Open Data Kit Documentation

Release 1.0

Open Data Kit

Mar 09, 2018

CONTENTS

1 Getting Started With ODK 3

2 ODK Collect 7

3 Setting Up ODK Collect 9

4 Using ODK Collect 21

5 Tips and Best Practices 77

6 ODK Aggregate 143

7 Setting Up ODK Aggregate 145

8 Using ODK Aggregate 223

9 Tips and Best Practices 261

10 ODK Briefcase 269

11 Setting Up ODK Briefcase 271

12 Using ODK Briefcase 273

13 Briefcase and Aggregate 279

14 Intro to Forms in ODK 281

15 ODK Build 283

16 XLSForm 295

17 Form Widgets 297

18 Interaction of constraints, relevants and read only 409

19 Using regular expressions 415

20 Updating forms 419

21 pyxform 427

22 ODK Form Validate 429

23 Launching External Apps from Collect Forms 433

24 Contributing to ODK Docs 437

25 OpenRosa 515

26 ODK JavaRosa 541

27 Launching ODK Collect from External Apps 543

28 Briefcase Aggregate API 547

29 Encrypted Forms 553

30 Security and Privacy 561

31 Verifying Downloads 567

32 FAQ 573

33 Glossary 579

Open Data Kit (ODK) is a suite of open source applications that help organizations engaged in enumerator-mediated data collection. ODK tools assist with the collection and management of survey data using digital forms. These include the primary ODK applications:

- *ODK Collect*, an Android mobile app that replaces paper-based forms.
- *ODK Aggregate*, a server-side data storage and analysis tool.

Also part of the ODK suite are several tools that support form creation and data management:

- **ODK Build** lets you design forms with a drag-and-drop form interface.
- **ODK Validate** validates forms against the ODK XForms specification.
- **ODK Form Uploader** uploads blank forms and their media files to ODK Aggregate.
- **ODK Briefcase** packages and transfers data between instances of Collect and Aggregate.

ODK also maintains libraries and specifications that support these applications.

- **ODK XForm** is a subset of the W3 XForm specification, for use in the ODK ecosystem.
- **ODK JavaRosa** is a Java library that renders forms complying with the ODK XForm specification.
- **XLSForm** defines an Excel format for designing forms.

For a complete list of our projects, check out Open Data Kit on Github.

CHAPTER

ONE

GETTING STARTED WITH ODK

1.1 Using ODK

In most cases, users of ODK:

- Create survey forms using Build or XLSForm
- Upload forms to an Aggregate server
- Load forms into Collect on an Android device
- Use Collect to fill out forms with *participants*
- Upload survey data from Collect to Aggregate
- Analyze or export data in Aggregate

This requires:

- *Installing Collect on a phone or other mobile device*
- *Installing Aggregate on a server*

Tip: While this is the *typical* use pattern, it is not the only way to do things. For example:

- Forms can be created using tools other than Build or XLSForm
- Forms can be transfered directly to Collect, without using Aggregate

ODK is a very flexible set of tools, and organizations will find their own best practices for adopting it.

1.2 Install Collect

The easiest way to install the ODK Collect App is to get it from the Google Play store.

For other installation options, see *Installing Collect*.

1.3 Install Aggregate

The easiest, recommended way to setup an ODK Aggregate instance is to use Google App Engine and the ODK Aggregate Installer.

You'll set up a new Google Cloud project, and then run the install utility locally. This will connect to your Google Cloud account and install Aggregate there.

For full details, and other installation methods, see the *ODK Aggregate Installation and Setup Guide*.

You can also watch this video which explains the installation procedure of ODK Aggregate on Google App Engine and also this Aggregate tutorial video which describes few of its functionalities.

1.4 Create and Upload Survey Forms with ODK Build

The quickest and easiest way to start using your own survey forms is to create them online with ODK Build. To create a form, you can follow the steps below or watch this tutorial which explains the creation process.

- Go to build.opendatakit.org, create a new account, and log in.
- Once logged in, a blank survey is created. Give it a name (*rename* in the upper left-hand corner) and add a few questions (click on question types in the *+Add New* bar along the bottom).
- Once your new form is complete, go to *File* → *Upload form to Aggregate…* to upload your form.

Tip:

- ODK Build is a great tool for simple forms. For more complex forms, try XLSForm.
- ODK Build can also be run locally. Desktop versions are available for download here.

