td>
</tr>
</tbody>
</table>

15.2.5 Inline HTML

Many Markdown implementations support inline HTML, but Collect only supports a small subset of HTML elements. Support of HTML is further limited because:

- Your exact Android device, operating system version, and other device-related factors will affect what HTML can be rendered, and how it is rendered.

- HTML is not supported by other form rendering tools in the XForms ecosystem. For example, HTML elements that work in Collect may not work in Enketo.

For these reasons, we do not recommend using HTML in forms (except the `<span>` element noted below).

See also:

The list of HTML tags currently supported in Collect.

15.3 Fonts and colors

To add custom styling to hint, label, and choice labels, use the `style` attribute on a span tag. The style attribute accepts CSS-like key-value pairs for setting color and font-family.

- For color, try one of the named HTML color values or use a hex color.

- For font-family, it is best to use generic font categories rather than specific fonts:
  - serif
  - sans-serif
  - monospace
  - cursive
  - fantasy

This will ensure support across most devices. You can also use specific font choices, but you should test these on the actual devices being used.
# Going red

Going `<span style="color:red">red</span>`

# Going green

Going `<span style="color:#008000">green</span>`
15.3. Fonts and colors

Cursive text

<span style="font-family:cursive">Cursive text</span>

Formatting works on labels for Choices also.

- Yes
- No
XLSForm

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>note</td>
<td>red</td>
<td>Going &lt;span style=&quot;color:red&quot;&gt;red&lt;/span&gt;</td>
</tr>
<tr>
<td>note</td>
<td>green</td>
<td>Going &lt;span style=&quot;color:#008000&quot;&gt;green&lt;/span&gt;</td>
</tr>
<tr>
<td>note</td>
<td>cursive</td>
<td>&lt;span style=&quot;font-family:cursive&quot;&gt;Cursive text&lt;/span&gt;</td>
</tr>
<tr>
<td>select_one</td>
<td>colored_choices</td>
<td>Formatting works on labels for Choices also.</td>
</tr>
<tr>
<td>note</td>
<td>combo</td>
<td># &lt;span style=&quot;font-family:cursive;color:purple&quot;&gt;Color and font styling can be combined.&lt;/span&gt;</td>
</tr>
</tbody>
</table>

**Table 15.8: choices**

<table>
<thead>
<tr>
<th>list_name</th>
<th>name</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>yn</td>
<td>yes</td>
<td>&lt;span style=&quot;color:green&quot;&gt;Yes&lt;/span&gt;</td>
</tr>
<tr>
<td>yn</td>
<td>no</td>
<td>&lt;span style=&quot;color:red&quot;&gt;No&lt;/span&gt;</td>
</tr>
</tbody>
</table>
15.4 Text alignment

To add alignment to hint, label, and choice labels, use the style attribute on a p or div tag.

Table 15.9: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>select_one</td>
<td>select_question</td>
<td>&lt;p style=&quot;text-align:center&quot;&gt;Centered label&lt;/p&gt;</td>
<td>&lt;p style=&quot;text-align:center&quot;&gt;Centered hint&lt;/p&gt;</td>
</tr>
</tbody>
</table>

Table 15.10: choices

<table>
<thead>
<tr>
<th>list_name</th>
<th>name</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>options</td>
<td>a</td>
<td>&lt;p style=&quot;text-align:center&quot;&gt;a&lt;/p&gt;</td>
</tr>
<tr>
<td>options</td>
<td>a</td>
<td>&lt;p style=&quot;text-align:center&quot;&gt;b&lt;/p&gt;</td>
</tr>
<tr>
<td>options</td>
<td>c</td>
<td>&lt;p style=&quot;text-align:center&quot;&gt;c&lt;/p&gt;</td>
</tr>
</tbody>
</table>

Note: The style will be applied to the list of selected choices and that this might have unexpected results especially if used with a select_multiple question.
15.5 Emoji

Emoji can be used in form labels, hints, and answer choices.

Note: The exact visual representation of each emoji character is controlled by the device operating system, and may vary from device to device. If possible, you should check how your rendered forms look on the devices you are using for data collection.

| XLSForm |

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>select_one</td>
<td>pain_level</td>
<td>What is your current pain level?</td>
</tr>
</tbody>
</table>
Table 15.12: choices

<table>
<thead>
<tr>
<th>list_name</th>
<th>name</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>pain</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>pain</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>pain</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>pain</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>pain</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>pain</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>pain</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

See also:

- Styling prompts in XLSForm
- Sample XLSForm with Style
ODK Collect and XLSForm support multi-language forms.

To add additional languages to your XLSForm, add columns of user-facing content with language-specific columns. All columns representing user-facing text or media can be multi-lingual:

- label
- hint
- media::*
- constraint_message
- required_message

Each language column adds two colons and the language name, followed by the two letter language code in parentheses. For example:

- label::English (en)
- hint::French (fr)
- media::image::Español (es)

Note: The text shown in Collect’s user interface (e.g., buttons, menus, dialogs) is controlled by application, not the form. See user interface settings for how to change the interface language.

XLSForm — Single language

Table 16.1: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>hint</th>
</tr>
</thead>
<tbody>
<tr>
<td>select_multiple colors</td>
<td>color_prefs</td>
<td>What colors do you like?</td>
<td>Select three.</td>
</tr>
</tbody>
</table>
Table 16.2: choices

<table>
<thead>
<tr>
<th>header</th>
<th>name</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>list_name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>colors</td>
<td>red</td>
<td>Red</td>
</tr>
<tr>
<td>colors</td>
<td>blue</td>
<td>Blue</td>
</tr>
<tr>
<td>colors</td>
<td>yellow</td>
<td>Yellow</td>
</tr>
<tr>
<td>colors</td>
<td>green</td>
<td>Green</td>
</tr>
<tr>
<td>colors</td>
<td>orange</td>
<td>Orange</td>
</tr>
<tr>
<td>colors</td>
<td>purple</td>
<td>Purple</td>
</tr>
</tbody>
</table>

XLSForm — Multiple languages

Table 16.3: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label::English (en)</th>
<th>label::Español (es)</th>
<th>hint::English (en)</th>
<th>hint::Español (es)</th>
</tr>
</thead>
<tbody>
<tr>
<td>select_multiple colors</td>
<td>color_prefs</td>
<td>What colors do you like?</td>
<td>¿Qué colores te gustan?</td>
<td>Select three.</td>
<td>Seleccione tres.</td>
</tr>
</tbody>
</table>

Table 16.4: choices

<table>
<thead>
<tr>
<th>list_name</th>
<th>name</th>
<th>label::English (en)</th>
<th>label::Español (es)</th>
</tr>
</thead>
<tbody>
<tr>
<td>colors</td>
<td>red</td>
<td>Red</td>
<td>Rojo</td>
</tr>
<tr>
<td>colors</td>
<td>blue</td>
<td>Blue</td>
<td>Azul</td>
</tr>
<tr>
<td>colors</td>
<td>yellow</td>
<td>Yellow</td>
<td>Amarillo</td>
</tr>
<tr>
<td>colors</td>
<td>green</td>
<td>Green</td>
<td>Verde</td>
</tr>
<tr>
<td>colors</td>
<td>orange</td>
<td>Orange</td>
<td>Anaranjado</td>
</tr>
<tr>
<td>colors</td>
<td>purple</td>
<td>Purple</td>
<td>Púrpura</td>
</tr>
</tbody>
</table>
What colors do you like?
Select three.
- Red
- Blue
- Yellow
- Green
- Orange
- Purple

¿Qué colores te gustan?
Seleccione tres.
- Rojo
- Azul
- Amarillo
- Verde
- Anaranjado
- Púrpura
Warning: There is no fallback language.

If you have specified languages for a column, the non-specific version of that column will be treated as if it were a separate language. (The Change Language menu will list it as Default.)

To avoid this, all columns that can be made multi-lingual need to be created as such for a multi-language form. For example, even if using the same image for a question prompt you will need a media::image::* column for each language. However, you may provide the same media filename for each.

Blank cells in a language-specific column will be blank in the form when that language is active, even if the ”default” column has a value.

### XLSForm — Multiple languages with media example

#### Table 16.5: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label::English (en)</th>
<th>label::Español (es)</th>
<th>media::image::Español (es)</th>
<th>media::image::English (en)</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>coffee</td>
<td>Do you want coffee?</td>
<td>¿Quieres café?</td>
<td>mug_es.jpg</td>
<td>mug_en.jpg</td>
</tr>
</tbody>
</table>

### 16.1 Switching languages

If your form defines multiple languages and you know most of your data collectors will need to use the same one, you should set an explicit default language. You can do this in your XLSForm’s settings sheet:

### XLSForm — setting a default language

#### Table 16.6: settings

<table>
<thead>
<tr>
<th>form_id</th>
<th>version</th>
<th>default_language</th>
</tr>
</thead>
<tbody>
<tr>
<td>my_form</td>
<td>2024050301</td>
<td>Español (es)</td>
</tr>
</tbody>
</table>

Otherwise, Collect will default to the first language defined.

To switch between available languages on a form, go to → Change Language.
16.1. Switching languages

**Note:** Collect will remember the last language you switched to on a form, even if you switch device language.

Changing the form’s language display will not change the device language. If you are in a context that requires switching languages often, make sure you know where to do this in your device’s *Settings* menu.
Chapter 16. Form Language

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
Collect can log the behavior of enumerators as they navigate through a form. This log has many uses including discovering:

- questions that take a long time to answer
- how enumerators typically navigate through a form
- enumerators who take a particularly long or short time to answer
- if enumerators were at the correct location when filling out a form
- when, why and who changed answers

This information can inform form design and training or feed into data validation processes.

See also:

*Lightweight timestamping*

**Warning:** If using Aggregate, Aggregate 1.5.0+ required

If a version of Aggregate lower than 1.5.0 is used, audit files will not be saved on the server.

- *Enabling audit logging*
  - *Location tracking*
  - *Change tracking*
  - *Reason for changes*
  - *Enumerator identification*
- *Viewing audit logs*
- *Log structure*
Chapter 17. Logging enumerator behavior

- Event types
- Timestamps
- Known limitations

17.1 Enabling audit logging

To enable logging for a form, add a row of type audit and name audit in an XLSForm:

Table 17.1: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>audit</td>
<td>audit</td>
</tr>
</tbody>
</table>

A form may contain at most one row of type audit.

17.1.1 Location tracking

You may add the location of events to the log. To do this, add the following parameters to the XLSForm. All three parameters are required.

location-priority high-accuracy: The most accurate location provided by the device, regardless of power use.

balanced: Block level accuracy (~100 meters). Uses less power than high-accuracy.

low-power: City level accuracy (~10 kilometers). Uses less power than balanced.

no-power: No locations will be returned unless another application on the device has requested location updates. Uses no additional power.

location-min-interval The desired minimum time, in seconds, at which location updates will be fetched by the device.

location-max-age The maximum time, in seconds, locations will be considered valid by the device. Must be greater than or equal to location-min-interval.

Table 17.2: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>audit</td>
<td>audit</td>
<td>location-priority=balanced</td>
</tr>
<tr>
<td></td>
<td></td>
<td>location-min-interval=60</td>
</tr>
<tr>
<td></td>
<td></td>
<td>location-max-age=120</td>
</tr>
</tbody>
</table>

When location tracking is enabled, ODK Collect requests location updates from Android periodically, with an interval determined by location-min-interval. The requests are sent

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
17.1. Enabling audit logging

with location-priority to ensure Android does not use more power than is desired.

When Collect receives the location updates, it stores the locations in a timestamped cache. At the time of an event, Collect checks the cache for locations stored over the last location-max-age and returns the most accurate location in the cache.

For the most accurate locations, set location-priority to *high-accuracy*. For the most recent locations, use low numbers for location-min-interval and location-max-age.

**Note:** since v1.30, when a mock location provider is detected, the accuracy is set to 0. Achieving such perfect accuracy is not possible using GPS so that indicates it comes from a mock provider.

**Warning:** Location tracking can be an invasion of privacy. Users of ODK Collect will be informed that their location is being tracked when they open a form with this feature enabled.

Users can control their privacy by disabling location providers in Android, refusing to grant Collect location permissions, or by disabling location tracking of specific forms in Collect.

Disabling location tracking will not prevent users from filling out forms, but these changes are logged as events in the log.

17.1.2 Change tracking

You can enable change tracking so that old answers and new answers will be added to the question events. To do this, add the following parameter to the XLSForm: `track-changes=true`.

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>audit</td>
<td>audit</td>
<td>track-changes=true</td>
</tr>
</tbody>
</table>

17.1.3 Reason for changes

New in version 1.25: ODK Collect v1.25.0

You can add to `track-changes-reasons=on-form-edit` to prompt enumerators to enter a reason before they save changes to a form:
Table 17.4: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>audit</td>
<td>audit</td>
<td>track-changes-reasons=on-form-edit</td>
</tr>
</tbody>
</table>

This will prevent filled out forms being edited without a reason being given. If a reason is given the form will be saved normally and the audit log will include a change reason event with the reason recorded in the change-reason column.

### 17.1.4 Enumerator identification

**ODK Collect v1.25.0**

If your form needs a record of the identity of the enumerator you can use identify-user=true.

Table 17.5: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>audit</td>
<td>audit</td>
<td>identify-user=true</td>
</tr>
</tbody>
</table>

This will cause Collect to prompt the enumerator for their identity before filling out or editing a form instance. In the audit log, a user column will be included that will be populated for each event. The enumerator will not be able to fill in or edit the form without entering a non-blank identity.

**Tip:** identify-user is useful for data collection workflows where devices might be passed between multiple enumerators for data verification or completion.

In cases where a device will only ever used by a single enumerator, it might make more sense to use *username metadata*. This will write the username to each submission instead of to the audit log.

### 17.2 Viewing audit logs

Central will export a CSV with audits from all submissions if an export is requested for a form with an audit.

If using Aggregate, audit logs can be reviewed and downloaded for further analysis using Briefcase.

In Aggregate 1.5.0+, audit logs can be viewed by clicking on the media icon in the meta audit column on the Submissions page:
17.3 Log structure

This displays a popup with the audit contents:

Audit CSV contents

<table>
<thead>
<tr>
<th>Event</th>
<th>Node</th>
<th>Start</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>form start</td>
<td>/data/text_field</td>
<td>Mon Mar 19 11:46:24 GMT-700 2018</td>
<td>N/A</td>
</tr>
<tr>
<td>form save</td>
<td>Mon Mar 19 11:46:26 GMT-700 2018</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>form save</td>
<td>Mon Mar 19 11:46:33 GMT-700 2018</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>form exit</td>
<td>Mon Mar 19 11:46:33 GMT-700 2018</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>form finalize</td>
<td>Mon Mar 19 11:46:33 GMT-700 2018</td>
<td>N/A</td>
<td></td>
</tr>
</tbody>
</table>

**Tip:** Aggregate currently only displays event, node, start, and end in the audit popup. To view locations, changed answers, or to perform more sophisticated analysis, logs can be downloaded along with their submissions using *Briefcase.*

17.3 Log structure

If a form includes an audit, Collect will create an `audit.csv` file as the form is filled out. The `audit.csv` file has the following structure:
Values in the event column represent a particular user action such as opening a form, saving a form, or displaying a question. Possible event types are described in the Event types section.

Values in the node column represent the node in the form that the event refers to, if applicable.

Values in the start and end columns are timestamps represented as the number of milliseconds since midnight, January 1, 1970 UTC. This is known as epoch time and provides a standard way of representing date/time even across timezones. The Timestamps section contains more information about timestamps.

If both location tracking and change tracking are enabled in the log, the CSV will look like this:

<table>
<thead>
<tr>
<th>event</th>
<th>node</th>
<th>start</th>
<th>end</th>
</tr>
</thead>
<tbody>
<tr>
<td>question</td>
<td>/data/name</td>
<td>1523403169208</td>
<td>1523403170733</td>
</tr>
</tbody>
</table>
### Table 17.7: audit.csv

<table>
<thead>
<tr>
<th>event</th>
<th>node</th>
<th>start</th>
<th>end</th>
<th>latitude</th>
<th>longitude</th>
<th>accuracy</th>
<th>old-value</th>
<th>new-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>form start</td>
<td></td>
<td>1550615022663</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>location tracking enabled</td>
<td></td>
<td>1550615022671</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>question</td>
<td>/data/name</td>
<td>1550615026882</td>
<td>1550615097082</td>
<td>37.4229983</td>
<td>-122.084</td>
<td>14.086999893188477</td>
<td>John</td>
<td></td>
</tr>
<tr>
<td>location permissions granted</td>
<td></td>
<td>1550615068610</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>location providers enabled</td>
<td></td>
<td>1550615068665</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>location tracking disabled</td>
<td></td>
<td>1550615095914</td>
<td>37.4229983</td>
<td>-122.084</td>
<td>14.086999893188477</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>question</td>
<td>/data/age</td>
<td>1550615076850</td>
<td>1550615097655</td>
<td>37.4229983</td>
<td>-122.084</td>
<td>14.086999893188477</td>
<td>John Smith</td>
<td></td>
</tr>
<tr>
<td>location tracking enabled</td>
<td></td>
<td>1550615099271</td>
<td>37.4229983</td>
<td>-122.084</td>
<td>14.086999893188477</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>question</td>
<td>/data/age</td>
<td>1550615102351</td>
<td>1550615107630</td>
<td>37.4229983</td>
<td>-122.084</td>
<td>14.086999893188477</td>
<td></td>
<td></td>
</tr>
<tr>
<td>end screen</td>
<td></td>
<td>1550615109199</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>form save</td>
<td></td>
<td>1550615109199</td>
<td>37.4229983</td>
<td>-122.084</td>
<td>14.086999893188477</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>form exit</td>
<td></td>
<td>1550615109199</td>
<td>37.4229983</td>
<td>-122.084</td>
<td>14.086999893188477</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>form finalize</td>
<td></td>
<td>1550615109199</td>
<td>37.4229983</td>
<td>-122.084</td>
<td>14.086999893188477</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Values in the latitude and longitude columns represent the latitude and longitude in decimal degrees. Values in the accuracy column represents accuracy in seconds.

**Note:** Locations will often be repeated in the log. This is because locations are not captured at the time of the event, but rather retrieved from a cache of the most accurate points captured over the last location-max-age.
Chapter 17. Logging enumerator behavior

**Note:** Answers will be recorded only if they differ (if the new answer is different than the old one), otherwise, cells should be empty. Answers which contain commas will be surrounded by double quotes.

### 17.4 Event types

The event column of the audit log can have the following values:
## 17.4. Event types

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
<th>Node?</th>
<th>Timestamps?</th>
<th>Coordinates?</th>
<th>Answers?</th>
</tr>
</thead>
<tbody>
<tr>
<td>form start</td>
<td>Start filling in the form</td>
<td>No</td>
<td>start only</td>
<td>If enabled and available</td>
<td>No</td>
</tr>
<tr>
<td>question</td>
<td>View a question</td>
<td>Yes</td>
<td>Yes</td>
<td>If enabled and available</td>
<td>No</td>
</tr>
<tr>
<td>group questions</td>
<td>View multiple questions on one screen (field-list)</td>
<td>Yes</td>
<td>Yes</td>
<td>If enabled and available</td>
<td>No</td>
</tr>
<tr>
<td>jump</td>
<td>View the jump screen</td>
<td>No</td>
<td>start only</td>
<td>If enabled and available</td>
<td>No</td>
</tr>
<tr>
<td>add repeat</td>
<td>Add a repeat</td>
<td>Yes</td>
<td>Yes</td>
<td>If enabled and available</td>
<td>No</td>
</tr>
<tr>
<td>delete repeat</td>
<td>Delete a repeat</td>
<td>Yes</td>
<td>Yes</td>
<td>If enabled and available</td>
<td>No</td>
</tr>
<tr>
<td>end screen</td>
<td>View the end screen</td>
<td>No</td>
<td>Yes</td>
<td>If enabled and available</td>
<td>No</td>
</tr>
<tr>
<td>form save</td>
<td>Save the form</td>
<td>No</td>
<td>start only</td>
<td>If enabled and available</td>
<td>No</td>
</tr>
<tr>
<td>form exit</td>
<td>Exit the form</td>
<td>No</td>
<td>start only</td>
<td>If enabled and available</td>
<td>No</td>
</tr>
<tr>
<td>form resume</td>
<td>Resume the form</td>
<td>No</td>
<td>start only</td>
<td>If enabled and available</td>
<td>No</td>
</tr>
<tr>
<td>form finalize</td>
<td>Finalize the form</td>
<td>No</td>
<td>start only</td>
<td>If enabled and available</td>
<td>No</td>
</tr>
<tr>
<td>save error</td>
<td>Error trying to save</td>
<td>No</td>
<td>start only</td>
<td>If enabled and available</td>
<td>No</td>
</tr>
<tr>
<td>finalize error</td>
<td>Error trying to finalize the form (probably encryption related)</td>
<td>No</td>
<td>start only</td>
<td>If enabled and available</td>
<td>No</td>
</tr>
<tr>
<td>constraint error</td>
<td>Constraint or required error on finalize</td>
<td>No</td>
<td>start only</td>
<td>If enabled and available</td>
<td>No</td>
</tr>
<tr>
<td>location tracking</td>
<td>Toggle location tracking in Collect</td>
<td>No</td>
<td>Yes</td>
<td>If enabled and available</td>
<td>No</td>
</tr>
<tr>
<td>location providers</td>
<td>Toggle location providers in Android</td>
<td>No</td>
<td>Yes</td>
<td>If enabled and available</td>
<td>No</td>
</tr>
</tbody>
</table>

The documentation is updated frequently. Get the latest version at https://docs.getodk.org.
17.5 Timestamps

If we relied entirely on the time reported by the device for timestamps, users or the network could change the device time and manipulate the correctness of the audit log. For this reason, we only use device time for the form start timestamp. All subsequent event timestamps are the result of elapsed time, which users cannot change, added to the form start timestamp. This means that while the timestamps themselves may potentially be inaccurate, the time elapsed within and between the timestamps are always accurate within one form editing session.

Using epoch time makes it easy to compute elapsed time by subtracting start from end. For example, given the following log:

<table>
<thead>
<tr>
<th>event</th>
<th>node</th>
<th>start</th>
<th>end</th>
</tr>
</thead>
<tbody>
<tr>
<td>form start</td>
<td></td>
<td>1488761807863</td>
<td></td>
</tr>
<tr>
<td>question</td>
<td>/data/name</td>
<td>1488761807868</td>
<td>1488761809157</td>
</tr>
</tbody>
</table>

The enumerator spent 1488761809157 - 1488761807868 = 1289 milliseconds on the screen showing the /data/name question. This corresponds to 1289 / 1000 = 1.289 seconds.

To convert from epoch time to time in UTC in most common spreadsheet programs, divide the epoch time by 86400000 ms per day and add 25569 days between January 1, 1900 (what spreadsheet programs use as "day zero") and January 1, 1970. For example, to convert the timestamp 1488761807868:

\[
(1488761807868 / 86400000) + 25569 = 42800.03944
\]

When the cell is set to type date time in common spreadsheet programs, it will show 3/6/2017 0:56:48 UTC. A common workflow if device time is needed in a human-readable format will be to add a column for the calculation above and change that column’s type to date time.

17.6 Known limitations

- If the device is turned off while a form is being filled, Collect will not record a log entry for the screen that was shown at the time of device shutdown. Events before and after the shutdown will be logged.

- Editing a saved form that was saved using different audit log options can result in a corrupt audit. It might take place when a user saves a form then updates a form definition (changing audit log options) and tries to edit the saved form.
CHAPTER
EIGHTEEN

LAUNCHING EXTERNAL APPS

• Launching external apps to populate single fields
• Specifying a URI as intent data
• Designing an app to return a single value to Collect
• External apps to populate multiple fields

See also:

Launching ODK Collect from External Apps

18.1 Launching external apps to populate single fields

ODK Collect can launch external applications using the `ex:intentAction(intentExtras)` appearance on `string, integer, decimal, image, video, audio` and `file` fields. The application launched could be one that the enumerator uses and quits without Collect needing any data from it. For example, a form could configure the Maps.me application to provide directions to a destination and not need any information back from the app. An external application could also be used to populate the field that launched it. In order to populate a field, the app that is launched must be designed to return a value as described in the external app design section below.

This feature configures an Android intent. Intents are messaging objects used to request an action from another app component. Learn more in the Android docs.

Collect builds the action using the text immediately after `ex:` and before an opening parenthesis (if it exists). For example, in the string `ex:org.opendatakit.counter(intentExtras)`, the action name is `org.opendatakit.counter`. An action name must include a namespace, such as `org.opendatakit` or `com.google`. For example, `ex:org.opendatakit.counter` is valid but `ex:counter` is not.
Chapter 18. Launching External Apps

The parameters defined in the optional parentheses represent extended data ("extras") to be added to the intent. Extras are specified by a comma-delimited list of name=value pairs. The text to the left of the equals sign represents the extra name and may require a namespace. The text to the right of the equals sign represents the extra value. All extra values are sent as strings.

The values of extras can be:

- XPath expressions referring to other fields and including function calls
- String literals defined in single quotes
- Raw integers or decimals

An extra named value that holds the current value for the current field is always passed with the intent. Additional parameters with user-defined names can also be specified. There are two reserved names: value and uri_data.

Since Collect v1.16.0, the data that the intent operates on can be set by using the reserved uri_data parameter. This is particularly useful for implicit intents such as android.intent.action.SENDTO.

### 18.1.1 XLSForm

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>appearance</th>
</tr>
</thead>
<tbody>
<tr>
<td>integer</td>
<td>counter-click</td>
<td>launch to start the counter app</td>
<td>ex:org.opendatakit.counter(form_id='counter-form', form_name='Counter Form', question_id='1', question_name='Counter')</td>
</tr>
</tbody>
</table>

In the examples above, the extras specified have names form_id, form_name, question_id, and question_name.

### 18.2 Specifying a URI as intent data

Since Collect v1.16.0, the value for the reserved parameter name uri_data is converted to a URI and used as the data for the intent. The intent data determines which application to launch when using implicit intents such as SENDTO. For example:

**ex:**android.intent.action.SENDTO(uri_data='smsto:5555555', sms_body=${message})

Launches a new message in an SMS app with the destination number set to 5555555 and the message body set to the contents of the message field.

**ex:**android.intent.action.SENDTO(uri_data='mailto:example@example.com?subject=${subject}&body=${message})

Launches a new message in an email app with destination address set to...
18.3. Designing an app to return a single value to Collect

example@example.com, the subject set to the contents of the subject field and
the body set to the contents of the message field.

ex:android.intent.action.DIAL(uri_data='tel:5555555') Launches a phone dialer
with the number 5555555 as the number to dial.

Notice that the URI must include a scheme, such as mailto: or https://.

18.3 Designing an app to return a single value to Collect

When an activity that is launched from a string, integer or decimal question returns to
Collect, Collect will look for an intent extra named value and use its value to populate the
field that triggered the application launch. Your app should provide a String extra named
value and set its value to what you want to send to Collect. See a counter app for an
example of how this is done.

When an activity that is launched from an image, video, audio or file question returns to
Collect, Collect will take the contents of the ClipData associated with the return Intent
and save it to a file. Any intent extras are ignored. To set your app’s return Intent ClipData:

```java
Uri uri = ...
Intent returnIntent = new Intent();
returnIntent.clipData = ClipData.newRawUri(null, uri);
intent.addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);
setResult(Activity.RESULT_OK, intent);
finish();
```

If using a file question, all file types will be accepted. If you know that your custom app will
return an image, video or audio file, use the more specific question type so that Collect will
show a type-appropriate preview of the returned file.

Additionally, your external app can receive values from Collect through Intent extras. As
described above, Collect will always provide an extra with name value and the current
value of the field. You can also document additional extras that your app uses for form
designers to specify. There are two reserved names: value and uri_data. All extras sent
to your app from Collect will be of type String. You must document any restrictions on
the extra values and validate them on app launch. For example, you might document that
a test_mode extra accepts values yes or no. Collect passes on any text it is given as extra
values without validation so your app should define fallback behavior in case it is given an
invalid value.

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
18.4 External apps to populate multiple fields

A field-list group can have an intent attribute that allows an external application to populate it. Notice that the ex: prefix used when populating a single field is not included to populate multiple fields.

### 18.4.1 XLSForm

Table 18.2: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>appearance</th>
<th>body::intent</th>
</tr>
</thead>
<tbody>
<tr>
<td>begin_group</td>
<td>my-group</td>
<td>Fields to populate</td>
<td>field-list</td>
<td>org.mycompany.myapp(my_text='Some text', uuid=/myform/meta/instanceID)</td>
</tr>
<tr>
<td>text</td>
<td>some_text</td>
<td>Some text</td>
<td></td>
<td>some_integer=Some integer</td>
</tr>
<tr>
<td>integer</td>
<td>some_integer</td>
<td>Some integer</td>
<td></td>
<td>some_decimal=Some decimal</td>
</tr>
<tr>
<td>decimal</td>
<td>some_decimal</td>
<td>Some decimal</td>
<td></td>
<td>some_image=Some image</td>
</tr>
<tr>
<td>image</td>
<td>some_image</td>
<td>Some image</td>
<td></td>
<td>some_video=Some video</td>
</tr>
<tr>
<td>video</td>
<td>some_video</td>
<td>Some video</td>
<td></td>
<td>some_audio=Some audio</td>
</tr>
<tr>
<td>audio</td>
<td>some_audio</td>
<td>Some audio</td>
<td></td>
<td>some_file=Some file</td>
</tr>
<tr>
<td>file</td>
<td>some_file</td>
<td>Some file</td>
<td></td>
<td></td>
</tr>
<tr>
<td>end_group</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```xml
<group ref="/myform/mygroup" appearance="field-list"
intent="org.mycompany.myapp(my_text='Some text',
uuid=/myform/meta/instanceID)"/>
<label>Fields to populate</label>
<input ref="/myform/mygroup/some_text"
<label>Some text</label>
</input>
<input ref="/myform/mygroup/some_integer"
<label>Some integer</label>
</input>
<input ref="/myform/mygroup/some_decimal"
<label>Some decimal</label>
</input>
<input ref="/myform/mygroup/some_image">
```
18.4. External apps to populate multiple fields

```xml
<group>
  <input ref="/myform/mygroup/some_image">
    <label>Some image</label>
  </input>
  <input ref="/myform/mygroup/some_video">
    <label>Some video</label>
  </input>
  <input ref="/myform/mygroup/some_audio">
    <label>Some audio</label>
  </input>
  <input ref="/myform/mygroup/some_file">
    <label>Some file</label>
  </input>
</group>
```

The **intent** attribute is only used when the group has an appearance of **field-list**. The format and the functionality of the **intent** value is the same as above. If the **Intent** extras returned by the external application contains values with keys that match the type and the name of the sub-fields, then the values from the **Intent** extras overwrite the current values of those sub-fields.

The external app is launched with the parameters that are defined in the intent string plus the values of all the sub-fields that are either text, decimal, integer, or binary (filename is sent). Any other sub-field is invisible to the external app.

Since Collect v1.30.0, it is possible to populate binary fields (**image**, **video**, **audio** or **file**) in field lists. An app that returns one or more binary files to Collect as part of a field list must provide a **content URI** as the value for each **extra** that corresponds to a binary question to populate. Additionally, the external application must specify **ClipData items** for each URI it returns. Your app can then grant Collect temporary permissions to the files using the **Intent.FLAG_GRANT_READ_URI_PERMISSION** intent flag. See a similar code example above.

Typically, an external app creator decides on the names of input and output extras and documents those. Form designers use the names of the expected input extras in the **appearance** of the **field-list** used to launch the external app (e.g. **my_text** in **org.mycompany.myapp(my_text='Some text')** above). Form designers use the names of the expected outputs from the external app to name the questions in the field list.
19.1 ODK Build

*ODK Build* is a form designer with a drag-and-drop user interface. Build is an HTML5 web application and works best for designing simple forms.

Tip: For a more powerful form designer from ODK, try *XLSForm*.

19.1.1 Using ODK Build

Go to [http://build.getodk.org](http://build.getodk.org) and sign in.
Note: There is a downloadable version of ODK Build available.

Tip: Clicking on About ODK Build... option in the Help menu provides basic information about ODK Build.
Creating and Saving forms

Options to create, open and save forms are available in the *File* menu.
• Click on New Form to start creating a new form.

**Tip:** When you start building a form on ODK Build, it is named as Untitled Form. To rename it click on rename next to the form name and give a name to your form.

• Click on Open Form... to open a saved form and edit it. Select the form and click on Open option in the open form window. You can also delete a saved form by clicking on X beside the form name.

• To save a form click on Save. If you want to save an edited version of a previously saved form as a new form, click on Save Form as... and provide a name to the form.

• To save your form on your local machine, click on Save Form to File.... You can open a form saved into your local machine by clicking on Load Form from File....
19.1. ODK Build

**Note:** Forms which are saved to your local machine have extension `.odkbuild` and only these forms can be loaded into ODK Build.

**Form building**

- Add a new prompt, by dragging the elements from the bottom of the screen onto the blank canvas.
- You can remove a prompt by clicking on X sign on the prompt.

- For each prompt, modify its properties on the right portion of the screen.

- Prompts can also be rearranged through drag and drop.
- You can collapse the question prompts by checking the *Collapse Questions* option in the *View* menu.
• Information about prompts is displayed in the properties section.

• Unchecking the *Show Information* option in the *Help* menu will hide the information which is displayed in properties section.
19.1. ODK Build

Upload forms to Aggregate

You can upload a form to Aggregate server by clicking on Upload Form to Aggregate... in the File menu. In the window which opens up, enter the url for your Aggregate server. In the username and password box, you need to enter your Aggregate account credentials with Form Manager or greater capabilities. The account type in Aggregate has to be ODK, not Google. You may leave these blank if your Aggregate instance allows anonymous form uploading.
Export forms

- To view XML for your form, click on Export to XML... in the File menu. You can download the .xml file for your form by clicking on Download option in the output XML window.

- You can download .xlsx file for your form by clicking on Export to XLSForm in the File menu.

Changing form properties

You can change the form properties by clicking Form Properties... in the Edit menu.
A form properties window will appear, where you can enter the instance name, public key and submission url for your form.
Instance name specifies names you want to give to submitted data. You can see more info on public key here. Submission url directs your submissions somewhere other than the Aggregate that supplied the form. This is the ODK Aggregate website url with Aggregate.html replaced by submission.

**Tip:** You can give an instance name which uses calculated expressions to name submissions based on submission data. Note that whatever you put in the instance name box should evaluate to a string.

For example, you might use a concatenation of a unique student ID (sid) with the student name (s_name) as the name of the filled-in form. So you can provide a instance name as: `concat(${sid},' - ', ${s_name})`.

This is implemented in XML as an instanceName field within the meta block. If this value is present and not an empty string (""), it will be used as the name of the filled-in form. Otherwise, the current default naming, based upon the date the form was first saved, will be used.

```xml
<instance>
  <data id="build_example1_1508999324">
    <meta>
      <instanceID/>
      <instanceName/>
    </meta>
    <s_name/>
    <sid/>
  </data>
</instance>
```

Fig. 19.1: Instance name as sid-s_name: 12345-Ankita

**Add, remove and display new languages**

You can add a new language for your form by clicking on ManageTranslations... in the Edit menu.
When you add a new language, the language box will be displayed in the properties section of the form.

You can also remove any translation by clicking on remove option in the Translation box.
You can change the display language for your form by checking the language you want to use as display language in the View menu.

19.2 ODK Validate

*ODK Validate* is a tool that ensures an XForms XML file conforms to the XForms specification.

Validate should be used to check hand-edited XForms. It is not needed when creating forms with *XLSForm* or *ODK Build*, unless you edit those forms manually after creating them.

19.2.1 Setting up Validate
19.2. ODK Validate

**Note:** The Validate file available for download is an executable Java application. Once downloaded, it can be run directly and does not need to be installed.

---

**Before you begin...**

Make sure *Java 8* is installed on your system.

---

1. Download Validate.
2. If you wish, move **Validate** to your **Applications** directory or another suitable location.

---

### 19.2.2 Using Validate

1. Open **Validate**.
2. Find your XForms *.xml* file using **Choose file**, and **Open** it.
3. Review and fix any warning messages.

---

4. If needed, **Validate Again**.
Chapter 19. Form Building Tools

Command line

Validating a single form:

```bash
$ java -jar {path/to/validate-jar-file} {path/to/xform.xml}
```

Validating multiple forms:

```bash
$ java -jar {path/to/validate-jar-file} [--fail-fast] {path/to/xform1.xml} {path/to/xform2.xml} {path/to/xform3.xml}
```

The optional `--fail-fast` flag tells Validate to exit on the first error rather than validating all forms and reporting an error at the end.

**Warning:** This tool validates XML files against the XForms specification. It does not check every detail needed to ensure smooth operation in the ODK ecosystem. For example, Aggregate requires that forms have a unique form ID, which this tool does not check.
20.1 Repeat Recipes and Tips

See also:

Repeating questions describes repeat basics.

- Referencing repeated questions from inside the repeat
- Referencing repeated questions from outside the repeat
- Counting repeats and answers
  - Counting the total number of repeat instances
  - Counting the number of times a particular answer was given
- Using additional repeats to follow up on repeated questions
- Setting a max limit on repetitions
  - Using a constraint to limit repetitions
  - Using relevants to limit repetitions

20.1.1 Referencing repeated questions from inside the repeat

Within a repeat, you can reference other questions in that same repeat instance in the usual manner.
XLSForm

Table 20.1: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>note</td>
<td>child_questions</td>
<td>Please provide the following details about each child in your household.</td>
</tr>
<tr>
<td>begin_repeat</td>
<td>child_details</td>
<td>Children in household</td>
</tr>
<tr>
<td>text</td>
<td>child_first_name</td>
<td>Name</td>
</tr>
<tr>
<td>text</td>
<td>child_age</td>
<td>Age of ${child_first_name}</td>
</tr>
<tr>
<td>end_repeat</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

To reference a question from a different repeat instance, or from outside the repeat, use `indexed-repeat()` and `position()`.

### 20.1.2 Referencing repeated questions from outside the repeat

A question in a repeat can be referenced from outside the repeat with `indexed-repeat(${question-name}, ${repeat-name}, index)`.

### 20.1.3 Counting repeats and answers

**Counting the total number of repeat instances**

Use `count(${name-of-repeat})` to get the number of repeat instances.

**Counting the number of times a particular answer was given**

To count the number of times a specific response is given, add a calculate field inside the repeat which evaluates to 1 or 0 depending on the answer. Then, outside the repeat, calculate the `sum()` of the calculate field.
20.1. Repeat Recipes and Tips

XLSForm

Table 20.2: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>begin_repeat</td>
<td>guest_details</td>
<td>Guest details</td>
<td></td>
</tr>
<tr>
<td>text</td>
<td>guest_name</td>
<td>Guest name</td>
<td></td>
</tr>
<tr>
<td>select_one</td>
<td>meal_options</td>
<td>Meal preference</td>
<td></td>
</tr>
<tr>
<td>calculate</td>
<td>chkn</td>
<td></td>
<td>if(${meal_preference} = 'chicken', 1, 0 )</td>
</tr>
<tr>
<td>calculate</td>
<td>fsh</td>
<td></td>
<td>if(${meal_preference} = 'fish', 1, 0 )</td>
</tr>
<tr>
<td>calculate</td>
<td>veg</td>
<td></td>
<td>if(${meal_preference} = 'vegetarian', 1, 0 )</td>
</tr>
<tr>
<td>end_repeat</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>calculate</td>
<td>chkn_count</td>
<td>sum(${chkn})</td>
<td></td>
</tr>
<tr>
<td>calculate</td>
<td>fsh_count</td>
<td>sum(${fsh})</td>
<td></td>
</tr>
<tr>
<td>calculate</td>
<td>veg_count</td>
<td>sum(${veg})</td>
<td></td>
</tr>
</tbody>
</table>

Table 20.3: choices

<table>
<thead>
<tr>
<th>list_name</th>
<th>name</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>meal_options</td>
<td>chicken</td>
<td>Chicken</td>
</tr>
<tr>
<td>meal_options</td>
<td>fish</td>
<td>Fish</td>
</tr>
<tr>
<td>meal_options</td>
<td>vegetarian</td>
<td>Vegetarian</td>
</tr>
</tbody>
</table>

20.1.4 Using additional repeats to follow up on repeated questions

Sometimes it is convenient to gather an initial set of responses, and then ask more detailed question after you have collected the whole set.

For example:

- collecting the names of all the people in a household, and then asking questions about each person
- collecting the names of each type of crop being grown, and then asking questions about each crop

This can be done by using count() and position(..). count() is used to guarantee that the second repeat has the same number of instances as the original repeat. position(..) provides the index of the repeat instance it was called from. This is used to refer to questions from the first repeat in the follow-up repeat.
XLSForm

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>repeat_count</th>
<th>calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>note</td>
<td>person_list_note</td>
<td>Please list the names of the people in your household.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>begin_repeat</td>
<td>person</td>
<td>Member of household</td>
<td></td>
<td></td>
</tr>
<tr>
<td>text</td>
<td>name</td>
<td>Name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>end_repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>begin_repeat</td>
<td>person_details</td>
<td>Details</td>
<td>count(${person})</td>
<td></td>
</tr>
<tr>
<td>calculate</td>
<td>current_name</td>
<td>$\text{Birthday of } ${current_name}</td>
<td>indexed-repeat(${name}, ${person}, position(..))</td>
<td></td>
</tr>
</tbody>
</table>

20.1.5 Setting a max limit on repetitions

If you want the user to decide how many times to repeat, but you also want to limit the maximum number of repetitions, you have a few options.

Using a constraint to limit repetitions

If the user knows how many repetitions they will complete, you can ask them this in a question before the repeat group and set a constraint on that question.
20.1. Repeat Recipes and Tips

**XLSForm**

Table 20.4: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>constraint</th>
<th>repeat_count</th>
</tr>
</thead>
<tbody>
<tr>
<td>integer</td>
<td>number_in_party</td>
<td>How many guests are in your party?</td>
<td>. &lt;= 8</td>
<td></td>
</tr>
<tr>
<td>note</td>
<td>party_names_note</td>
<td>Please provide details for each guest.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>begin_repeat</td>
<td>guest_details</td>
<td>Guest details</td>
<td></td>
<td>${number_in_party}</td>
</tr>
<tr>
<td>text</td>
<td>guest_name</td>
<td>Guest’s name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>text</td>
<td>guest_dietary</td>
<td>Does this guest have any dietary restrictions?</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:** If the count is decreased by a user, no groups will be deleted. This avoids accidental data loss: a user who accidentally sets the count too low can set it to a higher number and still have the repetitions that were previously created.

A recommended way to handle this case is using relevance to hide any extra values:

Table 20.5: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>constraint</th>
<th>repeat_count</th>
<th>relevant</th>
</tr>
</thead>
<tbody>
<tr>
<td>integer</td>
<td>number_in_party</td>
<td>How many guests are in your party?</td>
<td>. &lt;= 8</td>
<td></td>
<td>position(..) &lt;= ${number_in_party}</td>
</tr>
<tr>
<td>note</td>
<td>party_names_note</td>
<td>Please provide details for each guest.</td>
<td></td>
<td></td>
<td>${number_in_party}</td>
</tr>
<tr>
<td>begin_repeat</td>
<td>guest_details</td>
<td>Guest details</td>
<td></td>
<td>${number_in_party}</td>
<td></td>
</tr>
<tr>
<td>begin_group</td>
<td>guest_details_gr</td>
<td>Guest details</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>text</td>
<td>guest_name</td>
<td>Guest’s name</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>text</td>
<td>guest_dietary</td>
<td>Does this guest have any dietary restrictions?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>end_group</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>end_repeat</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
**Using relevants to limit repetitions**

If asking the user ahead of time doesn’t make sense, another strategy is to manually repeat the question in the form and use the relevant column to *skip repetitions if* the previous question is left blank. This sets a maximum number of responses: the number of times you included the question in the form.

To check if the previous question has a response, *reference the question* in the relevant column.

**XLSForm**

Table 20.6: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>relevant</th>
</tr>
</thead>
<tbody>
<tr>
<td>note</td>
<td>images_note</td>
<td>Take up to five pictures.</td>
<td></td>
</tr>
<tr>
<td>image</td>
<td>image_1</td>
<td>Image 1</td>
<td></td>
</tr>
<tr>
<td>image</td>
<td>image_2</td>
<td>Image 2</td>
<td>${image_1}</td>
</tr>
<tr>
<td>image</td>
<td>image_3</td>
<td>Image 3</td>
<td>${image_2}</td>
</tr>
<tr>
<td>image</td>
<td>image_4</td>
<td>Image 4</td>
<td>${image_3}</td>
</tr>
<tr>
<td>image</td>
<td>image_5</td>
<td>Image 5</td>
<td>${image_4}</td>
</tr>
</tbody>
</table>

This pattern can be combined with *required responses* to enforce a minimum number of responses.

**XLSForm**

Table 20.7: survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>required</th>
<th>relevant</th>
</tr>
</thead>
<tbody>
<tr>
<td>note</td>
<td>images_note</td>
<td>Take 3-5 pictures.</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>image</td>
<td>image_1</td>
<td>Image 1</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>image</td>
<td>image_2</td>
<td>Image 2</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>image</td>
<td>image_3</td>
<td>Image 3</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>image</td>
<td>image_4</td>
<td>Image 4</td>
<td>yes</td>
<td>${image_3}</td>
</tr>
<tr>
<td>image</td>
<td>image_5</td>
<td>Image 5</td>
<td>yes</td>
<td>${image_4}</td>
</tr>
</tbody>
</table>
20.2 Using regular expressions

20.2.1 Regular Expressions

A regular expression is a special text string for describing a search pattern. Regular expressions find an important use in specifying constraints in your forms. You might want to set constraints on the length, character set allowed and various other fields in the data input by the user. All this can be achieved by using regular expression constraints in your form.

Some basic operations on regular expressions are:

1. **Boolean or**: A vertical bar separates alternatives. For example, `gray|grey` can match gray or grey.

2. **Grouping Parentheses**: They are used to define the scope and precedence of the operators. For example, `gray|grey` and `gr(a|e)y` are equivalent patterns which both describe the set of gray or grey.

3. **Quantification**: A quantifier after a token (such as a character) or group specifies how often that preceding element is allowed to occur.
   - Square brackets indicate the occurrence of elements within. For example, `[abc]` matches either a, b or c character, `[a-g]` matches any character from a to g, `[^abc]` matches any character except a,b or c.
   - The question mark indicates zero or one occurrence of the preceding element. For example, `colou?r` matches both color and colour.
   - The asterisk indicates zero or more occurrences of the preceding element. For example, `ab*c` matches ac, abc, abbc, abbbc, and so on.
   - The plus sign indicates one or more occurrences of the preceding element. For example, `ab+c` matches abc, abbc, abbbc, and so on, but not ac.
   - `{n}`: The preceding item is matched exactly n times.
   - `{min,}`: The preceding item is matched min or more times.
   - `{min,max}`: The preceding item is matched at least min times, but not more than max times.

For a clear understanding of regular expressions, try these regex online checker tools:

- [https://regex101.com/](https://regex101.com/)
- [https://www.regextester.com/](https://www.regextester.com/)
- [https://regexr.com/](https://regexr.com/)
20.2.2 Tips on using regular expressions

- Regular expressions only apply to strings. The function used is of the form: `regex(string value, string expression)` where it returns the result of regex test on provided value. If the result is true, the input value is valid and satisfies the constraint. If the result is false, input value is not valid and user may need to re-enter the input. The regex function will be placed in the constraints column in your XLSForm.

- Be careful while setting limits on length of a constraint. For example, if you want to restrict a text input to a maximum of six alphabetic characters, you can do so as follows:

  In XLSForm:

  ```
  Table 20.8: Survey
  
<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>constraint</th>
<th>constraint_message</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>ex</td>
<td>Enter example</td>
<td><code>regex(.,'[a-zA-Z]{0,6}$')</code></td>
<td>Input can have a maximum of 6 alphabetic characters.</td>
</tr>
</tbody>
</table>
  
  In XForm XML:
  ```xml
  <bind constraint="regex(.,'[a-zA-Z]{0,6}$')" jr:constraintMsg="Input can have a maximum of 6 alphabetic characters." nodeset="/regex_ex/ex" type="string"/>
  
  <input ref="/regex_ex/ex">
    <label>Enter example</label>
  </input>
  ```

  Instead if you used a constraint of the form `regex(.,'[a-zA-Z]{6}$')`, it would require an input of exactly six alphabetic characters.

**Note:**

- . denotes the input string.
- ^ and $ denote start and end of string respectively.

- If you want to use a regular expression constraint on a number, make sure that the type of your question is text and appearance is numbers and then apply the constraint.

  The following example will validate a 10-digit North American telephone number. Separators are not required, but can include spaces, hyphens, or periods. Parentheses around the area code are also optional.

  In XLSForm:
### 20.2. Using regular expressions

#### Table 20.9: Survey

<table>
<thead>
<tr>
<th>type</th>
<th>name</th>
<th>label</th>
<th>constraint</th>
<th>constraint_message</th>
<th>appearance</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>tel_no</td>
<td>Enter your Telephone number</td>
<td><code>regex(.,'^((\[0-9\]{1})*[- .]*([0-9]{3})[- .]*[0-9]{3})+$')</code></td>
<td>Telephone numbers should have 10 digits with optional separators.</td>
<td>numbers</td>
</tr>
</tbody>
</table>

In XForm XML:

```xml
<bind constraint="regex(.,'^((\[0-9\]{1})*[- .]*([0-9]{3})[- .]*[0-9]{3})+$')" jr:constraintMsg="Telephone numbers should have 10 digits with optional separators." nodeset="/regex_ex/tel_no" type="string">

<input appearance="numbers" ref="/regex_ex/tel_no">
    <label>Enter your Telephone number</label>
</input>
```

An other alternative to this would be to use a regular expression of the form: `regex(string(.),'...')`. But this should be avoided since the value of `string(.)` would be after whatever you entered was converted to an integer. So if you entered 0004, `string(.)` would be 4.

- Integers are limited by binary representation to 9 decimal digits. If you want something longer (like 10 numbers) then make sure to use a text type with appearance as numbers and add a constraint restricting the input string to be a number. Constraint is required since appearance setting changes the keyboard style of the pop-up keyboard to the number keyboard when you attempt to enter data into the field but does not prevent non-numbers from being entered. This relies upon the device’s keyboard supporting (See this).

For example, a constraint of the form `regex(.,'^[0-9]{11}$')` will restrict the input string to be a number of exactly 11 digits.

- Avoid using complex regex patterns as that may cause stack overflow crashes. Also, avoid placing constraints on names since your regex will certainly not capture all the punctuation or random characters that names can contain and they are hard error-prone and hard to maintain.

See also:

You can refer this list for various common regex patterns.
**ODK BRIEFCASE**

*ODK Briefcase* is a desktop application that runs on macOS, Windows, and Linux. It is used for pulling, pushing, and exporting forms on ODK servers such as *ODK Central* and aggregate-intro. You can also use Briefcase to pull forms directly from *ODK Collect*.

With Briefcase you can:

- Pull forms from *Collect, Central, Aggregate*.
- Push forms to a *Central* or *Aggregate* server.
- Export data from forms to CSV files.
- Decrypt encrypted forms.
- Back up forms from servers.
- Transfer forms between servers.
- Work from a command line.
- Work offline.