1.5 Load, Complete, and Upload a Form with ODK Collect

- *Install* and open ODK Collect on your Android device.
- Open the menu and then select *General Settings*.

- Select *Server*.
- Edit the server settings to connect to your ODK Aggregate instance.
- Go back to the app home screen and select *Get Blank Form*, then select your form.
- Select *Fill Blank Form* to complete a survey.
- Select *Send Finalized Form* to upload your completed survey to ODK Aggregate.

Now, you should be able to log back into your ODK Aggregate instance and see your completed survey results.

CHAPTER

TWO

ODK COLLECT

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 geolocation, 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.*

2.1 Learn more about ODK Collect

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

CHAPTER

THREE

SETTING UP ODK COLLECT

3.1 Installing Collect

3.1.1 Installing from Google Play Store (Recommended)

The ODK Collect App 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://opendatakit.org/downloads/download-category/collect/ 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.

Note: You can also download the ODK Collect APK to your computer and load it on your device via adb or another tool like AirDroid.

Note: On older Android devices (4.0 and earlier), ODK Collect required an external SD Card. This is no longer an issue because Android devices have internal storage. Virtually all current Android devices will run ODK Collect.

Tip: You can also install ODK Collect on an Android emulator. However, this can be slow and is only recommended for developers actively working on Collect.

3.2 Connecting to a Server

ODK Collect is used to complete surveys with participants. But if you want to review and analyze your collected data, you'll need to upload your collected data to another application.

Note: When you first install Collect, it connects to the ODK Aggregate Demo server. This allows you to try out the app by *downloading blank example forms*, *filling them out*, and *uploading completed forms* back to the demo server.

Once you are done "trying out" Collect, and you start actually using it, you will need to decide on a plan for managing forms and data submissions.

The two most common options for form and data management are:

3.2.1 Connecting to your own ODK Aggregate Server

Before you get started...

You need to *install ODK Aggregate* before you can connect to it.

When connecting to an ODK Aggregate server from Collect, you will need a username and password from your Aggregate server. This can be the superuser created when you install Aggregate, or another user account created by you.

- See *Installing Aggregate* to setup your ODK Aggregate server.
- See *Aggregate Site Admin* for details on setting up new users.

1. From the Action Button ( ), select *General Settings*

2. Select *Server*

3. Select *Type*, and set it to *ODK Aggregate*

4. Edit *ODK Aggregate settings*

Connecting to another OpenRosa server app

Any server application that implements the OpenRosa API can be connected to, using *the instructions laid out above*. You would still choose *ODK Aggregate* as the server type.

In *Server Settings*, there is currently a *Type* option of *Other*. It is unlikely you will need this option, since any server application will need to implement the same API as ODK Aggregate.

If you think you might need to connect to a non-Aggregate server application, and are having trouble, we encourage you to visit our Support Forum.

The *Other* option will likely be deprecated in the future, and its use is not recommended.

3.2.2 Connecting to a Google Drive Account

1. From the Action Button ( ), select *General Settings*

2. Select *Server*

3. Select *Type*, and set it to *Google Drive, Google Sheets*

4. Select your *Google Account*. You can select any account which is already linked with your device or add a new account as well.

5. **Optional:** Set a *Fallback submission URL*

   When using Collect with a Google account, form submissions will be posted to a Google Sheet specified in the form.

   You have the option to specify a *Fallback submisison URL*. This is the URL of a Google sheet to which form submissions will be posted if the submitted form does not specify its own URL.

   If your forms will specify a submission URL, you can leave this setting empty. Otherwise, enter the URL of a Google sheet you would like to use.

ODK Aggregate provides a robust data repository with tools for data visualization, querying, and export. For complex data collection and aggregation tasks, it is usually the right choice. However, setting up and maintaining an Aggregate server is not a trivial matter.

Using Google Drive to manage form submissions is simpler and, in most cases, cheaper. With this simplicity you sacrifice a richer feature set. Additionally, using Google Drive may not meet your privacy requirements.

3.2.3 Other options

Managing forms from an ODK Aggregate server or Google Drive is typical. However, there are other ways to use Collect.

- *Upload blank forms directly to your device*
- *Download completed forms directly with adb*
- *Use ODK Briefcase*

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 Open Data Kit, *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*.