Our documentation is updated frequently. Get the latest version at [https://docs.getodk.org](https://docs.getodk.org).
Chapter 21. ODK Briefcase

21.1 Learn more about Briefcase

- Setting Up ODK Briefcase
- Using ODK Briefcase
Before you begin...

We verify Briefcase with Java 11. We recommend installing OpenJDK 11 LTS from AdoptOpenJDK.

1. Download the ODK Briefcase JAR file from GitHub.
   
   Some browsers may warn that JAR files might harm your computer. Do not worry, Briefcase when downloaded from our GitHub page is safe.

2. Double-click the downloaded JAR file or, from the command line, run `java -jar {path/to/ODK-Briefcase-vX.Y.Z.jar}`
   
   macOS will warn that Briefcase is from an unidentified developer. This is normal and expected. Follow these instructions from Apple to open the file.

3. Follow the instructions to start using Briefcase
Chapter 22. Setting Up ODK Briefcase

Welcome to ODK Briefcase! Here are three things you should know to get started.

1. You must set a Storage Location where Briefcase will store data that it needs to operate. You will not be able to use Briefcase until you set this location.

2. We send usage data (e.g., operating system, version number) and crash logs to the core developers to help prioritize features and fixes. If you do not want to contribute your usage data or crash logs, please uncheck that setting.

3. ODK is a community-powered project and the community lives at https://forum.opendatakit.org. Stop by for a visit and introduce yourself!

Use if you wish to pull every submission, regardless of last submission pulled.

Clear pull history

The form list in Briefcase can get out of date if files are moved manually. This reload is always safe.

Reload forms from storage location
CONTRIBUTING TO ODK DOCS

23.1 Docs Contributor Technical Guide

This document explains how to set up your computer and work locally as an ODK Docs contributor. Local set up includes installing some software, and working locally involves:

- writing documentation text or code in a code editor
- using the Terminal (the "Shell" or "Command Line")

We encourage all potential contributors to try to work locally, following this guide.

23.1.1 Before you begin

Learn a little about ODK

Read about the project and the community at ODK’s website.

Get started with the docs by going to the ODK Docs GitHub README.

Set up collaboration accounts

ODK is a collaborative community. Before diving in as a contributor, set up accounts on our three main collaboration platforms, GitHub, the ODK Developer Slack, and the ODK Forum.

Tip: As you are setting up your accounts, keep in mind that it is very helpful (but not required) to use the same (or similar) username on GitHub, the ODK Developer Slack, and the ODK Forum.

This makes it easy for other people to keep track of conversations which sometimes span multiple online platforms.
If you are willing and able to do so, a profile picture in each place is also very helpful. (But it is okay if you are unable or uncomfortable adding a picture.)

1. Set up a GitHub account.
   
   GitHub is a popular code storage and collaboration platform. You will need a GitHub account to contribute to ODK documentation, or any other ODK projects.
   
   • ODK on GitHub
   • ODK Docs on GitHub

2. Join the ODK Developer Slack.
   
   Slack is a popular chat platform. The ODK contributor community uses Slack to discuss development, plan work, and generally keep in touch. If you have a question about how to contribute to ODK Docs, or any other ODK project, the ODK Slack is the best place to ask it.
   
   Conversations related to documentation are held in the #docs-code channel. You may also want to check in with #general and #random.
   
   (a) Get an automated invitation from http://slack.getodk.org
   (b) Check your email for the invitation.
   (c) Follow the link in your email and set up your account.

3. Join the ODK Forum
   
   The ODK Forum is the main place for support questions and conversations that affect the whole ODK community (users and other stakeholders, as well as contributors).
   
   If you have a question about how to use any ODK software, or want to get connected with the larger ODK community, the forum is the best venue for that.
   
   **Tip:** The forum has a search feature, and a long history of archived support posts. When writing new documentation about an existing feature, old forum posts are an excellent source for figuring out what people need to know: If someone has asked a question about it, it should probably be in the documentation.

**Should I ask in the Forum, the Slack, or a GitHub issue?**

The ODK community talks a lot, in a lot of places. Sometimes it’s hard to know where to ask a question.

**Contribution-related questions and problems should be asked in Slack.** This includes things like:

• How do I set up my local editing environment?
• How do I use git?
• I’m having a merge conflict.
• I got an error at the terminal which I don’t understand.
• How do I add a picture to a document?
• What issue should I work on?

Work-specific questions and discussion should take place on the GitHub issue defining the work. This includes things like:

• I’m writing a piece of content, but I’m not sure where it should be organized.
• I’d like to work on this feature, but I don’t know how to implement it.
• Here’s my idea for solving this problem. Is that a good idea?
• I’m going to be working on this for the next few days. No one else should also work on it at the same time.
• I said I was working on this, but I didn’t finish and I’m no longer working on it.

User-related questions and problems should be asked in the Forum. (You should use the search feature first, since someone else may have already asked the same question.) This includes things like:

• How do I install an ODK application?
• How do I create a form?
• How do I add a specific feature to a form?
• My ODK application crashed.

But don’t worry about posting a question in the wrong place.

It is better to ask a question in the “wrong” venue than to not ask the question at all. Many of the same people are present in all three places, and we will help you wherever you happen to show up.

23.1.2 Initial Setup

Note: We generally recommend starting with the Docker platform for editing docs unless you already have a Sphinx environment set up. Local tools and workflows presented in this guide are what the authors feel would be easiest for newcomers and those unfamiliar with open source.
However, developer and authoring tools have a lot of options and alternatives. You should feel free to use your preferred tools.

Before you begin working the first time you will need to install a few tools on your computer. You should only need to do this one time on any computer.

1. Find and open a terminal or command line.
   - Windows
   - Mac
   - Linux

**Windows versions prior to Windows 10**

Use Windows PowerShell. (Not the DOS Prompt.)

We recommend using the Windows PowerShell ISE.

During initial setup (this section of the guide) you will need to Run as Administrator.

Throughout the rest of the instructions in this guide, follow the instructions labeled PowerShell or Windows.

**Windows 10**

In Windows 10, you have a choice:

- Use the Powershell (as described above)
- Use the Windows Subsystem for Linux.

If you decide to use the Powershell, follow the Powershell or Windows instructions throughout the contributor guides.

If you decide to use the Linux subsystem, follow the Bash or Windows instructions throughout the contributor guide.

**Note:** Computers with the Windows 10 Home operating system are incapable of installing some of the tools necessary to edit the docs. Other Windows operating systems, such as Windows 10 Enterprise or Pro, can be used to edit the docs.

Use the Terminal app, or another Bash-like shell.
If you’ve never used it before, the Terminal is probably in the Other directory in your App collection.

Follow the Bash or Mac instructions throughout the contributor guide.

Optional: Install Homebrew

Homebrew is a package manager for Mac OS. It makes it easier to install other apps and tools from the command line.

Follow the installation instructions.

Use a Bash-like shell of your choosing, and follow the Bash or Linux instructions throughout the contributor guide.

You will also need to be familiar with the relevant package manager for your system.

2. Install git.

Git is a version control system. It helps us keep track of changes to the documentation. (Similar to the undo history in a document editing program.)

Linux

Mac

Windows

Use your distribution’s package management system to install git on Linux.
Option 1: Download an installer

(a) Download the git installer for Mac.
(b) Open the installer package.
(c) Follow the prompts.
(d) Accept any default settings.

Option 2: Use Homebrew to install git

$ brew install git

(a) Download the git installer for Windows.
(b) Open the installer package.
(c) Follow the prompts.
(d) Accept any default settings.

3. Install Git LFS

Git Large File Storage (Git LFS) is a tool that helps us manage images, videos, and other files which are neither text nor code.

Linux
Mac
Windows

Use your distribution’s package management system to install Git LFS on Linux.
After initial installation by the package manager, complete the install by running:

$ git lfs install

(a) Download Git LFS from the Git LFS website.
(b) Open the downloaded installer.
(c) Follow the prompts.
(d) Accept any default settings.
(e) Open the Terminal and add LFS to git:

$ git lfs install
Option 2: Use Homebrew to install Git LFS.

```bash
$ brew install git-lfs
$ git lfs install
```

(a) Download Git LFS from the Git LFS website.
(b) Open the downloaded installer.
(c) Follow the prompts.
(d) Accept any default settings.
(e) Open Powershell and add LFS to git:

```bash
> git lfs install
```

4. Install Python 3

Python is a programming language.

Most of the ODK Docs tools are written in Python, so you need it installed on your computer in order to use those tools. (Don’t worry. You don’t need to know how to program in Python.)

We require Python 3, version 3.6 or later.

Linux

Mac

Windows

Use your distribution’s package management system to install Python 3.6+ on Linux.
(For more help, see Installing Python on Linux.)

Tip: Mac OS includes a legacy (outdated) version of Python. It’s best to just ignore it.

Option 1: Use the Python Installer for Mac

(a) Download the latest Python installer for Mac.

64-bit or 32-bit?

Python provides 64-bit and 32-bit installers. You probably need the 64-bit installer.
Chapter 23. Contributing to ODK Docs

If you are running a relatively recent Mac OS update (Mountain Lion or later — any Mac from the last several years) the 64-bit installer is for you.

If you have an older Mac, and are unsure if it can run a 64-bit installer, check the processor details in → About This Mac.

(b) Open the Installer.
(c) Follow the prompts.
(d) Accept the default settings.
(e) Open the Terminal to see if Python installed properly.

```
$ python3 --version
Python 3.7.0
```

The output from `python3 --version` might be a little different, but it should be higher than 3.6.

If you get an error here, something went wrong. Try running the installer again. If the problem persists, and you can’t debug it yourself, asks us about it on ODK Slack.

Option 2: Use Homebrew to install Python 3.6+

```
$ brew install python
.
.
.
$ python3 --version
Python 3.7.0
```

The output from `python3 --version` might be a little different, but it should be higher than 3.6.

If you get an error here, something went wrong. Try running `brew install python` again. If the problem persists, and you can’t debug it yourself, asks us about it on ODK Slack.

(a) Go to the Python Releases for Windows page.

(b) Under the latest numbered release for Python 3, find and download the Windows x86-64 web-based installer (for a 64-bit system) or the Windows x86 web-based installer (for a 32-bit system).

64-bit or 32-bit?
Well over 90% of computers running Windows are 64-bit. So you probably need the 64-bit version.

If you are running a very old or low-powered computer, and you are unsure if it is 64-bit or 32-bit, you can use this guide from HP (which will work for other computer brands) to find that information.

(c) Open the downloaded installer.
(d) Follow the prompts.
(e) Accept all default settings.
(f) Open Powershell and make sure the installation completed.

```bash
> python --version
Python 3.7.0
```

The output from `python --version` might be a little different, but it should be whatever numbered version you downloaded.

If you get an error here, something went wrong. Try running the installer again. You may also have to add Python to your Windows search path. You can do this by going to Advanced System Settings -> Environmental Variables -> Edit System Variables, then adding the path to the directory containing Python. If the problem persists, and you can’t debug it yourself, asks us about it on ODK Slack.

5. Set up your working directory

In whatever directory (folder) on your computer where you organize projects, create a new directory for ODK, and then navigate to that directory. (We recommend calling this directory `odk`, and the rest of the guide will assume that’s what you called it.)

Bash

```
$ mkdir odk
$ cd odk
/odk/
```

PowerShell

```
> mkdir odk
> cd odk
> /odk/
```

For the rest of this guide, we assume you are in the `/odk/` directory, or a subdirectory of it.

6. Set up a virtual environment
A virtual environment is a Python construct that lets you download and install tools for a specific project without installing them for your entire computer.

(a) Create the virtual environment.

Bash

PowerShell

```
/odk/ $ python3 -m venv odkenv
```

(b) Activate the virtual environment.

Bash

PowerShell

```
/odk/ $ source odkenv/bin/activate
```

The (odkenv) before the prompt shows that the virtual environment is active. You will need to have this active any time you are working on the docs.

If the file cannot be found, your activate file may be located under odkenv/scripts/activate.

Later, to deactivate the virtual environment:

Bash

PowerShell

```
/odk/ $ deactivate
```

7. Fork the ODK Docs repository to your own GitHub account.

A repository (repo) is a store of all the code and text for a project. The ODK Docs repo is kept at GitHub.

On GitHub, a fork is a copy of a repo, cloned from one user to another. In order to work on ODK Docs, you will create your own fork.
(a) Go to the ODK Docs repo on GitHub.
(b) Use the Fork button (top right) to create your own copy.
(c) After the process completes, you’ll be looking at your own fork on GitHub.

8. Clone down your copy to your local computer

(a) From your own fork of the repo on GitHub, select the Clone or download button.
(b) Copy the URI from the text box that opens. It will be something like: https://github.com/your-gh-username/docs.git
(c) Use your terminal to clone the repository.

You should already be in the odk directory, with the virtual environment active.

Bash

Powershell

Warning: Some of the git commands produce meaningless errors in PowerShell. If you get an error when using git, but everything seems to work otherwise, ignore the error.

Note: This will cause your computer to download the entire ODK Docs repository, including a large number of images. It will take several minutes to complete.

Your local directory

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
If you followed the instructions, you should now have the following directory structure:

- odk
  - docs
  - odkenv

The odkenv directory stores your virtual environment, and you should not need to open it or directly view its content. Just ignore it.

The docs directory is your copy of the ODK Docs repo. You will do most of your work in this directory.

If you need to download or create additional files which are not actually a part of the ODK Docs repository, keep them out of the docs directory.

You can use the main odk directory for any other files you need to work on. (For example, you may want to create a directory called odk/forms to hold XLSForm and XForm files.)

9. Set the upstream remote

   In git, a remote is a copy of a repo somewhere else. From your local computer’s point of view, your online copy at GitHub is a remote.

   When you cloned down a repo, your local copy gives your GitHub copy the name origin.

   You also need to give the primary ODK Docs repo a name, and our convention is to name it upstream.

   Bash

   ```
   (odkenv) /odk/docs/ $ git remote add upstream https://github.com/getodk/docs.git
   (odkenv) /odk/docs/ $ git remote -v
   origin https://github.com/your-github-username/docs.git (fetch)
   origin https://github.com/your-github-username/docs.git (push)
   upstream https://github.com/getodk/docs.git (fetch)
   upstream https://github.com/getodk/docs.git (push)
   ```

   PowerShell

   ```
   (odkenv) /odk/docs/ > git remote add upstream https://github.com/getodk/docs.git
   (odkenv) /odk/docs/ > git remote -v
   origin https://github.com/your-github-username/docs.git (fetch)
   origin https://github.com/your-github-username/docs.git (push)
   ```
23.1. Docs Contributor Technical Guide

```
upstream https://github.com/getodk/docs.git (fetch)
upstream https://github.com/getodk/docs.git (push)
```

If everything went right, you should see output similar to what is shown above.

10. Install Python tools with pip

   Pip is a package management tool that comes with Python. We use it to download and install our documentation tools. These Python tools are listed in requirements.txt.

   Bash

   ```
   (odkenv) /odk/docs/ $ pip install --upgrade pip
   (odkenv) /odk/docs/ $ pip install -r requirements.txt
   ```

   PowerShell

   ```
   (odkenv) /odk/docs/ > pip install --upgrade pip
   (odkenv) /odk/docs/ > pip install -r requirements.txt
   ```

   The first command upgrades pip itself to the latest version. Then second checks requirements.txt and installs everything listed in it. This will take several moments.

   **Note:** If you are ever running one of the build commands shown below and it fails with a message that includes `ModuleNotFoundError`, there might be changes to requirements.txt since you originally ran `pip install -r requirement.txt`. Run the installation again and then retry your build.

11. Choose a text/code editor

   The documentation source files are written in a plain text format called reStructuredText. This means special formatting (bullets, headers, bold text) is represented by visible characters, not hidden behind a graphical display. When working on a documentation file, you see and write something that looks like:

   ```
   #. Choose a text/code editor

   The documentation source files
   are written in a plain text format called `reStructuredText`.

   .. _reStructuredText: http://docutils.sourceforge.net/docs/user/rst/quickref.html
   ```

   You cannot write and edit these files in a typical document preparation program like MS Word or Google Docs. Instead, you need a coding editor.
There are a lot of editors, and people who use them often have very strong opinions about them. You are free to choose any editor you like.

If you’ve never used an editor before, you might want to start with one of the easier and more popular ones:

- Atom
- Sublime
- VS Code
- Notepad++ (Windows only)

Most of these have plugins that will make writing reStructuredText easier by color-coding the markup.

This completes the setup of your local working environment. Take a break before diving into how you actually work.

23.1.3 Working on the docs

1. Find an issue to work on.

Work on ODK Docs is planned using the GitHub repository’s issue tracker.

(a) Browse the issue tracker and find one you may want to work on.

(b) Make sure you understand the goal of the project. If the goal isn’t clear, ask. If there is anything in the issue that doesn’t make sense, ask about it. Feel free to make suggestions about how something could be accomplished.

(c) If you decide to work on an issue, assign yourself to it by writing `@getodk-bot claim` in a comment.

(d) If the issue requires a novel or creative solution not defined in the issue already (we’ve stated a problem and you think you know a way to fix it) write a comment describing your plan. It is a good idea to get feedback on an idea before working on it. Often, other contributors can provide additional context about why a particular solution may or may not work.

Your first issue

The very first issue you should work on as a new ODK Docs contributor is Issue 96 — Line Edits. The issue is very simple:

(a) Find a typo.

(b) Fix the typo.
This will help you get used to working with the documentation tools, and helps us get rid of the inevitable errors that creep in to our writing.

2. Make sure you are on the master branch

A branch is a named sequence of changes representing work on the repo. For example, if you were going to work on Issue 96 — Line Edits, you would create a new branch called line-edits to hold that work. When you were done, you would merge those changes back to the main branch, which we call master.

The first time you clone the docs repo and start working, you will be on the master branch.

Each time you come back to starting work on a new issue, make sure you are on the master branch before continuing.

(a) Check the current branch with `git branch`. This will output a list of branches, with a star next to the current one.

Bash

```
$ git branch
```

<table>
<thead>
<tr>
<th>branch-name</th>
<th>branch-name</th>
<th>branch-name</th>
<th>* master</th>
<th>branch-name</th>
</tr>
</thead>
</table>

PowerShell

```
(odkenv) /odk/docs/ > git branch
```

<table>
<thead>
<tr>
<th>branch-name</th>
<th>branch-name</th>
<th>branch-name</th>
<th>* master</th>
<th>branch-name</th>
</tr>
</thead>
</table>

(b) If you are not on master, switch to master with `git checkout`.

Bash

```
$ git checkout master
```

Switched to branch 'master'

Your branch is up to date with 'origin/master'.

Our documentation is updated frequently. Get the latest version at [https://docs.getodk.org](https://docs.getodk.org).
3. Pull in changes from upstream

Other people are constantly making changes to the docs, so you need to keep your local copy up to date.

Before you start working, use `git pull` to pull in the changes from the upstream repository’s master branch. Then, just to be sure, you can use `git status` to make sure everything is up to date.

**Bash**

```bash
(odkenv) /odk/docs/ $ git pull upstream master
```

**PowerShell**

```powershell
(odkenv) /odk/docs/ $ git pull upstream master
```

```powershell
(odkenv) /odk/docs/ $ git status
```

```
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

**Warning:** Some git commands (including `git pull` and `git checkout`) send error messages to PowerShell even when they work correctly. If everything seems to be working, you can ignore these.

4. Create a new branch for your work.

**Bash**

```bash
(odkenv) /odk/docs/ $ git checkout -b branch-name
```

**PowerShell**

```powershell
(odkenv) /odk/docs/ > git checkout -b branch-name
```

```
Switched to branch 'branch-name'
```

```
Switched to a new branch 'branch-name'
```
Branch names should be short, lowercase, and use hyphens as separators. They do not need to carry a lot of information (like your name or the date).

Good branch names:
- getting-started-guide
- contributing
- fix-issue-13

Bad branch names:
- getting started guide
- Getting started guide
- Getting_started_guide
- writing-the-getting-started-guide-adammichaelwood-july-2017-draft

5. Work on the documentation

Finally, you can open an editor of your choice and work on the documentation.

The source files for documentation text are in these directories:

src Files for the pages at https://docs.getodk.org

If you’re going to write or edit documentation text, please read:
- Docs Markup and Syntax Guide
- Docs Style Guide

If you’re working on code or deployment, please read:
- Docs Developer Guide

6. Local checks

Once you have worked on the documentation, we want to make sure your contribution will get accepted and published right away.

To ensure your changes will pass all the deployment tests, you should run the tests locally first and correct any problems.

(a) Spell check

If you’ve been working on files in src:

Bash
PowerShell

```
(odkenv) /odk/docs/ $ make spell-check
```
This will send some output to the terminal, which will include mentions of any words not in the dictionary.

- If the flagged words are really misspellings, correct them.
- If the flagged words are not misspelled, and should be in the dictionary add them to `spelling_wordlist.txt`.
- If the flagged words are not misspelled, but should not be in the dictionary (for example, they are non-words that make sense on a single page for a specific reason) add them at the top of the file in which they are being used, before the title heading:

```
.. spelling::
  
  abc
def
  exe
  functool

This Is The Page Title
====================
```

When adding new words to `spelling_wordlist.txt` or the top of a document file, please keep the words in alphabetical order.

(b) Style check

7. Build and check

We use a Python tool called Sphinx to compile all the .rst files into a working website. If you’ve been working on files in src:

Bash

```
(odkenv) /odk/docs/ $ make build
```

PowerShell

```
(odkenv) /odk/docs/ > rm -r -fo tmp
(odkenv) /odk/docs/ > rm -r -fo build
(odkenv) /odk/docs/ > Copy-Item src -Destination tmp -Recurse
(odkenv) /odk/docs/ > sphinx-build -b dirhtml tmp build
```

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
This generates a lot of output. Near the end of the output you may see a statement like:

```
build succeeded, 18 warnings.
```

Those warnings are problems with the text which you need to fix before submitting your changes. Scroll up in the terminal to find each warning, so that you can address it in the source files.

A Sphinx warning looks like this:

```
/path/to/file-name.rst:LINENUMBER: WARNING: warning message
short excerpt from the file
```

This tells you what file the problem is in, the approximate line number, and the nature of the problem. Usually that is enough to fix it. If you can not figure out the meaning of a particular warning, you can always ask about it on the ODK Slack.

**Note:** Because of a bug in Sphinx the line numbers in error and warning messages will be off by about 15 lines (the length of `rst_prolog` in `conf.py`).

As you fix each warning, run the build again to see if it disappears from the output.

**Note:** The warning messages will refer to the file name using the temporary directory path `tmp`. You need to correct the problems in the real source directory (`src`).

---

**When you just can’t fix the error...**

If you’ve done your best and asked on the ODK Slack, and you still cannot correct the warning, stop worrying about it and skip to the next step. When you submit your changes on GitHub, include a note about the warning. Other contributors will help solve the problem before merging.

Once you’ve corrected all the warnings that can be corrected...

8. Serve the documentation website locally and view it.

If you’ve been working on files in `src`:

- Bash
- PowerShell
(a) Open your browser and go to http://localhost:8000.
(b) Read through your doc edits in the browser.
(c) Go back to the source files to correct any errors you find.
(d) Go to your terminal, and press CTRL C to shut down the local web server.
(e) Re-run the build and serve steps.
(f) Continue proofreading.

Once you are reasonably sure your changes are ready...

9. Commit your changes to your local repository.

A commit is snapshot of your working files in a particular state, along with a record of all the changes that led up to that state. That snapshot is what you will submit to the main repository.

**Note:** We explain how to do a commit at this step because you need to do it before you can submit your changes. However, you don’t have to wait until you are done to commit. You can commit as many times as you like while working.

This can be especially helpful if you are working on a complicated set of changes, over several working sessions.

(a) Stage the files for commit with **git add**.

   To stage all changes for commit:

   **Bash**

   ```bash
   (odkenv) /odk/docs/ $ git add -A
   ```

   **PowerShell**

   ```powershell
   (odkenv) /odk/docs/ > git add -A
   ```

(b) Commit the staged files with **git commit**.

   **Bash**

   ```bash
   ```

   **PowerShell**

   ```powershell
   ```
Your commit message needs to be wrapped in quote marks. It should, in a sentence or less, explain your work.

10. Push your committed changes to your GitHub repo with `git push`.

   Bash
   ```bash
   (odkenv) /odk/docs/ $ git push origin branch-name
   ```

   PowerShell
   ```powershell
   (odkenv) /odk/docs/ > git push origin branch-name
   ```

   **Warning:** The `git push` command produces meaningless errors in PowerShell. If you get an error when using `git push`, but everything seems to work otherwise, ignore the error.

   **Tip:** You may be prompted to enter your GitHub username and password. When entering your password, the cursor won’t move — it will look like you aren’t entering anything, even though you are.

   To avoid having to retype these every time, you can store your GitHub credentials locally.

11. Issue a pull request from your GitHub repo to the main ODK Docs repo.

    A *pull request* (or PR) is a request from you to the ODK Docs maintainers to pull in your changes to the main repo.

    (a) Go to the ODK Docs repo on GitHub. (Make sure you are logged in.)

    (b) Find the message near the top of the page that mentions your recent pushed branches. Select *Compare & pull request* to start a pull request.

    (c) Follow GitHub’s instructions to start the pull request.

    These details should fill-in automatically, but be sure to double-check them:

    - *Base fork* should be the main repo (`getodk/docs`).
    - *base* should be `master`.
    - Your repo and working branch name should be listed beside them.
You will see either a green **Able to be merged** message or a message informing that the branch can not be merged. You can proceed in either case. If the branch cannot be merged, the maintainers will work with you to resolve the problem.

(d) **Write a PR message explaining your work.**

The PR message field includes a template to remind you of what to include. Fill in the template and delete any sections which are not applicable.

A good PR message includes:

- The issue number you are working on. (Write `closes #123` if the PR completes the work for the issue. If there’s still work to do, write `addresses #123`.)
- A summary of what you did.
- Details of work that still needs to be done.
- Details of new work created or implied by this PR.
- Details of any unresolved errors or warnings, including details of what you tried.
- Justification for any changes to `requirements.txt`.
- Details of any difficulties, questions, or concerns that came up while working on this issue.

(e) **Submit your pull request.**

The maintainers and other contributors will review your PR as quickly as possible. They may request changes to your work. If changes are needed:

(a) **Don’t worry.** Revision is a normal part of technical writing, and everyone (even the project’s founders and leaders) has their work reviewed and are frequently asked to revise it.

(b) Work on the files again locally. (Use `git branch` to make sure you are still in the same working branch.)

(c) **Stage and commit** your changes locally again (`git add -A; git commit -m "Commit message"`).

(d) **Push your commit** (`git push origin branch-name`).

(e) Your new commits will automatically update the PR. Do not start a new PR. Once everything has been approved, the changes will be merged in and will appear on this website. At that point... congratulations! You are now a contributor to ODK.
23.1.4 The next time you work

We hope that contributing to ODK Docs is a rewarding experience and that you’ll want to keep going. Each time you start work on a new issue the process is the same as outlined above.

Here are a few things to keep in mind when you start your next contribution.

1. Return to master with `git checkout master`.
   New work is done on new branches which are started from master. So, before you start a new branch, return to the master branch.
   
   Bash
   ```
   (odkenv) /odk/docs/ $ git checkout master
   ```
   PowerShell
   ```
   (odkenv) /odk/docs/ > git checkout master
   ```

2. Pull in changes with `git pull upstream master`.
   You need to start your new work from the latest version of everyone else’s work.
   
   Bash
   ```
   (odkenv) /odk/docs/ $ git pull upstream master
   ```
   PowerShell
   ```
   (odkenv) /odk/docs/ > git pull upstream master
   ```

3. Update the master branch of your online GitHub repository.
   
   Bash
   ```
   (odkenv) /odk/docs/ $ git push origin master
   ```
   PowerShell
   ```
   (odkenv) /odk/docs/ > git push origin master
   ```

4. Find a new issue to work on.

5. Start a new branch for your work with `git checkout -b branch-name`.
23.1.5 Keep improving

As you are getting comfortable with the contribution process, take a few minutes to read our Tips for Making Good Contributions. You may also want to dig deeper into the Docs Style Guide and the Docs Markup and Syntax Guide. (And if you are writing code, check out the Docs Developer Guide.)

And don’t forget to join us on the ODK Slack.

ODK is a community, and we depend on each other’s work. Thank you for your contribution to ODK Docs and your presence in this community.

23.2 Docs Markup and Syntax Guide

The ODK documentation is built using Sphinx, a static-site generator designed to create structured, semantic, and internally consistent documentation. Source documents are written in reStructuredText, a semantic, extensible markup syntax similar to Markdown.

- reStructuredText Primer — Introduction to reStructuredText
  - reStructuredText Quick Reference
  - reStructuredText 1-page cheat sheet
- Sphinx Markup — Detailed guide to Sphinx’s markup concepts and reStructuredText extensions

**Note:** Sphinx and reStructuredText can be very flexible. For the sake of consistency and maintainability, this guide is *highly opinionated* about how documentation source files are organized and marked up.

23.2.1 Indentation

Indentation is meaningful in Sphinx and reStructuredText text. Usually, indenting a section means that it “belongs to” the line it is indented under. For example:

```
.. figure:: path-to-image.*

  This is the caption of the figure. Notice that it is indented under the line defining the figure.
```

The rules for indentation are:

- Use *spaces*, not *tabs*. 
23.2. Docs Markup and Syntax Guide

- Generally, indent two spaces.

The exception to the two spaces rule is *Ordered (numbered) lists*, where indentation follows the content of the list item.

1. This is a list item.

   This is some additional content related to Item 1. Notice that it is indented to the same column as the first line of content. In this case, that's three (3) spaces.

10. The tenth item in a list.

   This related content will be indented four spaces.

### 23.2.2 Documentation Files

Sphinx document files have the `.rst` extension. File names should be all lowercase and use hyphens (not underscores or spaces) as word separators.

Normally, the title of the page should be the first line of the file, underlined with equal-signs.

```plaintext
Title of Page
==============
Page content is here...
```

You can alternatively wrap the title in two lines of asterisks, in some cases. (This should not be your default choice.)

```plaintext
***************
Title of Page
***************
Page content here.
```

The asterisks style is useful when you are combining several existing documents and don’t want to change every subsection headline. Or, you can use it when you are working on a document that you have reason to think might be split into separate documents in the future.

**Important:** If you use the double-asterisks style, your major section headlines (`<h2>`)
should use the equal-signs underline style. This allows major sections to be easily promoted to individual pages.

See *Sections and Titles* for more details.

### 23.2.3 Tables of Content

The `toctree` directive defines a table of content. The content of a `toctree` is a list of page file names, without the `.rst` extension. When rendered, the `toctree` becomes an unordered list of page links, including links to sections and subsections of the included pages.

```rst
.. toctree::

    page-name
    another-page
    this-other-page
```

The depth of section and subsection links to display in the output can be controlled using the `maxdepth` attribute. We typically use a depth of 2, but you should use your judgment if you feel it should be more or less in any given context.

```rst
.. toctree::

    :maxdepth: 2

    this-page
    that-page
    thick-page
    flat-page
```

See also:

- The TOC Tree

  The Sphinx documentation includes information about a number of other `toctree` attributes.

**Sidebar navigation menu**

The `index.rst` file serves as a front-page to the documentation and contains the main tables of content, defined using `toctree` directives.

These `toctree` directives control the sidebar navigation menu. To add a new document to a table of content, add the file name (without the `.rst` extension) to the relevant list of file names in `index.rst`. 
Secondary tables of content

Collections of documents are sometimes given their own table of content on an individual page. (See, for example, Setting Up ODK Collect and Using ODK Collect.)

In these cases, the page containing the toctree serves as a sort of intro page for the collection. That intro must, itself, be included in the Sidebar navigation menu.

The contents of a toctree appear as section links in another toctree it is included in. That is, if a toctree in index.rst lists collect-using, and collect-using.rst has a toctree, then the contents of that second toctree will appear in the Sidebar navigation menu, as sub-items to Using ODK Collect. (Indeed, this is precisely the case in the docs currently.)

How ODK Docs uses main and secondary tables of content

- Major topics get a toctree in index.rst
  
  Major topics include things like:
  
  - Each major product (Collect, Aggregate, Briefcase)
  
  - Large, general categories like Contributing

  Major topic tables of content include both sub-collection intro pages and also individual pages that don’t fit into a sub-collection.

  The caption attribute of the toctree directive defines the section label in the Sidebar navigation menu.

- Within a large topic, documents are grouped into collections of related pages, defined by a toctree on a topic intro page.

  Intro pages (pages that contain secondary toctree directives) may include additional content, introducing the collection or providing contextual wayfinding. However, this is not always necessary or desirable. Use your judgment, and avoid stating things just for the sake of having some text. ("Here are the pages in this collection.")

  We also (very occasionally) include toctree directives in sub-collection pages.

Tip: If it not obvious where a new document should appear in the navigation, the best practice is to simply ask about it in the GitHub issue driving the new page.

Note: For wayfinding purposes, we sometimes create an Unordered (bullet) lists of page links rather than a toctree directive. (For example, see collect-intro. We do this when using a toctree would create redundant links in the Sidebar navigation menu.)
Why are the docs files not grouped into folders in the source?

We use `toctree` directives as our primary way of organizing the documentation for readers. We do not organize the source `rst` files into subfolders.

The reason is that if we put them into topic-related subfolders, it would affect the URI of the document. Keeping all of our document files in a single flat directory results in a flat URI structure. Every page’s URI looks like `docs.getodk.org/page-name`.

If we used subdirectories, then our URIs would look like `docs.getodk.org/subdirectory-name/page-name`. This would mean that our URIs would change every time we moved a document from one folder to another, greatly increasing the time cost and broken-link risk of reorganizing the docs.

### 23.2.4 Sections and Titles

Headlines require two lines:

- the text of the headline, followed by
- a line filled with a single character.

Each level in a headline hierarchy uses a different character:

<table>
<thead>
<tr>
<th>Title of the Page - &lt;h1&gt; - Equal Signs</th>
</tr>
</thead>
<tbody>
<tr>
<td>------------------------------------------</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Major Section - &lt;h2&gt; - Hyphens</th>
</tr>
</thead>
<tbody>
<tr>
<td>------------------------------------------</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subsection - &lt;h3&gt; - Tildes</th>
</tr>
</thead>
<tbody>
<tr>
<td>------------------------------------------</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sub-subsection - &lt;h4&gt; - Double Quotes</th>
</tr>
</thead>
<tbody>
<tr>
<td>------------------------------------------</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sub-sub-subsection - &lt;h5&gt; - Single Quotes</th>
</tr>
</thead>
<tbody>
<tr>
<td>------------------------------------------</td>
</tr>
</tbody>
</table>

If you need to combine several existing pages together, or want to start a single-page doc that you think might be split into individual pages later on, you can add a top-level title, demoting the other headline types by one.
In either case, the underline of characters needs to be longer than the line of text. In the case of the asterisks, the two lines of asterisks need to be the same length.

**Note:** The exact order of underline characters is flexible in reStructuredText. However, this specific ordering should be used throughout the ODK documentation.

### Section labels

In order to facilitate efficient *Cross referencing*, sections should be labeled. This is done on the line above the section title. The format is:

- two dots
- underscore
- section label
  - lowercase
  - hyphen separators
- a single colon
The section label is usually a slugified version of the section title. Section titles must be unique throughout the entire documentation set. Therefore, if you write a common title that might appear in more than one document (Learn More or Getting Started, for example), you’ll need to include additional words to make the label unique. The best way to do this is to add a meaningful work from the document title.

**ODK Aggregate**

ODK Aggregate is a server application...

**23.2.5 Basic Markup**

**Escaping characters**

Markup characters can be escaped using the \ character.

*Italic.*

\*Not italic, surrounded by asterisks.\*

*Italic.*

*Not italic, surrounded by asterisks.*

**Emphasis and Inline Literal**

Single asterisks for *italic text* (`\<em>`).

Double asterisks for **bold text** (`\<strong>`).
Double back-ticks for ``inline literal text`` (``<code>``).

Single asterisks for italic text ( `<em>`).

Double asterisks for bold text ( `<strong>`).

Double back-ticks for inline literal text ( `<code>`).

Note: The bold, italic, and inline literal styles do not carry semantic meaning. They should not be used when a more semantically appropriate markup construct is available; for example, when *writing about GUI text*.

Hyperlinks

External hyperlinks — that is, links to resources outside the documentation — look like this:

This is a link to `example <http://example.com>`_.

This is a link to example.

You can also use "reference style" links:

This is a link to `example`_.

.. _example: http://example.com

This may help make paragraphs with a lot of links more readable. In general, the inline style is preferable. If you use the reference style, be sure to keep the link references below the paragraph where they appear.

You can also simply place an unadorned URI in the text: http://example.com

You can also simply place an unadorned URI in the text: http://example.com

Lists

Unordered (bullet) lists

Bulleted lists ( `<ul>`):

- use hyphens
Chapter 23. Contributing to ODK Docs

- are unindented at the first level
- must have a blank line before and after

- the blank line requirement means that nested list items will have a blank line before and after as well

- you may *optionally* put a blank line *between* list items

Bulleted lists (<ul>):

- use hyphens
- are unindented at the first level
- must have a blank line before and after
  - the blank line requirement means that nested list items will have a blank line before and after as well
  - you may *optionally* put a blank line between list items

Ordered (numbered) lists

Numbered lists (<ol>):

1. Start each line with a number and period
2. Can begin on any number
3. Must have a blank line before and after
4. Can have nested sub-lists
   a. nested lists are numbered separately
   b. nested lists need a blank line before and after
5. Can have automatic number with the `#` character.

Numbered lists (<ol>):

1. Start each line with a number and period
2. Can begin on any number
3. Must have a blank line before and after
4. Can have nested sub-lists
   a) nested lists are numbered separately
   b) nested lists need a blank line before and after
5. Can have an automatic number with the # character.
23.2. Docs Markup and Syntax Guide

Note: See Ordered and unordered lists in the Docs Style Guide for details on when to use ordered and unordered lists.

Definition Lists

Definition list (``<dl>``) a list with several term-definition pairs

Terms
should not be indented

Definitions
should be indented under the term

Line spacing
there should be a blank line between term-definition pairs

Definition list (<dl>) a list with several term-definition pairs

Terms should not be indented

Definitions should be indented under the term

Line spacing there should be a blank line between term-definition pairs

Paragraph-level Markup

Paragraphs are separated by blank lines. Line breaks in the source code do not create line breaks in the output.

This means that you *could*, in theory, include a lot of arbitrary line breaks in your source document files. These line breaks would not appear in the output. Some people like to do this because they have been trained to not exceed 80 column lines, and they like to write .txt files this way. Please do not do this.

There is **no reason** to put a limit on line length in source files for documentation, since this is prose and not code. Therefore, please do not put arbitrary line breaks in your files.
Paragraphs are separated by blank lines. Line breaks in the source code do not create line breaks in the output.

This means that you could, in theory, include a lot of arbitrary line breaks in your source document files. These line breaks would not appear in the output. Some people like to do this because they have been trained to not exceed 80 column lines, and they like to write .txt files this way. Please do not do this.

There is no reason to put a limit on line length in source files for documentation, since this is prose and not code. Therefore, please do not put arbitrary line breaks in your files.

**Block Quotes**

This is not a block quote. Block quotes are indented, and otherwise unadorned.

This is a block quote.

— Adam Michael Wood

This is not a block quote. Block quotes are indented, and otherwise unadorned.

This is a block quote. — Adam Michael Wood

**Line Blocks**

| Line blocks are useful for addresses, |
| verse, and adornment-free lists. |
| Each new line begins with a |
| vertical bar ("|"). |
| Line breaks and initial indents |
| are preserved. |

Line blocks are useful for addresses, verse, and adornment-free lists.

Each new line begins with a vertical bar ("|").

Line breaks and initial indents are preserved.
# 23.2. Docs Markup and Syntax Guide

## Tables

### Grid style

<table>
<thead>
<tr>
<th>Header 1</th>
<th>Header 2</th>
<th>Header 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>+------------+------------+-----------</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>body row 1</th>
<th>column 2</th>
<th>column 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>+------------+------------+-----------</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>body row 2</th>
<th>Cells may span columns.</th>
</tr>
</thead>
<tbody>
<tr>
<td>+------------+--------------------------</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>body row 3</th>
<th>Cells may span rows.</th>
<th>Cells contain blocks.</th>
</tr>
</thead>
<tbody>
<tr>
<td>+------------+----------------------+-----------------------</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>body row 4</th>
</tr>
</thead>
</table>

| +------------+--------------------------|

### Simple style

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>-------</td>
<td>--------</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A or B</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
</tbody>
</table>

---

Our documentation is updated frequently. Get the latest version at [https://docs.getodk.org](https://docs.getodk.org).
CSV Table

The csv-table role is used to create a table from CSV (comma-separated values) data. CSV is a common data format generated by spreadsheet applications and commercial databases. The data may be internal (an integral part of the document) or external (a separate file).

```plaintext
.. csv-table:: Example Table
   :header: "Treat", "Quantity", "Description"
   :widths: 15, 10, 30

   "Albatross", 2.99, "On a stick!"
   "Crunchy Frog", 1.49, "If we took the bones out, it wouldn't be crunchy, now would it?"
   "Gannet Ripple", 1.99, "On a stick!"
```

Table 23.1: Example Table

<table>
<thead>
<tr>
<th>Treat</th>
<th>Quantity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Albatross</td>
<td>2.99</td>
<td>On a stick!</td>
</tr>
<tr>
<td>Crunchy Frog</td>
<td>1.49</td>
<td>If we took the bones out, it wouldn't be crunchy, now would it?</td>
</tr>
<tr>
<td>Gannet Ripple</td>
<td>1.99</td>
<td>On a stick!</td>
</tr>
</tbody>
</table>

Some of the options recognized are:

**:widths:**
Contains a comma or space-separated list of relative column widths. The default is equal-width columns.

The special value auto may be used by writers to decide whether to delegate the determination of column widths to the backend.

In most cases, the best result is either the default or auto. If you’re unsure, try it both ways and see which looks better to you.

**:header:**
Contains column titles. It must use the same CSV format as the main CSV data.

**:delim:**
Contains a one character string used to separate fields. Default value is comma. It must be a single character or Unicode code.

The only reason to use something other than a comma is when copying large blocks of content from another source that uses a different style. If you are creating new table content yourself, use the comma.

```plaintext
.. csv-table:: Table using # as delimiter
   :header: "Name", "Grade"
```

---

526 Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
23.2. Docs Markup and Syntax Guide

```
:widths: auto
:delim: #

"Peter"#"A"
"Paul"#"B"

.. csv-table:: Table using | as delimiter
:header: "Name", "Grade"
:widths: auto
:delim: |

"Peter"|"A"
"Paul"|"B"
```

**align:**
It specifies the horizontal alignment of the table. It can be *left*, *right* or *center*.

```
.. csv-table:: Table aligned to right
:header: "Name", "Grade"
:align: right

"Peter", "A"
"Paul", "B"
```

Table 23.2: Table aligned to right

<table>
<thead>
<tr>
<th>Name</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peter</td>
<td>A</td>
</tr>
<tr>
<td>Paul</td>
<td>B</td>
</tr>
</tbody>
</table>

**file:**
Contains the local file system path to a CSV data file.

**url:**
Contains an Internet URL reference to a CSV data file.

**Note:**
- There is no support for checking that the number of columns in each row is the same. However, this directive supports CSV generators that do not insert "empty" entries at the end of short rows, by automatically adding empty entries.

```
.. csv-table:: Table with different number of columns in each row
:header: "Name", "Grade"

"Peter"
```

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
Table 23.3: Table with different number of columns in each row

<table>
<thead>
<tr>
<th>Name</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peter</td>
<td></td>
</tr>
<tr>
<td>Paul</td>
<td>B</td>
</tr>
</tbody>
</table>

- Whitespace delimiters are supported only for external CSV files.

For more details, refer to this guide on CSV Tables.

**Note:** In almost all cases, `csv-table` is the easiest and most maintainable way to insert a table into a document. It should be preferred unless there is a compelling reason to use one of the other styles.

### 23.2.6 Sphinx-specific Markup

#### Roles and directives

A *role* is an inline markup construct that wraps some text, similar to an HTML or XML tag. They look like this:

```
:rolename:`some text`
```

A directive is a block-level markup construct. They look like this:

```
.. directivename:: additional info or options here
   :option: optional-value
   :option: optional-value

   Content of block here, indented.
```

This *is* no longer part of the block controlled by the directive.

Most of the Sphinx-specific and ODK-specific markup will use one or both of these constructs.

#### Cross referencing

Cross referencing is linking internally, from one place in the documentation to another. This is **not** done using the *Hyperlinks* syntax, but with one of the several roles:
```markdown
:role:`target`
becomes...
<a href="target">reference title</a>

:role:`anchor text <target>`
becomes...
<a href="target">anchor text</a>

:doc:
- Links to documents (pages)
- `target` is the file name, without the `.rst` extension
- `title` is the first `headline` (<h1>) of the page

:ref:
- Links to sections
- `target` is the `Section labels`
- `title` is the `section title (headline)`

:term:
- Links to items in the `Glossary`
- `target` is the term, in the glossary
- `title` is the term itself

To recap: If you do not include an explicit `target`, the text inside the role will be understood as the target, and the anchor text for the link in the output will be the title of the target.

For example:

- Link to this document:
  - :doc:`contributing`
  - :doc:`anchor text <contributing>`

- Link to this section:
  - :ref:`cross-referencing`
  - :ref:`anchor text <cross-referencing>`

- Link to a term:
  - :term:`participant`
  - :term:`anchor text <participant>`
```

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
Writing about User Interface

Several roles are used when describing user interactions.

:guilabel:

Marks up actual UI text of form labels or buttons.

Press the :guilabel:`Submit` button.

:menuselection:

Marks up the actual UI text of a navigation menu or form select element.

Select :menuselection:`Help` from menu.

When writing about multi-level menus, use a single :menuselection: role, and separate menu choices with -->.

To save your file, go to :menuselection:`File --> Save` in the Main Menu.

Note: In some situations you might not be clear about which option (:menuselection: or :guilabel:) to use. GUIs in real life can sometimes be ambiguous. The general rule is:

- Actual UI text will always receive :guilabel: role unless the text could reasonably be understood to be part of a menu.
- If the actual UI text could be understood as a menu, :menuselection: should be used.

These both render the same on output, so don’t worry too much if you get it wrong. Just use your judgment and take your best guess.
23.2. Docs Markup and Syntax Guide

:kbd:
Marks up a sequence of literal keyboard strokes.

To stop the local server, type `CTRL C`.

:command:
Marks up a terminal command.

To build the documentation, use `sphinx-build`.

:option:
Marks up a terminal command option.

The `-b html` option specifies the HTML builder.

:gesture:
Describes a touch screen gesture.

:gesture:`Swipe Left`

Writing about forms

We have added several custom text roles for writing about forms and the XForms and XLSForm formats.

:th:
Used to refer to a table header cell.

:tc:
Used to refer to a table cell.

External App String Widget

The external app widget is displayed when the `appearance` attribute begins with `ex:`.

:formstate:
Specifies the state of the form in ODK Collect, which could be one of the following:

- Blank
- Finalized
- Saved
- Sent
- Deleted

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
Other Semantic Markup

**:abbr:**
Marks up an abbreviation. If the role content contains a parenthesized explanation, it will be treated specially: it will be shown in a tool-tip in HTML.

```latex
:abbr:`ODK`
```

**:dfn:**
Marks the defining instance of a term outside the glossary.

```latex
:dfn:`ODK` is a suite of open source applications that help organizations\[\[\]
engaged in enumerator-mediated data collection.
```

**:file:**
Marks the name of a file or directory. Within the contents, you can use curly braces to indicate a "variable" part.

```latex
is installed in :file:`/usr/lib/python2.{x}/site-packages`
```

In the built documentation, the x will be displayed differently to indicate that it is variable.

**:program:**
Marks the name of an executable program.

```latex
launch the :program:`ODK Aggregate Installer`
```

Images and Figures

**PNGs only**

All still images used in ODK Docs should be PNG files. This helps us keep our image compression tooling simple, and generally results in higher-quality screenshots.

Whenever possible, you should generate your images as PNGs rather than converting to PNGs from another format. If you have to start in another format, use lossless formats whenever possible. These include BMP, GIF, and TIFF. (Avoid JPG/JPEG if possible, as this is a lossy format that does not replicate screenshots very well.)
Where to put image files

Image files should be put in the /src/img/ directory in the source, and they should be in a subdirectory with the same name as the document in which they appear. (That is, the filename without the .rst extension.)

Image compression

Before committing images locally, run lossless compression on them using one of the following tools:

- ImageOptim
- Pngout

Inserting images in a document

To place an image in a document, use the image directive.

```
.. image:: /img/{document-subdirectory}/{file}.*
  :alt: Alt text. Every image should have descriptive alt text.
```

Note the literal asterisk (*) at the end, in place of a file extension. Use the asterisk, and omit the file extension.

Inserting images with captions (figures)

Use figure to markup an image with a caption.

```
.. figure:: /img/{document-subdirectory}/{file}.*
  :alt: Alt text. Every image should have descriptive alt text.

  The rest of the indented content will be the caption. This can be a short sentence or several paragraphs. Captions can contain any other rst markup.
```

Inline images

To information on creating inline images, see Substitutions.
Image File Names

Image file names should:

- be short yet descriptive
- contain only lower case characters and (in Sequentially numbered images only) numbers
- have no spaces
- use hyphens as the separator

Good image file names:

- collect-home-screen.png
- build-data-export-menu.png

Bad image file names:

- Collect home screen.png
- collect_home_screen.png
- 3987948p2983768ohl84692p094.jpg-large

Sequentially numbered images

In the case of sequentially numbered images, the numbers should:

- be zero-indexed
- have two digits with leading zeroes
- be separated from the rest of the file name with a hyphen
- be placed at the end of the file name

Good sequentially numbered image file names:

- map-widget-00, map-widget-01, map-widget-02

Bad sequentially numbered image file names:

- 1-map-widget, 2-map-widget
- map-widget_00, map-widget_01
- map-widget-1, map-widget-2
23.2. Docs Markup and Syntax Guide

Screenshots from ODK Collect

If you have set up *Android Debug Bridge*, you can connect your Android device to your computer and take screenshots from the command line.

- Connect your device via USB
- Enable Developer Settings
  - *Settings* → *About phone*
  - Tap *Build number* seven (7) times
- Turn on USB Debugging
  - *Settings* → *Developer options* → *USB debugging*

Now, at the command line, from the root directory of the *odk-docs* repo:

```bash
python ss.py {document-name}/{image-name}
```

- `{document-name}` is the filename (without extension) where the image will be used.
- `{image-name}` is the name (without extension) given to the image. - follow the *Image File Names* guidelines

**Warning:** Make sure you do not overwrite an existing image.

**Tip:** If you have a problem running ss.py, check to make sure your Python 3 virtual environment is activated.

**Tip:** Be sure to obscure any personally-identifiable information from screen shots. Crop to the smallest relevant screen area. Annotate screen shots with arrows or circles to indicate relevant information.

Videos

Video files should be put in the */src/vid/* directory in the source, and they should be in a subdirectory with the same name as the document in which they appear. (That is, the filename without the *.rst* extension.)

The purpose of on page videos is to illustrate complicated user interactions that might be difficult to describe otherwise. Longer tutorial videos should be hosted elsewhere and, if appropriate, linked to from the docs. Therefore:
• The length of the videos must be less than a minute.
• Videos should have no audio.

To insert a video, use the custom `video` directive.