To use ODK Collect:

1. *Get blank forms* from *ODK Aggregate Server or Google Drive*
2. *Fill out surveys* with participants
3. *Upload completed surveys* to Aggregate or Google Drive

ODK Collect User Guide

4.1 Managing Forms in Collect

See also:

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

4.1.1 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 ODK Aggregate Server or Google Drive

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

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 ODK Aggregate, you will see a list of available forms. Select the ones you would like download, and tap *Get Selected*.

Note: Before downloading blank forms from Aggregate to Collect, a form has to be uploaded. See *Form Management in ODK Aggregate*.

- 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.

> **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**.

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 via USB, using *Android Debug Bridge*.

```
$ adb push path/to/form.xml /sdcard/odk/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 `odk/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.

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 should be placed in a folder labeled `form-name-media`.

- When using ODK Aggregate, the form upload prompt includes instructions to upload the `-media` folder. The files are downloaded automatically when *fetching forms from Aggregate*.
- When using Google Drive, the `-media` folder should be uploaded to the same location as the form.
- If *loading forms directly to your device*, the `-media` folder needs to be placed in the `sdcard/odk/forms` directory, alongside the form itself.

4.1.2 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.

4.1.3 Uploading Finalized Forms

Eventually, you will want to upload completed form instances for analysis and data aggregation.

Doing this from within the Collect app marks the forms as sent. Sent forms are no longer editable, but they remain viewable until they are deleted.

Uploading to Aggregate or Google Drive

If you are connected to *an ODK Aggregate server* or *Google Drive Account*, use *Send Finalized Forms* to upload Finalized form instances.

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.

Pulling forms with Briefcase

For local form management, use *ODK Briefcase* to pull Finalized form instances to your local computer.

Pulling forms with `adb`

You can copy form instances from the device using **`adb pull`**, however this will not update the state of the form to Sent.

4.1.4 Deleting Forms

You can delete Saved, Finalized, Sent, and Blank forms.

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

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.

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 `sdcard/odk/instances`, with a directory for each instance.

4.2 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

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*.

4.2.1 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).

Fig. 4.5: Radio buttons accept one selection.

Fig. 4.6: Check boxes accept multiple answers.

Fig. 4.7: Select images by touching them.

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, geolocation widgets, and barcodes.

In all cases, buttons below the question text will guide you through providing the response.

3:29
All widgets
Image widgets
Signature widget
image type with signature appearance
Gather Signature

3:29
All widgets
Media widgets
Video widget
video type with no appearance
Record Video
Choose Video
Play Video

See also:

For a (mostly) complete guide to form question appearance, see *Form Widgets*.

4.2.2 Removing answers

To remove a response, Long Press on the *question label*.

4.2.3 Navigating the form

Swipe

To move between questions, Swipe Left or Right.

Video showing swiping between three questions.

Left and Right Buttons

If you prefer Forward and Back buttons for navigation, you can switch to that in -> *General Settings -> User Interface*.

1. Open the *Action Menu* ( )

2. Select *General 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.

Note: If a form contains questions in a looped group, those questions will only appear in

the Jump menu once an actual record is created.

Autoadvance Questions

Some questions will *auto-advance* after being answered.

Video showing auto-advance after the questions are answered.

Required Questions

Required questions will not allow you to advance unless answered.

4.2.4 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.

4.2.5 Saving a partially filled form

If you wish to save a partially filled form, you can click on the save icon ( ) 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.

4.2.6 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).

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*.

4.3 Collect Menus, Settings, and Security

4.3.1 Main Menu

Fill Blank Form Lists available blank forms and lets you select a form to begin filling out.

Edit Saved Form Lists completed and saved forms and lets you select a form to edit.

Send Finalized Form Lists finalized but unsent forms and lets you select forms to send to the server.

View Sent Form Lists forms that have been sent, even if they were deleted.

Get Blank form Lists blank forms available on the server and lets you download them.

Delete Saved Form Lists all the Saved and Blank Forms and lets you delete them.

4.3.2 General Settings

To access General Settings: → *General Settings*

Server Settings

Server settings *control the connection to* an *Aggregate* or *OpenRosa* server or a *Google Drive account*.

To access Server Settings: → *General Settings* → *Server*

See also:

Connecting to a Server

User Interface Settings

User Interface settings control Collect's appearance and behavior.

To access User Interface settings: → *General Settings* → *User Interface*

Language Sets the display language.

Text font size Sets the display font size.

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.

Mapping

Mapping SDK Sets the mapping engine that will be used for form question types that require map integration.

Options:

- Google Maps (default)
- OpenStreetMap

See also:

Geopoint, *GeoShape*, *GeoTrace*, *offline maps*

Basemap Sets the map to be displayed when a question with a mapping component is opened.

Form Management Settings

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

To access Form Management settings: → *General Settings* → *Form Management*

Form submission

Auto send When enabled, forms are sent immediately when they are finalized, if the device can connect to the internet. 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, forms are automatically finalized upon reaching the end of the form. You can opt out of this on any specific form during form completion.

Constraint processing Sets when form responses are validated against constraints.

Options:

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

High res video When enabled, *Video* 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.

Form import

Import saved forms as finalized When enabled, forms added directly to the `instances/` directory are automatically set to Finalized.

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: → *General Settings* → *User and device identity*

Form metadata settings

Form metadata settings control how identifying information is added to the metadata of forms completed on the device.

To access form metadata settings: → *General Settings* → *User and Device Identity* → *Form Metadata*

User-defined

You can edit the following:

- Username
- Phone number
- Email address

Note:

- If no username is set here, the username from *Server settings* is used instead.
- You can restrict editing of the username in *admin settings*.

Device-defined

You cannot edit these:

- Device ID
- Subscriber ID

- SIM serial number

Usage data

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

4.3.3 Admin Settings

Admin 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*.

Admin settings are useful when you are managing devices that will be used by many enumerators, and you would like to limit the options available to those enumerators.

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

To access Admin settings: → *Admin Settings*

General Settings Provides access to *General Settings*, with all items unrestricted.

Admin Password Lets you password protect this screen.

Reset application Lets you reset to default settings, delete forms, and empty caches.

Import/Export settings See:

Importing and Exporting Settings

QR Code

QR Code or the Quick Response Code is a two dimensional barcode. QR codes can be used to configure Collect on many devices. The QR image presented contains all of your current General and Admin settings, including admin and server passwords. When this QR code is scanned from the ODK Collect app on another device, these settings are imported.

Sharing QR code

You can click 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* to name a few, 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 may contain the admin and server passwords without encryption, you should be careful about how you share it. It is advised to not send it through an external application but through *bluetooth*, *MMS* or any other such service that doesn't allow the third party to access the data.

Saving QR code locally

You can go to → *Save settings to disk* to save the QR code.

Note: Settings are exported and saved so that they can be loaded to other devices. When you save external settings, they are saved to `/sdcard/odk/settings/collect.settings`. You can load external settings from this location to other devices as well.

At app launch, the settings are loaded from a different location `/sdcard/odk/collect.settings`, which tracks the state of your settings at all times.

Settings are saved to and loaded from different locations to avoid deleting the saved settings.

Importing settings from a QR saved on your device

You can import settings from a QR code saved on your device by clicking on *Select Code from SD Card* option.

Making your own QR code

QR code is a JSON object with a structure as shown below:

```
{
  "general": {
    "protocol": "google_sheets",
    "constraint_behavior": "on_finalize"
  },
  "admin": {
    "edit_saved": false
```

```
  }
}
```

The JSON object isn't encrypted but is compressed using zlib before encoding into QRCode. Therefore the creation process can be summarized as follows:

1. Write a JSON object containing the changed settings with a structure as shown above.
2. Compress it using zlib.
3. Encode into QR code.

After you finish generating the QR code, you can transfer it to your device and then import it by clicking on *Select Code from SD Card* option.

List of keys for all settings

Following is the list of keys for all settings and the set of values they can take:

```
{
  "admin" : {

    // Stores the admin password
    "admin_password": Boolean,
    "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,

    // User access control to form entry
    "save_mid": Boolean,
    "jump_to": Boolean,
    "change_language": Boolean,
    "access_settings": Boolean,
    "save_as": Boolean,
    "mark_as_finalized": Boolean,

    // User access control settings for General settings
    "change_autosend": Boolean,
    "change_navigation": Boolean,
    "change_constraint_behavior": Boolean,
    "change_font_size": Boolean,
```

```
    "change_app_language": Boolean,
    "instance_form_sync": Boolean,
    "default_to_finalized": Boolean,
    "delete_after_send": Boolean,
    "high_resolution": Boolean,
    "image_size": Boolean,
    "show_splash_screen": Boolean,
    "show_map_sdk": Boolean,
    "show_map_basemap": Boolean,
    "analytics" : Boolean,
    "change_form_metadata": Boolean,
    "change_server": Boolean,