```rst
.. video:: path/to/video
   Specify the source path of the video and a descriptive alt content in the video directive. Alternate content is displayed when the video cannot be played. It can contain long texts as well as any other rst content.

.. video:: /vid/{document-subdirectory}/{file}.ext
   Alt content. Every video should have descriptive alt content.
```

The following optional attributes are supported:

**:autoplay:**
   Specifies whether the video should start playing as soon as it is ready. Can take boolean value: true, false, yes or no. Default is **no**.

   It is almost never a good idea to turn `autoplay` on.

:**controls:**
   Specifies whether the video controls should be displayed. Can take boolean value: true, false, yes or no. Default is **yes**.

:**muted:**
   Specifies whether the audio output of the video should be muted. Can take boolean value: true, false, yes or no. Default is **yes**.

:**loop:**
   Specifies whether the video should start over again, every time it is finished. Can take boolean value: true, false, yes or no. Default is **no**.

:**preload:**
   Specifies if and how the author thinks the video should be loaded when the page loads. Can take one of the following three values: `auto`, `metadata` or `none`.

:**poster:**
   Contains the source address for an image to be shown while the video is downloading, or until the user hits the play button.

---

**Note:** Images to be used as poster for a video should be in the same directory as the video and should have a filename like `[same-file-name-as-video]-poster.ext`.

:**class:**
   Specifies a class for the video element.
For more details on these attributes, see this guide.

To add a video in a document with the above options, you can do the following:

```
.. video:: /vid/{document-subdirectory}/{file}.ext
   :autoplay: yes/no
   :controls: yes/no
   :muted: yes/no
   :loop: yes/no
   :class: class-name
   :preload: auto/metadata/none
   :poster:: /vid/{document-subdirectory}/{file}.ext
```

Alt content. Every video should have descriptive alt content.

---

### Capturing video from Android

Android Debug Bridge (ADB) can be used to capture a screen recording from an Android app.

```
$ adb shell screenrecord /sdcard/example.mp4
```

On pressing the enter key the video recording starts. Recording stops automatically after 3 minutes. Since videos have to be less than a minute, press CTRL C to stop the recording.

The video file is saved in your Android device to a file at `/sdcard/example.mp4` file.

To pull the video locally:

```
$ adb pull /sdcard/example.mp4 local/path/to/save/to
```

### Downloadable files

Downloadable files should be put in the `/src/downloads/` directory in the source, and they should be in a subdirectory with the same name as the document in which they appear. (That is, the filename without the `.rst` extension.)

To place a downloadable file in a document, use the `download` role.

```
See this :download:`example script </downloads/contributing/example_script.py>` to understand the procedure better.
```
Code Samples

Use the code-block directive to insert code samples. Specify the language on the same line as the directive for syntax highlighting.

```rst
Use the `code-block` directive to markup code samples.
```

```python
print("Hello ODK!")
```

```console
$ python --version
```

```java
public class HelloWorld {
    public static void main(String[] args) {
        // Prints "Hello, World" to the terminal window.
        System.out.println("Hello, World");
    }
}
```

Note: rst code-blocks wrap overflow lines by default. To unwrap overflow lines, use unwrap class with rst code-blocks.

Code-blocks for other languages don’t wrap overflow lines. Instead of wrapping, you need to scroll side-ways. To wrap overflow lines with other code-blocks, use wrap class with them.

Substitutions

Substitutions are a useful way to define a value which is needed in many places.
Substitution definitions are indicated by an explicit markup start (".. ") followed by a vertical bar, the substitution text (which gets substituted), another vertical bar, whitespace, and the definition block.

A substitution definition block may contain inline-compatible directives such as image or replace. For more information, refer this guide.

You can define the value once like this:

```
.. |RST| replace:: reStructuredText
```

and then reuse it like this:

```
We use |RST| to write documentation source files.
```

Here, |RST| will be replaced by reStructuredText

You can also create a reference with styled text:

```
.. |slack| replace:: **ODK Slack**
.. slack: https://getodk.slack.com
```

You can use the hyperlink reference by appending a "_" at the end of the vertical bars, for example:

```
You can ask about your problem in |slack|_.
```

You can ask about your problem in ODK Slack.

The rst_epilog in conf.py contains a list of global substitutions that can be used from any file. The list is given below:

- If you want to create a hyperlink reference for ODK Slack, you can use |odk-slack|_.

```
You can use |odk-slack|_ to ask your questions.
```

You can use ODK Slack to ask your questions.

- To create a hyperlink reference for docs related issues, use |docs-issue|_.

```
If you find a problem, file an |docs-issue|_.
```

If you find a problem, file an issue.
Chapter 23. Contributing to ODK Docs

- To create a hyperlink reference for ODK Forum, use \texttt{|forum|_}.

  You can ask support questions in \texttt{|forum|_}.

  You can ask support questions in ODK Forum.

- To create a hyperlink reference for contributors guide, use \texttt{|contrib-guide|_}.

  Be sure to read the \texttt{|contrib-guide|_}.

  Be sure to read the contributors guide.

You can add inline images in the document using substitutions. The following block of code substitutes arrow in the text with the image specified.

```
The \texttt{|arrow|} icon opens the jump menu.

\ldots\texttt{|arrow| image:: /img/{document-subdirectory}/{file}.\*}
  :alt: Alt text.
```

23.3 Docs Style Guide

23.3.1 Spelling and grammar

American spelling and grammar

Whenever U.S. English and British (or other) English spelling or usage disagree, standard U.S. spelling and usage is preferred.

Wrong

The colour of the button is grey.

Right

The color of the button is gray.
@memoize
def check_ukus(text):
    """UK vs. US spelling usage."""
    err = "style-guide.uk-us"
    msg = "uk-vs-us-spell-check. '{}' is the preferred spelling."

    preferences = [
        ["gray", "grey"],
        ["color", "colour"],
        ["accessorizing", "accessorising"],
        ["acclimatization", "acclimatisation"],
        ["acclimatize", "acclimatise"],
        ["acclimatized", "acclimatised"],
    ]

    return preferred_forms_check(text, preferences, err, msg)

    # This is a sample code. The complete code can be found in the file:
    # proselint-extra.py.

Quote marks

- Quote marks should generally be avoided if possible.
- *Smart* quotes (also known as *curly quotes* or *directional quotes*) are not permitted in source files.

Avoid quote marks

Quote marks are used in prose writing to indicate verbatim text. This is rarely useful in technical writing, as verbatim text usually requires a more specific semantic markup.

Wrong

Click the button that says, "Save."

Right

Click `Save`. 

Our documentation is updated frequently. Get the latest version at [https://docs.getodk.org](https://docs.getodk.org).
Wrong

You may see an error message that says, "Something went wrong."

Right

You may get an error: ```Something went wrong.```

```python
def check_quotes(text):
    """Avoid using straight quotes.""
    err = "style-guide.check-quote"
    msg = "Avoid using quote marks."
    regex = r""""[a-zA-Z0-9 \{1,15}""

    errors = []

    for matchobj in re.finditer(regex, text):
        start = matchobj.start()+1
        end = matchobj.end()
        (row, col) = line_and_column(text, start)
        extent = matchobj.end()-matchobj.start()
        errors += [(err, msg, row, col, start, end,
                     extent, "warning", "None")]

    return errors
```

Straight quotes

Any time that you do need to use quotation marks, use straight (or plain) quotes. Sphinx and Docutils will output the typographically correct quote style.

```python
def check_curlyquotes(text):
    """Do not use curly quotes.""
    err = "style-guide.check-curlyquote"
    msg = "Do not use curly quotes. If needed use straight quotes."
    regex = r""""[a-zA-Z0-9 \{1,15}""

    errors = []

    for matchobj in re.finditer(regex, text):
        start = matchobj.start()+1
        end = matchobj.end()
        (row, col) = line_and_column(text, start)
```

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
23.3.  Docs Style Guide

```python
extent = matchobj.end() - matchobj.start()
errors += [(err, msg, row, col, start, end, extent, "warning", "None")]

return errors
```

Serial comma

In a comma-delineated list of items, the penultimate item should be followed by a comma.

Wrong

Apples, oranges and pears.

Right

Apples, oranges, and pears.

A bulleted list is often more clear than an inline list.

Correct

You will need to be familiar with git, GitHub, and Python.

Possibly Better

You will need to be familiar with:

- git
There’s no hard rule about which to use in any situation. Use your judgement: try it both ways and see which is more clear.

**Direct Address**

Direct address — speaking directly to the reader using the second person “you” — is preferred over passive voice (“it can be done”), first-person plural (“we can do it”), or other constructions.

First person plural (“we”) should only be used when speaking of the ODK project team (“We recommend...”).

**Ordered and unordered lists**

An ordered list is numbered. It should be used when the order of the list is essential. For example, when enumerating a series of steps in a procedure.

**Wrong**

- First we do this.
- And then we do this.
- And then we do this.

**Right**

1. Do this.
2. Do this.
3. Do this.

An unordered list is bulleted. It should be used for a collection of items in which order is not essential.

**Wrong**

1. apples
2. oranges
3. bananas
23.3. Docs Style Guide

Right

- apples
- oranges
- bananas

Avoid Latin

Several Latin abbreviations are common in written English:
At best, these present a minor barrier to understanding. This is often made worse by unintentional misuse.
Avoid Latin abbreviations.

Wrong

If you are writing about a specific process (e.g., installing an application)...

Right

If you are writing about a specific process (for example, installing an application)...

@memoize
def check_latin(text):
    """Avoid using Latin abbreviations.""
    err = "style-guide.latin-abbr"
    msg = "Avoid using Latin abbreviations like "etc\.", "i.e.\.""

    list = [
        "etc\.", "etc", "\*etc\.*", "\*etc\*",
        "i\.*\.", "ie", "\*ie\.*", "\*ie\*",
        "e\.*\.", "eg", "\*eg\.*", "\*eg\*",
        "viz\.", "viz", "\*viz\.*", "\*viz\*",
        "c\.*\.", "cf", "\*cf\.*", "\*cf\*",
        "n\.*\.", "nb", "\*nb\.*", "\*nb\*",
        "q\.*\.", "qv", "\*qv\.*", "\*qv\*",
        "ibid\.", "ibid", "\*ibid\.*", "\*ibid\*",
    ]

    return existence_check(text, list, err, msg, ignore_case=True)
Etc.

*Et cetera* (or *etc.*) deserves a special mention.

*Et cetera* means "and all the rest," and is often used to indicate that there is more that could or should be said, but which is being omitted.

Writers often use *etc.* to gloss over details of the subject which they are not fully aware of. If you find yourself tempted use *etc.*, ask yourself if you really understand the thing you are writing about.

Avoid unneeded words

Adverbs

Adverbs often contribute nothing. Common offenders include:

- simply
- easily
- just
- very
- really
- basically
- extremely
- actually

Wrong

To open the file, simply click the button.

Right

To open the file, click the button.
23.3. Docs Style Guide

Wrong

You can easily edit the form by...

Right

To edit the form...

```python
@memoize
def check_adverb(text):
    """Avoid using unneeded adverbs.""
    err = "style-guide.unneed-adverb"
    msg = "Avoid using unneeded adverbs like \"just\", \"simply\"."

    list = [
        "simply",
        "easily",
        "just",
        "very",
        "really",
        "basically",
        "extremely",
        "actually",
    ]

    return existence_check(text, list, err, msg, ignore_case=True)
```

Filler words and phrases

Many words and phrases provide no direct meaning. They are often inserted to make a sentence seem more formal, or to simulate a perceived style of business communication. These should be removed.

Common filler phrases and words include:

- to the extent that
- for all intents and purposes
- when all is said and done
- from the perspective of
- point in time
This list is not exhaustive. These "canned phrases" are pervasive in technical writing. Remove them whenever they occur.

```python
@memoize
def check_filler(text):
    """Avoid using filler phrases.""
    err = "style-guide.filler-phrase"
    msg = "Avoid using filler phrases like \"to the extent that\"."
    list = [
        "to the extent that",
        "when all is said and done",
        "from the perspective of",
        "point in time",
    ]
    return existence_check(text, list, err, msg, ignore_case=True)
```

**Semicolons**

Semicolons are used to separate two independent clauses which could stand as individual sentences but which the writer feels would benefit by close proximity.

Semicolons can almost always be replaced with periods (full stops). This rarely diminishes correctness and often improves readability.

**Correct**

These "canned phrases" are pervasive in technical writing; remove them whenever they occur.

**Better**

These "canned phrases" are pervasive in technical writing. Remove them whenever they occur.

```python
@memoize
def check_semicolon(text):
    """Avoid using semicolon.""
    err = "style-guide.check-semicolon"
    msg = "Avoid using semicolon."
    regex = ";"
Pronouns

Third-person personal pronouns

Third-person personal pronouns are:

- he/him/his
- she/her/her(s)
- they/them/their(s)

Note: While some people consider they/them/their to be non-standard (or "incorrect") as third-person singular, it has gained wide use as a gender-neutral or gender-ambiguous alternative to he or she.

There are two issues with personal pronouns:

- gender bias
- clarity

To avoid gender bias, the third person gender-neutral they/then/their(s) is preferred over he or she pronouns when writing about abstract individuals.

Wrong

The enumerator uses his device.

Right

The enumerator uses their device.

Unfortunately, they/them/their is not a perfect solution. Since it is conventionally used as a plural pronoun, it can cause confusion.

Therefore, avoid the use of personal pronouns whenever possible. They are often out of place in technical writing anyway. Rewriting passages to avoid personal pronouns often makes the writing more clear.
Chapter 23. Contributing to ODK Docs

Correct

When using Collect, first the enumerator opens the app on their device. Then they complete the survey.

Better

To use Collect:

- open the app
- complete the survey

@memoize
def check_pronoun(text):
    """Avoid using third-person personal pronouns."""
    err = "style-guide.personal-pronoun"
    msg = "Avoid using third-person personal pronouns like "he", "she". In case of absolute need, prefer using "they"."
    list = [
        "he",
        "him",
        "his",
        "she",
        "her",
        "hers",
    ]
    return existence_check(text, list, err, msg, ignore_case=True)

"Same"

_Same_, when used as an impersonal pronoun, is non-standard in Modern American English. It should be avoided.

Wrong

ODK Collect is an Android app. The same can be used for...
23.3. Docs Style Guide

Right

ODK Collect is an Android app. It can be used for...

Right

ODK Collect is an Android app that is used to...

```python
@memoize
def check_same(text):
    """Avoid using impersonal pronoun same."""
    err = "style-guide.check-same"
    msg = "Avoid using "The same"."
    regex = ". The same"

    return existence_check(text, [regex], err, msg, ignore_case=False,
                            require_padding=False)
```

Titles

Title case and sentence case

Document titles should be in Title Case – that is, all meaningful words are to be capitalized. Section titles should use Sentence case – that is, only the first word should be capitalized, along with any proper nouns or other words usually capitalized in a sentence.

Verb forms

If a document or section describes a procedure that someone might do, use a verb ending in -ing. (That is, a gerund.) Do not use the "How to..." construction.

Wrong

```markdown
How to install ODK Collect
--------------------------
```
Right

**Installing ODK Collect**
----------------------

If section title is a directive to do something (for example, as a step in a procedure), use an imperative.

**Installing ODK Aggregate**
------------------------

Download ODK Aggregate
~~~~~~~~~~~~~~~~~~~~~~

Section content here.

```python
@memoize
def check_howto(text):
    """Avoid using how to construct.""
    err = "style-guide.check-howto"
    msg = "Avoid using "How to" construction."
    regex = "(How to.*)\([=\-\-\"]*\)+"
    return existence_check(text, [regex], err, msg, require_padding=False)
```

Section labels

Section titles should almost always be preceded by labels.

The only exception is very short subsections that repeat — like the **Right** and **Wrong** titles in this document or the **XLSForm Rows** and **XForm XML** sections in the **Question Types** document.

In these cases, you may want to use the **rubric** directive.

```python
def check_label(text):
    """Prefer giving a section label.""
    err = "style-guide.check-label"
    msg = "Add a section label if required."
    regex = r"(.*\n\n\n[=\-\-\"]\{3,})"n
    errors = []
    sym_list = [',', '---', '---', '---', '---', '\\"]
    is_doc_title = True
```
for matchobj in re.finditer(regex, text):
    if is_doc_title:
        is_doc_title = False
        continue
    label = matchobj.group(1)
    start = matchobj.start()+1
    end = matchobj.end()
    (row, col) = line_and_column(text, start)
    row = row + 2
    if any(word in text.splitlines(True)[row] for word in sym_list):
        row = row - 1
    col = 0
    extent = matchobj.end()-matchobj.start()
    catches = tuple(re.finditer(r'\.. _', label))
    if not len(catches):
        errors += [(err, msg, row, col, start, end,
                    extent, "warning", "None")]

return errors

Other titling considerations

- Do not put step numbers in section titles.
- Readers skim. Section titles should be clear and provide information.

23.3.2 Writing code and writing about code

ODK Documentation includes code samples in a number of languages. Make sure to follow generally accepted coding style for each language.

Indenting

In code samples:

- Use spaces, not tabs.
- Two spaces for logical indents in most languages.
  - Python samples must use four spaces per indent level.
- Strive for clarity. Sometimes nonstandard indentation, especially when combined with non-syntactic line breaks, makes things easier to read.
– Make sure that line breaks and indentation stay within the valid syntax of the language.

Using two spaces keeps code sample lines shorter, which makes them easier to view.

**Example of indenting for clarity**

```
HTTP/1.0 401 Unauthorized
Server: HTTPd/0.9
Date: Sun, 10 Apr 2005 20:26:47 GMT
WWW-Authenticate: Digest realm="testrealm@host.com",
                   qop="auth,auth-int",
                   nonce="dcd98b7102dd2f0e8b11d0f600bf0c093",
                   opaque="5ccc069c403ebaf9f0171e9517f40e41"
Content-Type: text/html
Content-Length: 311
```

**Meaningful names**

When writing sample code, avoid meaningless names.

**Wrong**

```python
def myFunction(foo):
    for bar in foo:
        bar[foo] = foo[spam] + spam[foo]

return foobar
```

**XML and HTML**

Some of the terms often used to describe XML and HTML code structures are imprecise or confusing. For clarity, we restrict certain terms and uses.

Likewise, coding practices and styles for XML and HTML vary widely. For the sake of clarity and consistency, samples should follow the guidelines set forth here.

**Element**

The following piece of code represents an element:
Note: An element is **not** a block or a tag.

- *Tag* is defined below.
- *Block* has a specific meaning in HTML and XML templates, and should generally be avoided outside those contexts.

### Tag

A **tag** is the token that begins or ends an element.

```html
<element> <!-- The opening tag of this element. -->
    Some content.
</element> <!-- The closing tag. -->
```

The word `tag` has often been used to refer to the entire element. For clarity, we will avoid that here.

### Node

The word `node` is often used interchangeably with `element`.

For clarity, we make the following distinction:

- An HTML or XML document has *elements*, not *nodes*.
- A *node* is part of a ”live” DOM tree or other dynamic representation.
  - An XML or HTML element becomes an *element node* in a DOM tree.
  - There are also other types of nodes in a DOM tree.

### Attributes and values

An element may have attributes. Attributes have values. Values are wrapped in straight double-quotes.

```html
<element attribute="value">
    Content.
</element>
```
Other names for attributes, such as *variables* or *properties*, should be avoided.

**Element content**

The code between the opening and closing tags of an element is the content. Content can include other elements, which are called *child elements*.

```xml
<element>
  Content.
  <child-element>
    More content.
  </child-element>
</element>
```

When an element is empty, it can be called a *null element*.

```xml
<null-element attribute="value"/>
```

In XML, null element tags always self-close. This is not the case in HTML.

- HTML elements that are always null (for example, `<img>`) do not need to be self-closed.
- Empty HTML elements that normally accept content have a separate closing tag.

```html
<img src="awesome-picture.jpeg">
<script src="some-javascript.js"></script>
```

**Capitalization**

For all HTML samples, tag names and attribute names should be all *lowercase*. Newly-written XML examples should also be all *lowercase*.

XML examples that show actual code generated by tools in the ODK ecosystem should replicate that code exactly, regardless of its capitalization practice.

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
23.3. Docs Style Guide

23.3.3 ODK jargon

ODK and ODK Docs

Wrong

- Odk
- odk
- OpenDataKit
- the ODK
- ODK docs
- ODK documentation

Right

- ODK
- getodk
- ODK Docs
- ODK Documentation

Probably want to avoid...

- ODK Documentation

```python
@memoize
def check_odkspell(text):
    """ODK spelling usage."""
    err = "style-guide.spelling-odk"
    msg = "ODK spell check. '{}' is the preferred usage."

    preferences = [
        ["ODK", ["OpenDataKit"]],
        ["ODK", ["Odk"]],
        ["ODK", ["{0} odk"]],
        ["ODK Docs", ["ODK docs"]],
        ["ODK Documentation", ["ODK documentation"]]
    ]
```
ODK app and project names

ODK includes a number of components, including:

- Collect
- Aggregate
- Briefcase

These should always be capitalized.

The ODK prefix (as in, ODK Collect) should be used the first time a document mentions the app or project, or any other time it would be unclear.

A few projects should always use the ODK prefix:

- ODK XForm
- ODK Javarosa
- ODK Docs

```python
@memoize
def check_appspell(text):
    """ODK spelling usage.""
    err = "style-guide.spelling-odk"
    msg = "ODK spell check. '{}' is the preferred usage."

    preferences = [
        ["Aggregate", ["{0} aggregate"],
        ["Briefcase", ["{0} briefcase"]
    ]

    return preferred_forms_check(text, preferences, err, msg, ignore_case=False)
```

XForms and XLSForm

- **XForms** refers to XML-encoded forms.
- **XLSForm** refers to a spreadsheet format used to define forms.
23.3. Docs Style Guide

Wrong

- Xforms
- X-Forms
- xforms
- XFORMS
- XForm (no s, when referring to the specification)
- xlsform
- XLSForm
- Xlsform

Right

- XForms
- an Xform (when referring to a single form)
- XLSForm

```python
@memoize
def check_formspell(text):
    
    """ODK spelling usage.""
    err = "style-guide.spelling-odk"
    msg = "ODK spell check. '{}' is the preferred usage."

    preferences = [
        ["XForms", ["Xforms"]],
        ["XForms", ["X-Forms"]],
        ["XForms", ["{0} xforms"]],
        ["XForms", ["XFORMS"]],
        ["an XForm", ["a XForm"]],
        ["an XLSForm", ["a XLSForm"]],
        ["XLSForm", ["{0} xlsform"]],
        ["XLSForm", ["XLSform"]],
        ["XLSForm", ["Xlsform"]]
    ]

    return preferred_forms_check(text, preferences, err, msg, ignore_case=False)
```
Chapter 23. Contributing to ODK Docs

XForms Spec, XForms Tools, XForms

*XForms* can refer to:

- The XML-based form format
- The official XForms specification from the W3C
- The ODK XForms Specification, which is a subset of the full W3C recommendation.
- The general idea of an XML-based form.

*XForm* (without an s) refers to:

- A specific XML document that encodes a form.

When writing about any of these things, make sure you are clear — in your mind as well as in your writing — which one you are talking about.

XLSForm

*XLSForm* can refer to:

- The XLSForm format for describing form in an Excel spreadsheet
- A spreadsheet file that describes a form using the format.
- A *tool* for converting *.xls(x) files to XForm documents.

When writing about any of these things, make sure you are clear — in your mind as well as in your writing — which one you are talking about.

23.4 Docs Developer Guide

This document is for contributors working on the design, templating, deployment, or development of the ODK Docs website.

23.4.1 Tech Overview

ODK Docs uses:

- *Sphinx*, a static-site generator written in *Python*

  Sphinx uses:
  
  - *Docutils* for parsing *reStructuredText*
  
  - *Jinja* for templating
23.4. Docs Developer Guide

- `sphinx_rtd_theme`, a Sphinx theme/template
  `sphinx_rtd_theme` uses:
  - `JQuery`, a JavaScript library
- `Proselint` for style testing
- `git` and `GitHub` for version control
- `CircleCI` for testing and deployment
- `Amazon S3` for hosting

23.4.2 Custom HTML templating

ODK Docs uses the `sphinx_rtd_theme`, with some minor customizations.

ODK-specific versions of HTML/Jinja templates are in `_templates`. Any file in that directory will override the file of the same name in the `sphinx_rtd_theme` source.

So, to customize a portion of the HTML template, copy the source file from `sphinx_rtd_theme` and then edit it.

Please commit the copied file unchanged before editing, so that it is easy to track what you have changed.

23.4.3 Custom JavaScript

Custom JavaScript should be added in `src/_static/js/custom.js`. Comment your code with an explanation of what the JS accomplishes, and a reference to the issue number you are working on.

The ODK Docs template includes `JQuery`, so you can use it in your custom JS.

23.4.4 Custom CSS

Custom CSS should be added in `src/_static/css/custom.css`. Comment your code with an explanation of what the CSS accomplishes and a reference to the issue number you are working on.

For example:

```javascript
/* Example CSS PR #xyz */

div[class~='example'] {
  color: black;
}
```
Chapter 23. Contributing to ODK Docs

It is helpful to keep the CSS file organized. There are several sections in the custom.css file:

- Styling for rst roles and directives
- Responsive CSS
- Styling for JS implementation
- Utility classes

Each of these sections are enclosed in start and end comments. Add your code to the relevant section. If you don’t find any section relevant, add a new section and add your code there.

For example:

```html
/* New section starts */
/* Example CSS PR #xyz */

div [class~='example'] {
  color: black;
}
/* New section ends */
```

23.4.5 Style Guide checks

Proselint is used for style testing the docs. Apart from the built-in tests in proselint, custom checks are added for style guide testing. Following a literate programming model, style checks are defined in docs-style-guide.rst. After each style rule, you can define a python code-block containing the code for style testing. When the style-test script is run, these python code-blocks are parsed to generate a testing script.

Proselint dependent checks

In most of the custom checks, a new function is written that calls one of the built-in proselint functions as a return value.

All the checks use a decorator memoize() to cache the check for faster execution.

```python
@memoize()
```

Use @memoize above function definition to cache the result.

Proselint provides several functions for defining style tests:
existence_check(text, list, err, msg, ignore_case=True, str=False, max_errors=float("inf"), offset=0, require_padding=True, dotall=False, excluded_topics=None, join=False)

To check for existence of a regex pattern(s) in the text. The parameters offset, excluded_topics and join are not needed for style guide testing.

Parameters

- **text (str)** – Text to be checked
- **list (list)** – List of regex expressions
- **err (str)** – Name of the test
- **msg (str)** – Error or warning message
- **ignore_case (bool)** – For using re.IGNORECASE
- **str (bool)** – For using re.UNICODE
- **max_errors (float)** – Maximum number of errors to be generated
- **require_padding (bool)** – To use padding with the specified regex (It is better to set it as False and specify the regex accordingly)
- **dotall (bool)** – For using re.DOTALL

Returns The error list consisting of error tuples: [(start, end, err, msg, replacement)].

Return type list

preferred_forms_check(text, list, err, msg, ignore_case=True, offset=0, max_errors=float("inf"))

To suggest a preferred form of the word used. The parameter offset is not needed for style guide testing.

Parameters

- **text (str)** – Text to be checked
- **list (list)** – list of comparison (words or regex): [correct form , incorrect form]
- **err (str)** – Name of the test
- **msg (str)** – Error or warning message
- **ignore_case (bool)** – For using re.IGNORECASE
- **max_errors (float)** – Maximum number of errors to be generated

Returns The error list consisting of error tuples: [(start, end, err, msg, replacement)].

Return type list

---

Our documentation is updated frequently. Get the latest version at [https://docs.getodk.org](https://docs.getodk.org).
**consistency_check**(*text, word_pairs, err, msg, offset=0*)

To check for consistency for the given word pairs. The parameters *offset* is not needed for style guide testing.

**Parameters**
- *text* (*str*) – Text to be checked
- *word_pairs* (*list*) – Word pairs to be checked for consistency
- *err* (*str*) – Name of the test
- *msg* (*str*) – Error or warning message

**Returns** The error list consisting of error tuples: [(start, end, err, msg, replacement)].

**Return type** list

**Note:** The checker functions are used by the built-in proselint function *lint()* to generate an error list of different format. The returned list finally is: [(check, message, line, column, start, end, end - start, "warning", replacements)]

**See also:**
Proselint source code

**Example Usage**

```python
@memoize
def example(text):
    """Example check.""
    err = "style-guide.example"
    msg = "A demonstration for writing checks."
    regex = "[\./?!](example)"

    return existence_check(text, [regex], err, msg, ignore_case=False, require_padding=False)
```

When you define code-blocks which use built-in proselint testing, specify the class *style-checks*.

```python
.. code-block:: python
   :class: style-checks
```

The generated file after parsing code for style checks is *style-checks.py*. 
23.4. Docs Developer Guide

If the test is too large to be defined in the file `docs-style-guide.rst`, you can use a snippet from the test (as here). The code-blocks for such snippets should specify the class `proselint-extra-checks`. Define the complete test in the file `/style-guide/proselint-extra-checks.py`.

**Independent checks**

Apart from the checks, which are to be run through proselint, you can add extra checks to be run independently. They are not enabled in `proselintrc` as well. For example, the checks for finding quote marks and section labels do not use any built-in functions to obtain an error list.

**Example Usage**

```python
def check_quotes(text):
    """Avoid using straight quotes."""
    err = "style-guide.check-quote"
    msg = "Avoid using quote marks."
    regex = r"\[a-zA-Z0-9 ]{1,15}\"
    errors = []
    for matchobj in re.finditer(regex, text):
        start = matchobj.start()+1
        end = matchobj.end()
        (row, col) = line_and_column(text, start)
        extent = matchobj.end()-matchobj.start()
        errors += [(err, msg, row, col, start, end, extent, "warning", "None")]
    return errors
```

The code-blocks for extra checks should specify the class `extra-checks`. The generated file after parsing code for extra checks is `extra-checks.py`.

**Note:** Built-in proselint function `line_and_column()` is used with extra checks to obtain the row and column of the matched text.

`line_and_column(text, start)`

To find the line number and column of a position in a string.

**Parameters**

- **text** *(str)* – Text to be searched for
• **start** (*int*) – Starting position of matched pattern

**Returns** Tuple containing row and column number

**Return type** `tuple`

### Error vs warning

- Warnings are intended to provide guidance to authors.
- Errors enforce “hard” rules, and raising an error will stop the build.

You can classify the result of a check as an error if you are sure that no false positives would be produced. The checks classified as errors should return a replacement for fixing the errors. Proselint dependent checks which use the function `preferred_forms_check()` or `consistency_check()` always return a preferred form. If you create an independent check which generates an error make sure to return a replacement in the error list.

To generate an error from a check, specify the check name in the list of errors in the function `get_errlist()` in the file `style-test.py`.

### Excluding built-in proselint checks

To exclude a built-in proselint check, specify the check name in the check list in the function `exclude_checks()` in the file `style-test.py`.

### 23.5 Tips for Making Good Contributions

#### 23.5.1 Smallest meaningful PR

A PR should normally address one issue. This makes it easier to review, easier to deploy, and easier to roll back in case of a problem. Additionally, the smaller the PR, the less likely it is to create a merge issue.

The exception is when several issues are closely related or can reasonably be worked on together. In this case, it should be clear by looking at the conversation on the Github issues that the items are related and will be worked on together. Your PR message should also make it clear which issues are being worked on, and whether the PR closes the issues or not. Mention the PR by number:

- addresses #123
- closes #123

Our documentation is updated frequently. Get the latest version at [https://docs.getodk.org](https://docs.getodk.org).
23.5.2 Descriptive PR names

A PR title should answer the question, ”What does this Pull Request do?”

Good PR titles:
- adds a video directive
- makes navigation buttons responsive
- swaps placement of nav buttons and file an issue note

Bad PR titles:
- fix issue
- fix #123
- collect

23.5.3 Small, atomic commits

When working locally, commit often. Don’t wait until you have 100 lines of changes across multiple files.
- If you need to copy or move a large section or file, commit that change before also editing it.
- If you have to create a new template file based on an existing template file, copy the file in one commit and then work on the changes. This makes it easier to know what you actually did.
- If writing a new doc, commits after each new section are a good idea.

Commit messages should answer the question, ”What does this commit do?”

Small, well-named commits will help you keep track of your own work and make rollbacks and other changes easier to deal with.

23.5.4 Discuss issues before working

Take the time to clarify the needs and scope of an issue before committing to work on it. Especially for coding tasks, make sure you state your understanding and your plan before working.

If you have a question, ask. Don’t guess.

Note: Many new contributors don’t ask questions because they are worried about appearing under informed. Please set this worry aside.
You will never be judged harshly for asking clarifying questions or for seeking more information.

### 23.5.5 Claim issues

If you decide to work on an issue, let the community know you are working on it by claiming the issue.

```
@getodk-bot claim
```

Once you’ve claimed an issue, other people won’t work on it. So make sure you’re actually going to work on it before claiming it.

Don’t claim more than one or two open issues at a time.

### 23.5.6 Share work in progress

It can be helpful to share your in-progress work. To mark a PR as a work in progress, append `WIP:` to the beginning of the PR title. We will not merge WIP PRs, and we won’t do a review on them unless you ask.

If you want a review, comment, opinion, or help on a WIP PR, please tag the relevant person in the PR comments.

If you finish the work and want the PR to be merged, you do not need to open a new one. Just edit the PR title.

### 23.5.7 If you get stuck while working

- Ask for help in the issue comments. Maybe you can get back on track and complete the issue.
  - Asking questions is always better than guessing.
- Submit a WIP (work in progress) pull request.
  - If we can see what progress you have made, it is easier to offer help.
  - Even if you don’t complete the task, perhaps someone else can pull in your in-progress work and build on it.
  - Sometimes your in-progress work is an improvement over not having it, and so we’ll merge in something even if it isn’t complete.
- If you really cannot move forward, it is okay to abandon an issue.
23.5. Tips for Making Good Contributions

**It is okay to abandon an issue**

Sometimes you simply cannot complete work you have said you were going to complete. This could happen because you don’t have all the required skills or knowledge to complete the work, or because the issue cannot actually be completed as scoped, or because you don’t have the time.

Please let the community know in the issue discussion.

```
@getodk-bot unclaim
```

This way, everyone knows that someone else can take up the project (or that we need to rethink it).

If you did significant work on a project before abandoning it, consider filing a WIP (work in progress) PR, so that others can see what you did and potentially build off of it. (Be sure to mention the issue, so the work is easy to find later.)

### 23.5.8 If an issue takes a long time to complete

For our purposes, a "long time" is a week or more, from when you first announce your intention to work on something until submitting a merge-ready PR.

An issue might take a long time because:

- it is complex and requires lots of hours
- you only have a short period of time each day to work on it
- you are new to the project and are having to learn as you go

The thing that matters is: **Are you actively working on the issue, and making progress, at least a little bit?**

If you are actively working on it, we do not want someone else to jump on and try to work on it at the same time. So please keep the community informed of your work by filing a WIP (work in progress) PR and committing to it as you work.

### 23.5.9 Issues only

All PRs must be directly connected to open issues. PRs should not represent suggestions, good ideas, or independent initiative.

If you have a good idea, file an issue. If you are curious about whether something should be an issue, chat with one of the core team in the #docs-code channel on the Slack.

Once you have filed an issue, wait for comment and approval before diving into the work. We do not want surprise PRs.
23.5.10 Actually install and use ODK or other tools

You cannot write effectively about tools you have not used. If you’re going to write or edit documentation about any of the apps in the ODK ecosystem, you need to spend some time actually using it.

Before diving into writing documentation, try out the core tools here https://getodk.org/software and become familiar with them.

This is also true of writing about Sphinx or any of our documentation build tools. Reading existing documentation is not enough to write about something.

And actually do the thing

If you are writing about a specific process (installing an application, for example), you need to actually complete the process yourself. If possible, follow your own instructions after writing them to make sure they make sense.

23.5.11 Always build locally

Before submitting a PR, run the build locally to make sure you do not produce any errors or warnings. We do not accept PRs that produce errors or warnings.

It is best to run the build frequently as you work. You’ll often catch simple mistakes that are harder to track down later.

23.5.12 You are not an impostor

Impostor syndrome is the feeling that you are not good enough or accomplished enough to do the work you are doing.

We all feel this way sometimes, and that’s okay. But it is important to realize that you are not an impostor.

You can contribute to this community, no matter your background or skills.

- If there is something you don’t know how to do, you can ask.
  - If it is issue related, ask on the issue.
  - If it is more general, try the #docs-code channel in the ODK Developer Slack.
- If you want to try something even though you aren’t sure you can do it, go ahead and try.

Another worry you may have is that something will take you a long time when an ”expert” might be able to do it quickly. You may feel, then, you aren’t the ”right person” for the
job. But if you are the only one with the time or desire to work on something, you are the right person to work on it.

### 23.6 Working with Docs in Windows using Cygwin

The main contribution guide :doc: `contributing` was built for *nix systems, and the same commands may not work in Windows. Cygwin is a Windows tool, equivalent to *nix bash terminal. This guide helps to set up ODK Docs platform from the default Windows command prompt (CMD).

#### 23.6.1 The prerequisites

The following are software tools that you need in the first place. If it is already installed. Just follow the steps and apply what is missing.

- Cygwin
- Python 3
- Virtualenv
- Virtualenvwrapper-win
- Git and GLFS

#### 23.6.2 The requirements

These are a set of the main packages. ODK team combined them in a file such that all will be installed at once.

- alabaster==0.7.10
- Babel==2.4.0
- docutils==0.13.1
- imagesize==0.7.1
- Jinja2==2.9.6
- MarkupSafe==1.0
- Pygments==2.2.0
- pytz==2017.2
- requests==2.14.2
- six==1.10.0

Our documentation is updated frequently. Get the latest version at [https://docs.getodk.org](https://docs.getodk.org).
• snowballstemmer==1.2.1
• Sphinx==1.6.1
• sphinx-rtd-theme==0.2.4
• sphinxcontrib-websupport==1.0.1
• typing==3.6.1
• update==0.4.4

Install Cygwin

Cygwin tool lets Windows users execute many *nix commands. Install Cygwin and add its path to Windows to work completely from the default command prompt. For instructions.

Warning: If you encounter any downloading problems, make sure to select a mirror site near you. The entire list is available on the Cygwin website.

Python 3

You need to install Python 3. Select Python installation that fits your system (32 or 64 bit). For instructions, see.

Make sure to select the option ”Add Python to the Path”, as shown below.
Next, make sure to select "pip", which is a package manager written in Python. We will use it to install packages. See the following image:

Alternatively, if you forgot to add Python 3 to the PATH, add it manually using the following
command:

```bash
> set PATH=%PATH%;C:\Users\your username\python3
```

**Tip:** `C:/Users/your username/python3` is the default Python 3 installation PATH. If you change it, please substitute the above path by the new one.

---

**Virtual Environment**

A virtual environment tool creates multiple Pythons environments, each has its packages and dependencies.

For easy installation, `pip` command can be used, which comes with Python 3 (as shown in Python installation).

```bash
> pip install virtualenv
```

Create a new directory for your odkdocs work:

```bash
> mkdir odk
```

To work with virtualenv, you have two options:

- Use the native virtualenv.
- Use `virtualenvwrapper` on the top of virtualenv.

---

**Native Virtual Environment**

Create a new Python 3 virtual environment, "odkenv" is the name of the virtualenv, you can choose any name.

```bash
> virtualenv -p <python path/python.exe> odkenv
```

After creating the virtualenv, multiple files are copied into the folder odkenv in your working directory.

```bash
> ls odkenv
```

The folder Scripts contains all virtualenv controls as ".bat" files.

To activate the odkenv:
23.6. Working with Docs in Windows using Cygwin

    > cd odkenv
    > cd Scripts
    > odk/odkenv/Scripts/activate.bat

To deactivate the odkenv:

    > odk/odkenv/Scripts/deactivate.bat

**Virtual Environment Wrapper**

**Tip:** This step is not an alternative to virtualenv. You must install virtualenv first.

The Virtualenvwrapper (<https://pypi.python.org/pypi/virtualenvwrapper-win>) mediates between user CMD and virtualenv to ease management and working with multiple virtual environments. To install virtualenvwrapper, use the following:

    > pip install virtualenvwrapper-win

Create a new virtualenv:

    > mkvirtualenv odkenv

Once the odkenv is created, it is automatically activated:

    (odkenv) /odk/docs

To deactivate the odkenv, write:

    > deactivate

To activate the odkenv:

    > workon odkenv

**Git and GLFS**

- Install Git for windows. Make sure that git is installed properly by typing (git) in the CMD.
• Install GLFS.

Android Tools

Install Android Debug Bridge <collect-adb> to learn more about ADB. ADB is part of Android studio, To download

To use ADB, you must run Android Studio once. The default location of ADB is C:/Users/your username/AppData/Local/Android/sdk/platform-tools. Add it to Windows PATH by using the following command:

```
set PATH=%PATH%;C:\Users\your username\AppData\Local\Android\sdk\platform-tools
```

Fork and Clone the ODK Docs repo

From Github, fork the ODK Docs. This will create a copy of the docs in your Github account called origin. Move to the ODk working directory, and clone the ODk Docs into your local machine.

```
> git clone https://github.com/your-github-username/docs.git
```

Set the Upstream Remote

```
> git remote add upstream https://github.com/getodk/docs.git
```

23.6.3 Install the Requirements

This step will install a bunch of packages that are listed in the :file: requirement.txt file. ODK team prepared this file for you to ease the installation.

First, you need to activate your virtual environment (odkenv):

```
> workon odkenv
```

Make sure you are inside the docs folder, then run:

```
$ pip install -r requirements.txt
```

You completed the installation and you can start change and build ODK Docs.
OpenRosa is a set of communication standards for moving blank forms and completed form instances between server applications (like ODK Aggregate) and clients (like ODK Collect). They were originally worked on by the OpenRosa WorkGroup, and they were an outgrowth of the JavaRosa project. The working group is no longer active. However, many apps within the ODK ecosystem (including Collect and Aggregate) continue to implement the API.

This set of docs is intended to provide information about the OpenRosa standards, for use by developers working on applications that communicate with other applications within the ODK ecosystem.

## 24.1 OpenRosa 1.0 APIs

OpenRosa 1.0 APIs were formally approved by the OpenRosa Working Group in December of 2011. To be considered “OpenRosa 1.0 Compliant,” a system must implement all 5 of the OpenRosa 1.0 APIs.

### 24.1.1 Metadata Scheme

This document details the metadata scheme for OpenRosa-compliant XForms. This metadata is used to ensure that critical or extremely useful data pertaining to the XForms and submissions made by client devices is captured and stored.

There are two types of metadata in a form submission.

- **Form metadata** — Data about the identity and version of the XForm used to create the data being submitted.
- **Submission metadata** — Data about the submission itself.

Our documentation is updated frequently. Get the latest version at [https://docs.getodk.org](https://docs.getodk.org).
Blank Form Metadata

A blank form must have two pieces of identifying metadata. These are provided as attributes in the top-level element inside `model/instance`.

```
<model>
  ...
  ...
  <instance>
    <form-name id="getodk.org:all-widgets" version="2.1.30"
      ...
      ...
    </form-name>
  </instance>
</model>
```

Two attributes are required. One provides the unique identity of the form, the other provides the current version number.

Form Identity

One of:

- `id`
- `xmlns`

These values should be in the form of `scheme:value`.

If specified, the `id` value takes precedence over any explicit `xmlns` declaration.

For `id`, the implementer’s registered domain name should be used as part of the scheme (for example: `getodk.org:widgetForm`).

Compliant systems MUST support `id` or `xmlns` lengths up to 249 chars; ideally, servers SHOULD be able to support arbitrary lengths.

Version

A string value indicating the version number of the form.

- MUST support a string value (not a number), to indicate an increment.
- SHOULD support any string schema.
- MUST support arbitrary strings up to 249 characters
24.1. OpenRosa 1.0 APIs

- SHOULD support arbitrary-length strings

The value of version MUST be incremented when any part of the form changes. If version is not present, it is handled as NULL.

**Completed Form Metadata**

In a completed form, metadata is provided in a `<meta>` element inside the `<model>` element.

```
<model>
  ...
  <meta>
    <instanceID>
      uuid:ca90905c-a2db-11e7-abc4-cec278b6b50a
    </instanceID>
    <userID>
      openid:http://www.google.com/profiles/adam.michael.wood
    </userID>
  </meta>
</model>
```

**Fields**

The only required element in the form submission metadata is `<instanceID>`, which must be a universally unique string identifying this specific submission.

Optional fields:

- `<timeStart>` — An ISO 8601 timestamp of when form entry started.
- `<timeEnd>` — An ISO 8601 timestamp of when form entry ended.
- `<userID>` — A unique identifier of the submitting user.
- `<deviceID>` — A unique identifier of device used to generate the submission.
- `<deprecatedID>` — the `<instanceID>` of the submission for which this is a revision. Server software can use this field to unify multiple revisions to a submission into a consolidated submission record.

**ID field formatting**

ID fields ( `<instanceID>`, `<userID>`, etc.) must follow the format `scheme:id`. 
Recommended ID schemes

`instanceID, deprecatedID uuid`

`userID mailto`

`openid`

`deviceID mac`

`uuid`

If you implement a custom ID scheme, it should be prefixed with your domain name, to ensure uniqueness.

```xml
<instanceID>getodk.org:123456789</instanceID>
```

**ID Field Constraints**

- The combined scheme:value keypair MUST be no longer than 249 characters (ie, so that varchar(249) can be used).
  - A robust server SHOULD be able to support an arbitrary length ID, however.
- Only ONE of each type of ID element can be included in a form submitted to the server. That is, only one `deviceID`, one `userID`, one `instanceID`, and one `deprecatedID`.
  - Each ID element MUST have one and only one scheme:value pair.

**Defining expected submission metadata in the blank form**

The blank form MUST specify which of the metadata fields are expected when the completed form is submitted. This is done within a `<meta>` element having the namespace `http://openrosa.org/xforms`. The `<meta>` appears inside the child node of the Primary Instance (the first instance element inside `<model>`).

```xml
<model>
  <instance>
    <data xmlns:orx="http://openrosa.org/xforms"
      id="example.org:myFormId"
      version="1" >
      <orx:meta>
        <orx:timeStart/>
        <orx:timeEnd/>
        <orx:instanceID/>
      </orx:meta>
    </data>
  </instance>
</model>
```
Examples

Blank form metadata

```xml
<h:head>
  <h:title>Metablock example</h:title>
  <model>
    <instance>
      <data xmlns:orx="http://openrosa.org/xforms"
            xmlns="http://example.org/meta"
            version="1">
        <orx:meta>
          <orx:deviceID/>
          <orx:timeStart/>
          <orx:timeEnd/>
          <orx:instanceID/>
        </orx:meta>
      </data>
    </instance>
  </model>
</h:head>
```

Form submission metadata

```xml
<?xml version='1.0'?>
<data version="1"
      xmlns:orx="http://openrosa.org/xforms"
      xmlns="http://example.org/meta">
  <orx:meta>
    <orx:deviceID>uuid:38DN0236SAKWOJNQB3XJI9RW</orx:deviceID>
    <orx:timeStart>2010-08-12T04:08:29.765-5:00</orx:timeStart>
    <orx:timeEnd>2010-08-12T04:10:23.062-5:00</orx:timeEnd>
    <orx:instanceID>uuid.dimagi.org:GEPSJLOGK13TY8L77066GEJJW</orx:instanceID>
    <orx:userID>chwid.dimagi.org:Akende</orx:userID>
  </orx:meta>
</data>
```
24.1.2 HTTP Requests and Responses

Much of OpenRosa communication comes in the form of HTTP requests and responses. In order to properly maintain appropriate formatting and compatibility over time, it is ideal for some information to be consistently provided on both sides of these interactions.

HTTP Requests

HTTP requests (GET, POST) should contain the following headers:

<table>
<thead>
<tr>
<th>Header</th>
<th>Values</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept-Language</td>
<td>The key for what language a response is expected in.</td>
<td>No. Response acceptable in any locale</td>
</tr>
<tr>
<td>X-OpenRosa-Version</td>
<td>1.0</td>
<td>Yes</td>
</tr>
<tr>
<td>Date</td>
<td>The date on the device in format: Mon, 14 Feb 2011 16:48:15 GMT</td>
<td>Yes</td>
</tr>
</tbody>
</table>

HTTP Requests may additionally specify whether the item count is to be included in the response envelope. This is done with a query parameter added to the URI.

<table>
<thead>
<tr>
<th>Arg</th>
<th>Values</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>items</td>
<td>true</td>
<td>false</td>
</tr>
</tbody>
</table>

HTTP Responses

Servers should include the following headers in all responses.

<table>
<thead>
<tr>
<th>Header</th>
<th>Values</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>X-OpenRosa-Version</td>
<td>1.0</td>
<td>Yes</td>
</tr>
<tr>
<td>Date</td>
<td>The date on the server in the format: Mon, 14 Feb 2011 16:48:15 GMT</td>
<td>Yes</td>
</tr>
</tbody>
</table>
24.1. OpenRosa 1.0 APIs

When a response is issued from an OpenRosa server, the format of the response payload will often be defined by the specific API (Form List, for instance). However, APIs which are fundamentally transactional (user registration, form submission, etc) all contain similar semantics. As such, they will utilize a shared XML Envelope of the format:

```
<OpenRosaResponse xmlns="http://openrosa.org/http/response" items=""/> <!-- items: Optional number of how many payloads are included in this envelope -->
  <message nature=""/> <!-- 0 or 1: message payload to be displayed to the user. Nature is an optional tag -->
  <Registration xmlns="http://example.org/user/registration"/> <!-- to group messages by type -->
</OpenRosaResponse>
```

For Example:

```
<OpenRosaResponse xmlns="http://openrosa.org/http/response" items="3">
  <message nature="submit_success">Some message to be displayed to the user</message>
  <Registration xmlns="http://example.org/user/registration"/>
</OpenRosaResponse>
```

APIs using this response can then include their namedpaced payload inside of the response. This response should be handled by the client the same way, regardless of the specific submission. That is, any time an OpenRosa response envelope is received, any of its payloads should be parsed properly.

Any responses included in the envelope that are unrecognized by the client should be ignored without error.

Since the server cannot receive confirmation that a response was successfully retrieved, any responses included in the envelope should be considered idempotent.

The message component of a response envelope should be returned (if possible) in the language specified by the Accept-Language header. The nature attribute of a message is an optional ID which can be used to categorize the type of a response. If the nature of two messages is identical in a bulk operation, for instance, the assumption is that only one of them need be presented to a user (presumably the newest).

As an example, if a server submits 4 xforms, and receives the responses

```
<OpenRosaResponse xmlns="http://openrosa.org/http/response">
  <message nature="submit_success">Thanks, you've submitted 4 forms today</message>
  ...
</OpenRosaResponse>
```
A client could present a message like:

**Bulk Submit Completed**

- Thanks, you’ve submitted 6 forms today
- User ‘paul’ created successfully!

### 24.1.3 Authentication API

This standard specifies the Request and Response format through which OpenRosa compliant servers authorize HTTP transactions.

**Overview**

This API provides a standard means with which OpenRosa clients are authenticated with compliant servers.

OpenRosa compliant devices MUST support both:

- the subset of RFC2617 Digest Authentication defined below
- the Basic Authentication mechanism also outlined in RFC2617.

OpenRosa compliant servers MUST support at least one of either:

- the subset of RFC2617 Digest Authentication defined below
- the Basic Authentication mechanism also outlined in RFC2617.
24.1. OpenRosa 1.0 APIs

We are following RFC2617 with additional OpenRosa compliance requirements defined in the implementation section below to ensure that the Digest Authentication implementations across devices do not compromise security and that they all implement a well-defined common subset of the RFC2617 Digest Authentication mechanism.

Data Security Considerations

Any communication over HTTP (vs. HTTPS) can be observed by others and is susceptible to man-in-the-middle attacks (where a malicious intermediary inserts itself between the client and the server the client intended to contact). As a consequence, if communication is over HTTP, clients may be submitting their form data to a malicious intermediary. That intermediary will see the full contents of the form submission. Additionally, the intermediary may never forward the submission to the intended server — the client can never be certain that the submitted data has been recorded on the intended server.

HTTPS requires that the server be configured with an SSL certificate issued by a signing authority. Man-in-the-middle attacks are possible over HTTPS if clients do not authenticate the server’s SSL certificate (or, less commonly, if the client device or signing authority has been compromised).

OpenRosa compliant client devices MUST authenticate server certificates when establishing HTTPS channels to those servers.

Because client communications are often through unsecured hotspots, it is recommended that HTTPS (with the authentication of server certificates) be used for all communications.

Typical Interaction

A typical transaction consists of the following steps.

1. The client asks for a resource that requires authentication but does not provide a user name and password. Typically this is because the user simply entered the address or followed a link to the page.

2. The server responds with the 401 response code, either requesting Digest Authentication and providing the authentication realm and a randomly-generated, single-use value called a nonce, or requesting Basic Authentication (in which case the server should also redirect and negotiate TLS channel security (https) if the client is not already communicating over https).

3. At this point, the client will present the authentication realm (typically a description of the computer or system being accessed) to the user and prompt for a user name and password. The user may decide to cancel at this point.

4. Once a user name and password have been supplied, the client re-sends the same request but adds an authentication header that includes the response code.
5. Assuming the username and password are valid, the server accepts the authentication and the page is returned. If the user name is invalid and/or the password is incorrect, the server might return the 401 response code and the client would prompt the user again.

**Note:** A client may already have the required user name and password without needing to prompt the user; for example, if they have previously been stored by a client.

For Basic Authentication, the "response" value is simply a base-64 compression of the user name concatenated with "::" and the plain-text password, as specified in RFC2617.

**Warning:** Anyone with a network sniffer could read this value, decompress it, and obtain the user name and password. Hence, it is critical that this information only be transmitted over HTTPS or some other secure transport.

Even with HTTPS, sending this is not a highly secure practice.

For Digest Authentication, the "response" value is calculated in three steps, as follows. Where values are combined, they are delimited by colon symbols.

1. The MD5 hash of the combined user name, authentication realm and password is calculated. The result is referred to as HA1.

2. The MD5 hash of the combined method and digest URI is calculated, e.g. of "GET" and "/dir/index.html". The result is referred to as HA2.

3. The MD5 hash of the combined HA1 result, server nonce (nonce), request counter (nc), client nonce (cnonce), quality of protection code (qop) and HA2 result is calculated. The result, the user name and the cnonce are the "response" value provided by the client.

The Digest Authentication response value is thus sent in such a way that an adversary can extract the user name from the response, but cannot extract the password from the response. It can therefore be sent over an unsecured channel (for example, HTTP).

**Note:** Even with the more secure Digest Authentication, HTTPS is recommended.

**Implementation**

Servers which implement the Authorization API should follow the specifications provided below in order to be compliant with OpenRosa standards.

- All HTTP interactions MUST be HTTP 1.1
24.1. OpenRosa 1.0 APIs

- Servers MUST conform to RFC2617 for returning one or more authentication schemes in their 401 challenge. These define the authentication interactions that the server is willing to accept from the client (for example: Basic, Digest)
- Any server interactions MAY be unauthenticated.
- Non-device (for example, browser) interactions for which the server requires authentication are NOT required to support Basic or the OpenRosa Restricted Digest authentication scheme. That is, they are allowed to only support Form-based or some other authentication scheme.
- Device-and-server interactions for which the server requires authentication MUST implement either Basic authentication or the OpenRosa Restricted Digest authentication scheme as detailed below. The server or device MAY additionally implement other authentication schemes.
- The device MUST make every effort to proactively supply an Authentication header line if the requested URI falls within the list of domain URIs covered by a previous authentication interaction. This is to minimize the number of authentication challenges.

Authentication

Clients MUST NOT include authentication credentials in the URL to the server. That is, this syntax is strictly forbidden:

```
http://username:password@myhost.org/mypage
```

Basic Authentication

Basic Authentication MUST NOT be performed over a non-secure (HTTP) connection.

Once the client is aware that basic authentication is required, it SHOULD proactively supply the basic authentication credentials on every secure request to the server, rather than wait for the server to reject the request with a 401 response.

OpenRosa Restricted Digest Authentication

This is the Digest Access Authentication Scheme (RFC 2617 Section 3) with the following restrictions:
- algorithm — server MUST omit or specify "MD5"
- domain — server MUST specify to help device with proactive inclusion of Authenticate: header records.
- qop — device MUST support: omitted and "auth"; server MAY request any of these.
• opaque — device MUST return if supplied; server MAY supply this or omit it
• stale — device MUST make every effort to not prompt the user for username and password if this is TRUE but instead recompute the key with previously cached values for the username and password.
• cnonce — device SHOULD use a string representation of at least a 48-bit random value (a random UUID has 126-bit randomness) for the cnonce.

Digest Authentication Security Considerations

Digest Authentication is based upon the MD5 hash algorithm which is now considered too weak for mainstream cryptographic uses. Digest Authentication remains viable only when the cnonce and nonce values are random and reasonably long. The use of longer random strings (e.g., random UUIDs have 126 bits of randomness) is critical for the continued use of this authentication mechanism.

Digest Authentication Calculations

RFC 2069

| HA1 = MD5(A1) = MD5(username:realm:password) |
| HA2 = MD5(A2) = MD5(method:digestURI) |
| response = MD5(HA1:nonce:HA2) |

RFC 2617 (HTTP Authentication: Basic and Digest Access Authentication)

| HA1 = MD5(A1) = MD5(username:realm:password) |
| if qop directive's value is "auth" or unspecified, then HA2 is: |
| HA2 = MD5(A2) = MD5(method:digestURI) |
| if qop directive's value is "auth-int" then HA2 is: |
| HA2 = MD5(A2) = MD5(method:digestURI:MD5(entityBody)) |
| if qop directive's value is "auth" or "auth-int" then compute the response: |
| if qop directive is unspecified |

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
24.1. OpenRosa 1.0 APIs

response = MD5(HA1:nonce:HA2)

(the above shows that when qop is not specified, the simpler RFC2069 standard is followed)

Digest Authentication Example Interaction

No authentication

Request:

GET /dir/index.html HTTP/1.0

Response:

HTTP/1.0 401 Unauthorized
Server: HTTPd/0.9
Date: Sun, 10 Apr 2005 20:26:47 GMT
WWW-Authenticate: Digest realm="testrealm@host.com",
    qop="auth,auth-int",
    nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
    opaque="5ccc069c403ebaf9f0171e9517f40e41"
Content-Type: text/html
Content-Length: 311

<!DOCTYPE HTML PUBLIC "~-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/1999/REC-html401-19991224/loose.dtd">

<html>
    <head>
        <title>Error</title>
        <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
    </head>
    <body>
        <h1>401 Unauthorized.</h1>
    </body>
</html>

Request with username and password

Request (username "Mufasa", password "Circle Of Life")
GET /dir/index.html HTTP/1.0
Host: localhost
Authorization: Digest username="Mufasa",
realm="testrealm@host.com",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
uri="/dir/index.html",
qop=auth,
nc=00000001,
cnonce="0a4f113b",
response="6629fae49393a05397450978507c4ef1",
opaque="5ccc069c403ebaf9f0171e9517f40e41"

Response

HTTP/1.0 200 OK
Server: HTTPd/0.9
Date: Sun, 10 Apr 2005 20:27:03 GMT
Content-Type: text/html
Content-Length: 7984

Example calculation of response using MD5

HA1 = MD5( "Mufasa:testrealm@host.com:Circle Of Life" )
= 939e7578ed9e3c518a452acee763bce9

HA2 = MD5( "GET:/dir/index.html" )
= 39aff3a2bab6126f332b942af96d3366

Response = MD5( "939e7578ed9e3c518a452acee763bce9: \\
dcd98b7102dd2f0e8b11d0f600bfb0c093: \\
00000001:0a4f113b:auth: \\
39aff3a2bab6126f332b942af96d3366" )
= 6629fae49393a05397450978507c4ef1

24.1.4 Form Submission API

This standard defines the API for submitting data to an OpenRosa compliant server and receiving a response from the server about the submission. This API is designed to provide a base level of interoperability between OpenRosa clients and servers while enabling application-specific extensions as well.

Form Submission

OpenRosa servers must be able to receive XForms which are submitted by clients over HTTP.
24.1. OpenRosa 1.0 APIs

There are 3 major categories of requirements for submission which must be fulfilled:

- channel
- content
- correctness.

Channel

OpenRosa servers MUST provide a URI capable of accepting HTTP POST Requests. Access controls, firewalls and geographic or networking restrictions MAY limit the origins and/or clients that can access the server.

The server MUST support HTTP 1.1 chunked transfer encoding for receiving the POST content. There is no minimum set of standards for timeouts, maximum content length, or other http configurations, but servers are encouraged to support lenient connections and the largest possible content size, since many of the connections will be from unreliable channels and in environments where splitting content up is impractical.

For maximum compatibility with J2ME clients, it is recommended that a server SHOULD NOT issue redirects.

Note: Using digest authentication and https when communicating with a server does not require any redirects — you can have authentication and secure transport without redirects.

Content

OpenRosa submissions (and responses) MUST provide headers conforming to the OpenRosa Request standard.

Successful server responses MUST include an X-OpenRosa-Accept-Content-Length header in addition to the required OpenRosa Request response headers. The X-OpenRosa-Accept-Content-Length header specifies this server’s recommended maximum size for a POST body, in bytes.

Servers SHOULD include this header in their error responses. However, clients MUST NOT rely on the presence of this header (or any OpenRosa header) in every error response.

Note: Overly-large requests sent to Google App Engine will be rejected prior to being sent to any server, and will therefore not contain this (or any) OpenRosa header.

Our documentation is updated frequently. Get the latest version at https://docs.getodk.org.
OpenRosa submissions are POSTed to servers as a multipart MIME Envelope with at least 1 part — the XML content of the form itself. Each of these parts should adhere to the following requirements

**MIME Envelope**

- Content Type: multipart/form-data
- Contains Exactly 1 XForm Part
- Contains 0 or More Additional Parts

**XForm Part**

- Content Type: text/xml
- Name: xml_submission_file

**Additional Parts**

- Content Type: arbitrary
- Name: matches an appropriate element inside of the XForm Element’s XML.

Servers MAY be more permissive than this specification (for example, allowing multipart/mixed for the mime envelope), but MUST be capable of recognizing and properly receiving submissions in this format.

**Correctness**

The server MUST consume an entire HTTP POST in conformance with that protocol. Once a POST is received, the range and structure of the server’s response is specified below.

**Extended Transmission Considerations**

If the client is capable of negotiating authenticated and/or secure transmissions to the server, it is recommended that the client first attempt a HEAD request to the server to negotiate the authentication and channel security prior to the first POST of the data, regardless of any channel security stated in the <submission> element of the form. This ensures that submitted data is not inadvertently sent in the clear on that first request due to the client device possessing an out-of-date form definition with inaccurate <submission> content.
Issuing a HEAD request first also reduces overall transmission bandwidth in instances where client authentication occurs. Most authentication handshake protocols require the complete re-POST of the original request after the authentication is complete. By delaying the POST of the data until after the authentication has been negotiated on the HEAD request, overall transmission bandwidth is reduced.

The client, upon receiving the HEAD response, can check that the response contains the X-OpenRosa-Version header to indicate that the host is an OpenRosa server (as opposed to a network login screen or proxy page). In this way, the server can change the scheme from http to https or update the submission page URL without a wasted redirect during the actual submission process. The client can also use the value of the X-OpenRosa-Accept-Content-Length header to inform itself of how big a POST body should be sent to this server. ODK Collect also requires a 204 (No Content) status code in the HEAD response.

If a full POST of the form’s XML submission and its additional parts (for example, captured image, audio or video clips) would exceed the size specified in the X-OpenRosa-Accept-Content-Length header (the maxSize), it is recommended that the client split the POST into multiple individual POST requests, each containing the form’s XML submission and one or more additional parts such that each partial POST request is no greater than maxSize; if a single additional part is greater than maxSize, the POST should contain the form’s XML submission and that single additional part. Regardless of whether the client observes and honors the X-OpenRosa-Accept-Content-Length header, a compliant server with give its best effort to accept submissions of any length.

The X-OpenRosa-Accept-Content-Length header is provided to avoid failures that may otherwise arise due to restrictions on the overall size of HTTP messages, or due to physical or virtual memory configurations of the server.

The form’s XML submission is sent on each POST so that a client can avoid having any knowledge about the content of the files it is shipping around. Doing so also places the fewest restrictions on how the server handles the submission.

**Rationale for sending the form’s XML submission**

To avoid sending the form’s XML submission, you would need to inspect the submission and send up its identifying information. By avoiding inspection of the submission, a much simpler Ajax-enabled webpage could conform to this API.

On the server, having just the instanceID sent on subsequent POST requests might not be sufficient to process the request — sending only this information would burden those server implementations with maintaining a mapping from the instanceID to the natural key for this data. Not sending the form’s XML submission in subsequent POSTs biases against some server designs.

Finally, since most XML submission documents are smaller than 2K bytes, and if you have a 10M byte threshold for splitting a submission across multiple requests (a reasonable lower
limit), you’re burning only 0.02% of your bandwidth with the retransmission.

**Server Response Format**

The server response format will be XML formatted, and the response body will be an *OpenRosa Response*.

Example response:

```xml
<OpenRosaResponse xmlns="http://openrosa.org/http/response">
    <message>Form Received! You’ve submitted 5 forms today</message>
</OpenRosaResponse>
```

If the server is RESTful, the server MAY return an HTTP URI (using the standard HTTP Location header) where the form can be found.

A form should not be assumed to be submitted until a 201 or 202 response code is received with an OpenRosaResponse envelope body.

**Server Status Codes**

Server status codes will be the same as *standard http codes*. These use the general classification:

- **201-202** Successfully received by server.
- **4XX** Client Error
- **5XX** Server Error

*Note:* 1XX (informational) and 3XX (redirection) probably do not apply to these POSTs.

Some common interpretations of codes are below, but more could apply.
24.1. OpenRosa 1.0 APIs

<table>
<thead>
<tr>
<th>Code</th>
<th>HTTP Meaning</th>
<th>ODK Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>UN-USED</td>
<td>Since the request is a post, a 200 response is not a sign of a successful submission. Many intermediate proxies will return a 200 response for gateway pages on WI-FI, etc, so receiving a 200 shouldn’t be assumed to be meaningful.</td>
</tr>
<tr>
<td>201</td>
<td>Form Received</td>
<td>Everything went great. Thanks for submitting.</td>
</tr>
<tr>
<td>202</td>
<td>Accepted</td>
<td>We got and saved your data, but may not have fully processed it. You should not try to resubmit.</td>
</tr>
<tr>
<td>204</td>
<td>No Content</td>
<td>Status returned in response to a HEAD request.</td>
</tr>
<tr>
<td>401</td>
<td>Unauthorized</td>
<td>Client tried to post something it didn’t have permission to post.</td>
</tr>
<tr>
<td>403</td>
<td>Forbidden</td>
<td>You’re not allowed to post to this server.</td>
</tr>
<tr>
<td>404</td>
<td>Not Found</td>
<td>Unknown URI endpoint, domain, or other</td>
</tr>
<tr>
<td>413</td>
<td>Request too large</td>
<td>The request body is too large for the server to process</td>
</tr>
<tr>
<td>500</td>
<td>Internal Server Error</td>
<td>Something went awry on the server and we’re not sure what it was</td>
</tr>
</tbody>
</table>

24.1.5 Form List API

This standard specifies how clients discover a list of available blank forms on a server.

Discovery Request

The discovery request should be sent in compliance with the HTTP 1.1 protocol.

If a server will filter the set of forms based upon the user’s identity, then the server should require that the user be authenticated through either the Authentication API or through an
alternative authentication mechanism. The server can then make use of the user’s authenticated identity through those mechanisms to filter the set of forms to be returned.

The device will make a discovery request to a configured URI with a single query parameter, the deviceID. The deviceID should be the same id as provided by the default population mechanism defined in the Metadata Scheme. The server may filter the set of forms returned using this information.

**Warning:** The deviceID is advisory information only, and is inherently not authoritative. Anyone can claim to have a given IMEI, for example.

Together, the authentication and deviceID enable a server to tailor the set of forms to both the user and the device (and therefore the device’s capabilities).

**Query Parameters**

Optional query parameters MAY also be supplied:

- formID If specified, the server MUST return information for only this formID.
- verbose If specified with the value true, the server MAY include a `<descriptionText>` or `<descriptionUrl>` element providing a longer description of an XForm.
- listAllVersions If specified, provides a listing of all hosted versions of each form (including the `<version>` element) in the response document (see below).

If not supplied or not present, the server MUST NOT include this optional information in its response.

**Request Header**

Requests from the device SHOULD set the `Accept-Language` header to indicate the preferred language of the form name.

**Read More**

The `Accept-Language` HTTP header is documented here:

http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.4

and an article on its evolving use for determining locale is here:

http://www.w3.org/International/questions/qa-accept-lang-locales
Aside from the query parameters, the structure of the server URI and whether the request is made over http or https, is entirely implementer- and server-dependent.

Compliant servers MUST NOT require additional query parameters on this request.

Compliant devices MUST be able to handle arbitrary http and https URIs and must be able to authenticate with the server during this transaction, if required by the server.

Successful Response Document

If accepted by the server, the server will respond with a Content-Type: text/xml; charset=utf-8 response. An example of such a document is shown below:

```xml
<?xml version='1.0' encoding='UTF-8' ?>
<xforms xmlns="http://openrosa.org/xforms/xformsList">
  <xform>
    <formID>mydomain.org:formId</formID>
    <name>Form with zero or more additional files</name>
    <version>1.1</version>
    <hash>md5:c28fc778a9291672badee04ac880a05d</hash>
    <descriptionText>A possibly very long description of the form</descriptionText>
    <downloadUrl>http://myhost.com/app/path/getMe/formIdA</downloadUrl>
    <manifestUrl>http://myothehost.com/app/path/getOtherStuff?formId=formIdA</manifestUrl>
  </xform>
  <xform>
    <formID>http://mydomain.org/uniqueFormXmlns</formID>
    <name>Form without additional files</name>
    <version>v50 alpha</version>
    <hash>md5:c28fc778a9291672badee04ac770a05d</hash>
    <descriptionUrl>http://mysecondhost.com/a/description/getMe@formId=uniqueKey</descriptionUrl>
    <downloadUrl>http://mysecondhost.com/a/different/path/getMe@formId=uniqueKey</downloadUrl>
  </xform>
  <xforms-group>
    <groupID>someId</groupID>
    <name>Short name of grouping</name>
    <listUrl>http://whateverhost.com/other/path/forDownload?group=fido</listUrl>
    <descriptionText>Longer description of what is here</descriptionText>
    <descriptionUrl>http://morehost.com/description/link</descriptionUrl>
  </xforms-group>
</xforms>
```

This document consists of:
• a top-level `<xforms/>` element in the http://openrosa.org/xforms/xformsList namespace enclosing,
  – zero or more `<xform/>` tags followed by
  – zero or more `<xforms-group/>` tags.

Within each of the `<xform/>` tags, there must be exactly one of each of the following:

• `<formID/>`
• `<name/>`
• `<version/>`
• `<hash/>`
• `<downloadUrl/>`

There may be zero or one

• `<descriptionText/>`
• `<descriptionUrl/>`
• `<manifestUrl/>`

Within each `<xforms-group/>` tag, there must be exactly one of each of the following:

• `<groupID/>`
• `<name/>`
• `<listUrl/>`

There may be zero or one

• `<descriptionText/>`
• `<descriptionUrl/>`

The `<xform/>` tag provides information about a single form. The `<xforms-group/>` tag provides information about a group of forms; a further enumeration of the forms within that group can be obtained through the `<listUrl/>` of that group (which returns an `<xforms/>` document). Groups can be used to define sets of forms that a user may wish to download together (such as for clinical studies, for example).

**Elements within `<xform/>`**

• `<formID/>` The *form identity as defined in the metadata.*

• `<name/>` The user-friendly display name of the form. The server may localize this name (translate it) based upon the `Accept-Language:` header on the incoming request. Devices **SHOULD** send this header and servers **MAY** return different name and
24.1. OpenRosa 1.0 APIs

description text based upon its value. The default behavior is to return the text within the <title> element of the XForm.

- <version/> The form version as defined in the metadata The device MAY use this to determine if its XForm definition is out of sync with the server (over time, the server may roll the current version forward or backward).

- <hash/> The hash value of the form definition file available for download. The only hash values currently supported are MD5 hashes of the file contents; they are prefixed by md5:. If the hash value identified in the form list differs from the hash value for a previously-downloaded form definition file, then the file should be re-fetched from the server.

- <downloadUrl/> A fully formed URI for downloading the form to the device. It may be a valid http or https URI of any structure; the server may require authentication; the server may require a secure (https) channel, etc.

- <manifestUrl/> A URI from which the device can obtain a manifest defining additional supporting objects and files. Optional

- <descriptionText/> A detailed text explanation of the form. Optional, only returned if verbose=true.

- <descriptionUrl/> A fully qualified URI pointing to a media (audio, video) description of the form. Optional, only returned if verbose=true.

Tip: A media description of the form (audio or video) can be especially useful in low-literacy populations.

Elements within <xforms-group/>

- <groupId/> The unique id for this group. It is locale-invariant (in contrast to the <name/> element, below).

- <name/> The user-friendly display name of the xforms group. The server may localize this name (translate it) based upon the Accept-Language: header on the incoming request. Devices SHOULD send this header and servers MAY return different name and description text based upon its value.

- <listUrl/> A fully qualified URI for obtaining the <xforms/> document of this grouping of forms. (That is, the endpoint for the Form List API). It may be a valid http or https URI of any structure; the server may require authentication; the server may require a secure (https) channel, etc.

- <descriptionText/> A detailed text explanation of the form group. Optional, only returned if verbose=true.
• `<descriptionUrl/>` A fully qualified URI pointing to a media (audio, video) description of the form group. *Optional, only returned if `verbose=true`.*

**Note:**

• The server **MAY** dynamically construct the download and manifest URLs based upon the user identity and device id.

• The manifest **MAY** include additional (implementation specific) elements and data. These **MUST** be dealt with gracefully (ignored) by the client if it does not know how to interpret these fields.

### The Manifest Document

The structure of the manifest document returned by the manifest URI is as follows:

```xml
<?xml version='1.0' encoding='UTF-8' ?>
<manifest xmlns="http://openrosa.org/xforms/xformsManifest">
  <mediaFile>
    <filename>badger.png</filename>
    <hash>md5:c28fc778a9291672badd04ac880a05d</hash>
    <downloadUrl>http://funk.appspot.com/binaryData?blobKey=%3A477e3</downloadUrl>
  </mediaFile>
  <mediaFile>
    <filename>path/to/agilefrog.png</filename>
    <hash>md5:9fd39ac868eccdc0c134b3b7a6a25eb7</hash>
    <downloadUrl>http://other.appspot.com/blobSource?foo=222</downloadUrl>
  </mediaFile>
</manifest>
```

This document consists of:

• a top-level `<manifest/>` tag in the `http://openrosa.org/xforms/xformsManifest` namespace enclosing
  – zero or more `<mediaFile/>` tags.
  – Within each of the `<mediaFile/>` tags, there must be exactly one of each of the following:
    * `<filename/>`
    * `<hash/>`
    * `<downloadUrl/>`
24.1. OpenRosa 1.0 APIs

Elements within `<mediaFile/>`

- `<filename/>` The unique un-rooted file path for this media file. This un-rooted path must not start with a drive name or slash and must not contain relative path navigations (for example, . or ..).

- `<hash/>` The hash value of the media file available for download. The only hash values currently supported are MD5 hashes of the file contents; they are prefixed by `md5:`. If the hash value identified in the manifest differs from the hash value for a previously-downloaded media file, then the file should be re-fetched from the server.

- `<downloadUrl/>` A fully qualified URI for downloading the media file to the device. It may be a valid http or https URI of any structure; the server may require authentication; the server may require a secure (https) channel, etc.
JavaRosa is a Java library for rendering forms that are compliant with ODK XForms spec. It is the heart of many of the ODK tools. ODK JavaRosa is a fork of JavaRosa 1.0 that has been modified to NOT run on J2ME devices. The key differences are:

- Regularly updated to ensure spec compliance
- KoBo additional instance defn. and filter paths
- Remember all bind attributes and any additional attributes on <input>, <select>, <group>, etc. statements
- Numerous enhancements and contributions from SurveyCTO and others
- J2ME sub-projects removed

25.1 Architecture

25.2 Integration Points

This section describes how the different parts of ODK use JavaRosa.

25.2.1 Aggregate

25.2.2 Briefcase

25.2.3 Collect

25.2.4 Validate

Two source files in Validate use JavaRosa.
FormValidator.java

core/model/Constants.java
core/model/FormDef.java
core/model/FormIndex.java
core/model/GroupDef.java
core/model/IFormElement.java
core/model/SelectChoice.java
core/model/condition/EvaluationContext.java
core/model/condition/IFunctionHandler.java
core/model/data/IAnswerData.java
core/model/instance/InstanceInitializationFactory.java
core/model/instance/InvalidReferenceException.java
core/model/instance/TreeElement.java
core/model/instance/TreeReference.java
core/model/utils/IPreloadHandler.java
core/model/utils/QuestionPreloader.java
core/services/PropertyManager.java
core/services/PrototypeManager.java
form/api/FormEntryCaption.java
form/api/FormEntryController.java
form/api/FormEntryModel.java
form/api/FormEntryPrompt.java
model/xform/XFormsModule.java
xform/parse/XFormParseException.java
xform/util/XFormUtils.java

StubPropertyManager.java

core/services/properties/IPropertyRules.java
core/services/IPROPERTYMANAGER.java
LAUNCHING ODK COLLECT FROM EXTERNAL APPS

See also:

Launching External Apps

ODK Collect supports several intents which allow it to be launched by external applications. You can open a specific form or lists of empty forms, saved forms, finalized forms or sent forms.

This section describes how to launch ODK Collect and open its activities from an external app. The code samples go in your custom Android application.

26.1 Understanding Intents

An Intent is a messaging object you can use to request an action from another app component. For more details on intents, you can refer to these Android docs.

26.2 Launching Collect activities from external application

To start one of ODK Collect’s activities:

1. Create a new intent using an appropriate action.
2. Set the type of the created intent.
3. Start an activity using the intent.

26.2.1 Launching the form list or instance list activity

```
Intent intent = new Intent(Intent.ACTION_VIEW);
intent.setType("vnd.android.cursor.dir/vnd.odk.form");
startActivity(intent);
```
Chapter 26. Launching ODK Collect from External Apps

This displays a list of forms and allows the user to select one and fill it.

Similarly for an instance of the form:

```java
Intent intent = new Intent(Intent.ACTION_VIEW);
intent.setType("vnd.android.cursor.dir/vnd.odk.instance");
startActivity(intent);
```

This displays a list of saved forms and allows the user to select one and edit it.

### 26.2.2 Getting the URI of a form or instance chosen by the user

```java
Intent intent = new Intent(Intent.ACTION_PICK);
intent.setType("vnd.android.cursor.dir/vnd.odk.form");

static final int PICK_FORM_REQUEST = 1; // The request code
startActivityForResult(intent, PICK_FORM_REQUEST);
```

To get the result, override `onActivityResult()` method in the following way:

```java
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    // Check which request we're responding to
    if (requestCode == PICK_FORM_REQUEST) {
        // Make sure the request was successful
        if (resultCode == RESULT_OK) {
            // The Intent's data URI identifies which form was selected.
            Uri formUri = data.getData();
            // Do something with the form here
        }
    }
}
```

For an instance, change the intent type:

```java
intent.setType("vnd.android.cursor.dir/vnd.odk.instance");
```

### 26.2.3 Using a URI to edit a form or instance

If the URI of a form or instance is known, it can be viewed or edited. For example, a URI received in `onActivityResult()` as described above can be used.
26.2. Launching Collect activities from external application

```java
Intent intent = new Intent(Intent.ACTION_EDIT);
intent.setData("content://org.odk.collect.android.provider.odk.forms/forms/2");
startActivity(intent);
```

The same thing can be done with a specific instance.
CHAPTER TWENTYSEVEN

BRIEFCASE AGGREGATE API

27.1 Introduction

This describes the interfaces ODK Briefcase uses when interacting with ODK Aggregate v1.0.

In general, the server should adhere to the OpenRosa API described here.

27.2 Pull

Interfaces used during Pull actions.

The APIs used:
1. OpenRosa Form Discovery API
2. Download forms and media files using URLs form discovery.
3. view/submissionList to obtain a chunk of submission keys.
4. view/downloadSubmission to download an individual submission.

27.3 Push

Interfaces used during Push actions.

1. formUpload to upload a form and its media files to ODK Aggregate.
2. view/submissionList to obtain a chunk of submission keys.
3. OpenRosa Form Submission/Overwrite API (with extensions)
27.4 Interface Details

27.4.1 OpenRosa Form Discovery API

Vanilla conformance.

27.4.2 OpenRosa Form Submission/Overwrite API

Server Processing

The server does special processing to look for an instanceID or a submissionDate attribute on the top-level element of the submission.

If the OpenRosa Metadata block is not present, it uses the instanceID attribute value as the instance ID for this submission.

If the submissionDate is present, it uses that value as the submission date for this submission (rather than the current datetime). SubmissionDate should be in ISO8601 format.

Response

The response from the server looks like:

```xml
<OpenRosaResponse xmlns="http://openrosa.org/http/response">
  <message>...</message>
  <submissionMetadata xmlns="http://www.getodk.org/xforms"
                        id="formid" version="..." encrypted="yes" instanceID="..."
                        submissionDate="..."
                        isComplete="true" markedAsCompleteDate="..."/>
</OpenRosaResponse>
```

The <submissionMetadata/> tag contains all the metadata for this submission that the server has assigned it.

The form id and version are given (version is omitted if null).

If the form is an encrypted form, the encrypted attribute is present, otherwise it is omitted. If present, it has a value of yes (vs. true).

The instanceID is either the server-assigned instance id or the instanceID from the OpenRosa Metadata block.

The submissionDate is the date of first submission of this record on the server.

The isComplete flag is true if the server has all attachments associated with this filled-in form. Otherwise, it will be false. If it is true, the markedAsCompleteDate will be the date when the submission became complete.
The date fields are formatted using:

```java
import org.javarosa.core.model.utils.DateUtils;

DateUtils.formatDateTime(d, DateUtils.FORMAT_ISO8601)
```

The only 2 fields ODK Briefcase currently uses are the `instanceID` and `submissionDate` fields. It may eventually use the `isComplete` flag.

### Briefcase Treatment

After a submission has been pushed to the server, it updates the top-level element, inserting the `instanceID` and `submissionDate` fields if they are not already present.

If they are present, it compares the fields and warns if there is any discrepancy between them.

#### 27.4.3 Download forms and media files

It is same as the API ODK Collect uses.

ODK Briefcase downloads the forms so that it has the form model available for CSV exports and to ensure that a push uses matching form definitions.

#### 27.4.4 formUpload

This is `FormUploadServlet` in ODK Aggregate.

It is a multipart POST. The POST always contains the form definition file, and has as many media files as would fit in under 10MB. This follows the behavior of the OpenRosa Form Submission/Overwrite API.

Because of the limitations of HTML file input tags, all media files must be in a single folder (no sub-directories). The server assumes this is the case.

Here is a sample web page fragment to post to this servlet:

```html
<form id="ie_backward_compatible_form" action="/formUpload"
   enctype="multipart/form-data" encoding="multipart/form-data"
   method="POST" accept-charset="UTF-8">
   <table id="uploadTable">
   <tbody>
   <tr>
   <td>
   <label for="form_def_file">Form definition (xml file):</label>
   </td>
   ```
27.4.5 GET view/submissionList

Download the list of completed submissions for a given form. Incomplete or partial submissions must not be returned. You may wish to report only approved submissions if you have a QA review step.

There are 3 query arguments:

1. formId – identifies the form. Just the id from the form definition (e.g., geo_tagger_v2).
2. cursor – an opaque string used by the server to as a query resume point. Omit on the first call.
3. numEntries – the number of entries to return.

Returned Document

The returned XML document is of the form:
Where the idList contains a series of id elements containing strings that can be used to construct the needed string to pass to the view/downloadSubmission API.

The resumptionCursor holds opaque data that is used by the server to track the location at which to resume the list of ids.

**ODK Briefcase Treatment**

ODK Briefcase repeatedly calls this API, passing in the previous response’s resumptionCursor value until the returned resumptionCursor value matches that given in the request. Once it no longer changes, ODK Briefcase assumes that all id data has been downloaded from the server.

**27.4.6 GET view/downloadSubmission**

Download an individual submission.

One Query argument: formId – a multipart XPath-like string query that identifies the individual submission.

The formId is a string of the form:

```
formid[@version=null and @uiVersion=null]/topElement[@key=idvalue]
```

- formid is the form id, as would be provided to the submissionList API.
- version is either null if not used or the version number of the form (ODK Aggregate only supports integer version strings at this time).
- topElement is the name of the top-level element in the submission. This is the element nested just within the <instance> element of the <model> (it also has the id attribute, which should be equal to the formid).
- idvalue is the value of the individual submission id returned by the view/submissionList API, usually a UUID.

An example, where no version was attached to the form definition uploaded to Aggregate:
GET /view/downloadSubmission?formId=my_odk_form[@version=null and]
->@uiVersion=null]/data[@key=uuid:38fd4ef4-28b9-441e-a818-dd8cbe514b2c]

While it is desirable for the idvalue to be the instanceID of the submission, it is not required. For ODK Aggregate v1.x, it will be the instance ID; for ODK Aggregate 0.9.x, it is not the instance ID.

**Note:** formid may be a URL (since it might be an xmlns), so when parsing this query parameter, it is safest to find the last instance of @version and split the string at that location. In any case, it is your string to parse and interpret for your server configuration.

### Response Document

The response is of the form:

```xml
<submission xmlns="http://getodk.org/submissions"
xmlns:orx="http://openrosa.org/xforms">
  <data>
    ...reconstructed submission XML...
  </data>
  <mediaFile>
    <fileName>...</fileName>
    <hash>md5:...</hash>
    <downloadUrl>...</downloadUrl>
  </mediaFile>
  ...repeat as needed...
</submission>
```

The `<mediaFile>` tag has the same interpretation as it does in the OpenRosa Form Listing API’s manifest XML.

The reconstructed submission XML generally does not respect the namespaces of the original form definition. As a special case, if it finds a form group that could be interpreted as the OpenRosa Metadata block, it does use the orx namespace for that.

Like the `<submissionMetadata>` tag on the Form Submission/Overwrite API response, the top-level element in the submission XML contains all the metadata fields supplied by the server.
See also:

ODK Central can manage encryption for your projects.

## 28.1 Overview

Encrypted forms provide a mechanism to keep your data private even when using http: for communications (for example, when you do not have an SSL certificate or https: is not available). Encrypted forms may also enable deployments using cloud providers to comply with data privacy laws, eliminating the necessity for setting up your own servers to meet those requirements.

Encrypted forms apply asymmetric public key encryption at the time the form is finalized within ODK Collect. This encrypted form can then be submitted to a server. ODK Central provides a managed encryption option which means a decrypted data file can securely be downloaded without using any other tool. Central self-managed encryption and other servers require using ODK Briefcase to download and decrypt the data. Briefcase, when supplied with the asymmetric private key (which Briefcase never stores), can then decrypt and export the form data as a CSV file for your use.

The finalized form’s data (and media attachments) are encrypted before being submitted to a server and remain encrypted while stored on the server. When using Briefcase, the data remains encrypted as it is pulled, where it is again stored in encrypted form.

### Note:

- A server cannot present or share the data with another service since the encryption obscures the entire contents of the form and the server never possesses the asymmetric key required to decrypt the form.

- When using encrypted forms, the server serves only as a data aggregation point.
The non-encrypted data is available on the Collect device during data collection and whenever a form is saved without marking it as complete. Once you mark a form as complete (finalize it), Collect will generate a random 256-bit symmetric key, encrypt the form contents and all attachments with this key, then construct a submission manifest which describes the encrypted submission and an asymmetric-key encryption of the symmetric key used for the encryption. This manifest is the "form" that is uploaded to a server, with the encrypted form contents and its encrypted attachments appearing as attachments to this submission manifest "form."

### 28.2 Security Concerns

While Collect attempts to remove all unencrypted copies of a finalized form and its attachments from the device, because Collect uses third-party applications for image capture, etc., and because of the potential for Forced Close events during the clean-up process, we cannot guarantee that all copies will have been destroyed. Furthermore, because of the way an SD card writes and deletes information, there is a possibility of this data being recoverable from the free space on the SD card. Your organization should investigate the extra steps needed to ensure all data is deleted from the SD cards on your Collect devices and should establish procedures to periodically wipe and reinstall those devices.

**Note:** Encrypting a form ensures that the finalized form is not readable and is not tampered with. However, there is nothing preventing a malicious adversary from the wholesale replacement of a finalized form with falsified data or the synthesis and submission of extra data — these are not contingencies that encrypted forms seek to address.

### 28.3 Defining an Encrypted Form

In **XLSForm**, form encryption is enabled from the *settings sheet*. Encrypted forms must specify a public_key on this worksheet. See below to learn how to generate a public-private key pair. It is also recommended to set an explicit submission_url to make sure the submission goes to the intended destination. Central, this is the server URL configured in Collect for the App User followed by `/submission`. For Aggregate, this is the url with Aggregate.html replaced by *submission*. 
28.4. Creating RSA Key pair

XLSForm — configuring encryption

<table>
<thead>
<tr>
<th>form_id</th>
<th>version</th>
<th>submission_url</th>
<th>public_key</th>
</tr>
</thead>
<tbody>
<tr>
<td>my_form</td>
<td>2024050301</td>
<td><a href="https://my-server/submission">https://my-server/submission</a></td>
<td>MIIBIjANB...JCwIDAQAB</td>
</tr>
</tbody>
</table>

28.4 Creating RSA Key pair

RSA public-private key pairs are generated using the OpenSSL software package. This is pre-installed on macOS and Linux but needs to be downloaded and installed on Windows.

28.4.1 Install OpenSSL (Windows only)

For Windows, download and run the OpenSSL installer appropriate for your system from OpenSSL for Windows. When it asks whether to copy the DLLs to the Windows system directory or to the /bin directory, choose the /bin directory (either will work, but this will minimize the pollution of the Windows system directory).

28.4.2 Constructing the RSA Key Pair

If you are on Windows, open a PowerShell or command prompt window. Change directories to the /bin directory in the OpenSSL directory.

> cd C:\OpenSSL-Win32\bin

If you are on a Mac, open the terminal. Change directories to your Desktop.

$ cd ~/Desktop

28.4.3 Create a private key

The following command will create a 2048-bit private key and write it to the MyPrivateKey.pem file. This may complain about a missing configuration file. You can ignore this warning. If you are on Windows, run:

> openssl genpkey -out MyPrivateKey.pem -outform PEM -algorithm RSA -pkeyopt rsa_keygen_bits:2048
Warning: On Powershell

Check $env:path to be sure path\OpenSSL-Win64\bin is in there. If it is not, run the following command in Powershell:

```
> $env:path = $env:path + ';path to OpenSSL-Win64\bin'
```

If you are on a Mac, run:

```
$ openssl genrsa -out MyPrivateKey.pem 2048
```

### 28.4.4 Extract a public key

Next, you need to extract the public key for this private key.

Run the following command:

```
openssl rsa -in MyPrivatekey.pem -inform PEM -out MyPublicKey.pem -outform PEM -pubout
```

This may also complain about a missing configuration file. You can ignore this warning.

### 28.4.5 Storing and using the keys

Move the MyPrivateKey.pem file to a secure location. It does not have a password encoding it, so anyone can decrypt your data if they have access to this file. This is the private key file that you will give to Briefcase when decrypting the data.

### 28.4.6 Updating the public_key field in the XLSForm settings worksheet.

Open the MyPublicKey.pem file and copy the resulting very-long string inside —BEGIN/END— lines and paste it into the public_key field in the XLSForm settings worksheet. This very-long string will become the base64RsaPublicKey attribute in the resulting encrypted form definition.

**Note:**

- You need to be especially careful that this is ONLY the public key and not the contents of the original public-private key file (which would also appear to work but provide no security).
28.5 Operations

Tip:
- You can use Notepad (Windows) or TextEdit (Mac) to open MyPublicKey.pem
- Alternatively, you can use the command `less MyPublicKey.pem` to print the contents into the terminal and directly copy/paste from there.

See also:
- For reference, you can check out the tutorial encrypted-XLSForm. It is for viewing purposes only but you can make your own copy to edit it.

28.5 Operations

Operationally, you would add the form definition to the server identified in the `<submission>` tag’s action attribute, and deploy everything using Collect, figure out how you want to implement a periodic SD card wiping protocol for your devices, and you’re done. Submissions will be encrypted when marked as complete. Once the data is on your server, use Briefcase to download the encrypted submissions to your desktop computer, and then specify the private key PEM file when decrypting and generating the CSV files.

Note:
- Central or Aggregate will only hold the encrypted submission with no access to the private key
- Briefcase will emit the CSV with an extra final column that indicates whether the signature of the encrypted file was good or bad. It would be bad if any of the attachments are missing or if there was tampering (other than the wholesale replacement of a submission, which can’t be detected).
CHAPTER TWENTYNINE

SECURITY AND PRIVACY

This document details known security and privacy considerations of the ODK software.

- License
- Communication Channels
- 3rd-party Software
- Websites
- Google Play Store
- ODK Aggregate
  - Communications
  - Google App Engine and other Hosting Services
  - Username Authentication
- ODK Briefcase
- ODK Build
- ODK Collect
- XLSForm Online
- Cross-tool Concerns
  - Encrypted Form Security
  - Identifying Information Transmission and Storage

29.1 License

The ODK software is released under the Apache 2 License.
All other artifacts (for example, the ODK website and this documentation) are released under the Creative Commons Attribution 4.0 International License.

All our installers, programs, source code, and documentation are provided AS-IS with no warranty or conditions, and without any liability obligations. See the license text for details.

### 29.2 Communication Channels

Outside of usage analytics (typically opt-out) and crash reports (typically required), ODK software does not transmit or communicate any information (e.g., survey data) back to ODK’s maintainers. When we do gather data, we default to anonymous or aggregate methods.

The software we have written does not have any mechanisms that might allow us to access or control your devices or systems.

There is always the possibility that hackers can discover and exploit deficiencies or bugs in our software or in 3rd-party libraries to access or control your devices or systems.

### 29.3 3rd-party Software

Our software uses a number of open-source 3rd-party libraries from well-known and/or reputable sources, and a few from obscure sources. We do not vet the security of those software libraries.

Your security staff may want to review the libraries and source code on GitHub.

### 29.4 Websites

Our websites under the getodk.org domain use cookies and log all interactions. We also use web analytics tools (for example, Google Analytics) that may track visitors and their access patterns on our web properties.

### 29.5 Google Play Store

Downloads from the Google Play Store are compiled into aggregated usage statistics.

Crash reports you elect to send are provided to us anonymously. By design, these do not contain device or user specific data.
29.6 ODK Aggregate

29.6.1 Communications

When setting up your own web server to run ODK Aggregate, if you do not configure the server and ODK Aggregate to use an SSL certificate, a determined observer can see all data communicated to and from that server.

_Encrypted Form Security_ can prevent a determined observer from viewing encrypted form data being sent via insecure HTTP, but it cannot prevent the observer from silently removing encryption from form definitions and capturing any unencrypted form data that is then submitted.

Only transmissions over a secure HTTPS connection are obscured from observers and prevent tampering in transmission.

29.6.2 Google App Engine and other Hosting Services

With all 3rd party hosting services, you should expect your data to be viewable by the support staff of the hosting service. Different services go to differing lengths to restrict access to, encrypt, and/or secure the data and communications within their data centers.

_Encrypted Form Security_ can prevent hosting services from viewing encrypted form data at rest, but cannot prevent the hosted service from silently removing encryption from form definitions and viewing any unencrypted form data that is then submitted.

See aggregate-deployment-planning for other considerations.

29.6.3 Username Authentication

When authenticating ODK Aggregate usernames and passwords, the ODK tools use DigestAuth. This enables secure username/password authentication even while communicating with servers over HTTP. When using DigestAuth, the password is not sent over the network.

An encoded form of the username’s password is stored on the server. If that encoded value is stolen or revealed, it can allow others to log in and interact with the server as that user.

29.7 ODK Briefcase

We gather aggregate user behavior through Google Analytics and gather crash logs through Sentry. We use secure HTTPS communication to transfer this data to ODK’s maintainers. Users may disable analytics in the settings of the application. Crash logging cannot be disabled.
29.8 ODK Build

We require secure HTTPS connections to ODK Build. We gather aggregate user behavior through Google Analytics. We use secure HTTPS communication to transfer this data to ODK’s maintainers.

29.9 ODK Collect

We gather aggregate user behavior through Google Analytics and gather crash logs through Google Firebase Crashlytics. We use secure HTTPS communication to transfer this data to ODK’s maintainers. Users may disable analytics in the settings of the application. Crash logging cannot be disabled.

29.10 XLSForm Online

We require secure HTTPS connections to XLSForm Online. We gather aggregate user behavior through Google Analytics. We use secure HTTPS communication to transfer this data to ODK’s maintainers.

XLSForm Online stores both your submitted XLS and the generated XML form for a period of time on its disk drive before being deleted. This is necessary for the operation of the tool.

29.11 Cross-tool Concerns

29.11.1 Encrypted Form Security

The form definition and associated media files of an ODK encrypted form are stored on the server in plaintext (unencrypted). The form definition and media files are transmitted as plaintext (but perhaps through a secure HTTPS connection) to client devices (e.g., an Android phone running ODK Collect) and stored in plaintext.

All form data (e.g., incomplete forms, saved forms) and media files are stored in plaintext on the client device until they are finalized. It is only once the form data is finalized that those files are encrypted.

At the time form data and media attachments are finalized, a random 256-bit encryption/decryption key is generated for that form data using the SecureRandom number generator (found here). This ensures that every finalized form has its own unique 256-bit encryption/decryption key.
29.11. Cross-tool Concerns

The form data and media attachments are then encrypted with that key using 256-bit AES Cipher Feedback (CFB) streaming-block encryption. Once encrypted, all plaintext form data and attachments that were used in that process are deleted.

The random key is then padded and encrypted using the RSA public key declared in the form definition (recommended to be 2048-bit) and the OAEPWithSHA256AndMGF1Padding algorithm. The resulting encrypted key is transmitted to the server along with the encrypted data and encrypted attachments. This submission includes a signature field that enables the software to detect tampering to any of the encrypted attachments or to the encrypted form data.

On the device, copies of the deleted plaintext form data and attachments may remain in the free-list of the SD card until they are overwritten with new content.

On the server, if an observer were able to access your encrypted form data, since each form submission uses a different key, each submission would need to be cracked separately.

The secret key required for decryption is never stored on the server, thereby preventing anyone from seeing your form data and attachments unless they break the encryption.

Currently, cracking AES encryption is viewed as impossible for all but the most advanced governmental agencies (for example, the NSA).

29.11.2 Identifying Information Transmission and Storage

During data submission, some identifying information is transmitted and stored on the server:

- ODK Collect passes the deviceID of the device to the server during the submission process. The HEAD request that initiates the submission is a URL of the form: ... /submission?deviceID=imei%3A9117DD011813771. The ODK Aggregate server does not store this deviceID in any database tables, but it will generally be emitted into the webserver access log. This deviceID uniquely identifies the device from which the data is submitted. This can be useful when correlating events on the server with interactions from specific devices. Because this is logged, it is likely that a submission can be correlated with a device, and therefore a data collector.

- If ODK Aggregate is configured to require authentication for submission (that is, if the Data Collector permission is NOT granted to the anonymousUser), then the username that authenticated is written into the audit fields of the data tables storing the submission. If the anonymousUser is granted Data Collector privileges, no authentication is performed, and anonymousUser is written into those fields. The content of these audit fields is not exposed in exported CSV files, ODK Briefcase data pulls, or published to downstream systems. However, because it is present in the database tables, you can definitely correlate this authenticated username with the submitted data.

While interacting with an ODK Aggregate website, any actions that require authentication and that modify the server settings, set of form definitions, filters, exports, publishers, or data tables, will cause the authenticated username to be written into the audit fields of the
database tables that are being updated. If these modifications result in delete actions being performed against a database table, then this authenticated username will be identified in the server log together with summary information on what was deleted.

See also:

GLOSSARY

Aggregate  Part of ODK.
     A server-side data storage and analysis tool.

Briefcase  Part of ODK.
     A tool for packaging and transferring forms and data between instances of Collect and Aggregate.

Build  Part of ODK.
     An application that lets you design forms with a drag-and-drop form interface.
     • Use ODK Build online.
     • Download a desktop version of ODK Build.

Collect  Part of ODK.
     An Android mobile app that replaces paper-based forms.

CSV  Comma Separated Values. A plain-text file format for tabular (spreadsheet-like) data.

enumerator  A person who conducts a survey.

form  A defined set of questions and answer choices displayed by an application that can render forms written in the XForm standard.

hint  Additional help text on a single question, displayed after the label.

instance  The file representing a filled-in form.

JSON  JavaScript Object Notation. A serialized key-value data format.

KML  Keyhole Markup Language. A file format for geographic data.

ODK JavaRosa  Part of ODK.
     A Java library that renders ODK Compliant XForms.

ODK JavaRosa source on Github
ODK XForm  Part of ODK.

A specification defining valid XML-based forms for ODK. It is a subset of the W3C XForms 1.0 specification

[View ODK XForms Specification]

participant  A person being interviewed by a user of Collect. (Also sometimes called a "subject").

question label

question text  The main body of a form question or widget. In an XLSForm, the contents of the label.

Validate  Part of ODK.

A tool for validating forms against the ODK XForms specification.

widget  A single question, answer set, and attendant GUI elements, as rendered in a XForm compliant app such as Collect.

XLSForm  Part of ODK.

A tool for building forms with Microsoft Excel.