    // Server
    "import_settings": Boolean,
    "change_server": Boolean,
    "change_protocol_settings": Boolean,

    },

  "general" : {

    // Server settings
    "protocol": {"odk_default", "google_sheets", "other"},
    // Aggregate specific key
    "server_url": String,
    // Google sheets specific keys
    "selected_google_account": String,
    "google_sheets_url": String,
    "username": String,
    "password": String,
    // Other specific keys
    "formlist_url": String,
    "submission_url": String,

    // User interface
    "app_language": { "en", "af", "am", "ar", "bn", "ca", "cs", "de", "es",
↪"km",
         "et", "fa", "fi", "fr", "ha", "hi", "hu", "in", "it", "ja", "ka",
↪"zu",
         "lt", "mg", "mr", "my", "ml", "nb", "nl", "no", "pl", "ps", "pt",
↪"ro",
         "ru", "so","sq", "sw", "ta", "ti", "tl", "tr", "uk", "ur", "vi",
↪"zh"
         },
    "font_size": {13, 17, 21, 25, 29},
    "navigation": {"swipe" ,"buttons" ,"swipe_buttons"},
```

```
    "showSplash": Boolean,
    "splashPath": String, // If showSplash is true, then you specify the␣
↪path of image here.
    "map_sdk_behavior": {"google_maps", "osmdroid"},
    // if map_sdk_behavior is google_maps, then map_basemap_behavior can␣
↪take the following values:
    "map_basemap_behavior": {"streets", "satellite", "terrain", "hybrid"},
    // if map_sdk_behavior is osmdroid, then map_basemap_behavior can take␣
↪the following values:
    "map_basemap_behavior": { "openmap_streets", "openmap_usgs_topo",
                              "openmap_usgs_sat", "openmap_stamen_terrain",
                              "openmap_cartodb_positron", "openmap_cartodb_
↪darkmatter"
                            },

    // Form submission
    "delete_send": Boolean,
    "autosend": Boolean,
    "autosend_wifi": Boolean,
    "autosend_network": Boolean,

    // Form filling
    "constraint_behavior": {"on_swipe", "on_finalize"},
    "default_to_finalized": Boolean,
    "high_resolution": Boolean,
    "image_size": {"original", "small", "very_small", "medium", "large"},

    // Form import
    "instance_sync": Boolean,

    // User and Device identity
    "form_metadata": String,
    "metadata_migrated": Boolean,
    "metadata_username": String,
    "metadata_phonenumber": String,
    "metadata_email": String,
    "analytics": Boolean, // Anonymous usage data

  },

}
```

Note: The subkeys in the general key can be a part of the admin key too.

Note: QR code only contains settings whose values are not the default values because of the constraints on the amount of data a QR code can hold.

Warning: The QR code used for settings-import contains the admin and server passwords *in plain text*. To remove them from the code, tap the warning on the QR code screen.

User Access Control Settings

Main Menu Settings Displays a list of *Main Menu* features. To hide features, uncheck them.

User Settings Displays a list of user settings and other features accessible in the *General 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

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.

4.4 Using Offline Maps

Collect's *GeoShape*, *GeoTrace* and *Geopoint* question types can be configured to display different maps. To use online maps, *set the mapping engine and basemap* in *User Interface Settings*. Users will need to be online to load questions with maps.

Offline maps are useful for low-connectivity environments or to present custom geospatial data. 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. Tiles are images that represent a subset of a map. The only limitation is that tile data in Mapbox's pbf format are not supported.

4.4.1 Offline maps quick start

1. From *Collect's settings*, change the Mapping SDK to Google Maps SDK

2. *Get or create your MBTiles file* with TileMill or other software.
3. *Transfer tiles to devices*. The MBTiles file must be placed in a sub-folder on your device in the `/sdcard/odk/layers` folder.
4. Open a question that displays a map (types *GeoShape*, *GeoTrace*, *Geopoint* with certain appearances)
5. Tap the layers icon and select your map

4.4.2 Getting map tiles

Open Map Tiles hosts many free map tile files that can be used in ODK Collect.

To create MBTiles files, use one of the compatible applications that can write tiles. Commonly used free software packages are TileMill and QGIS with the QTiles plugin.

4.4.3 Transferring offline tiles to devices

To make MBTiles files available for use in ODK Collect they must be manually transferred to Android devices. The tile files need to be inside a folder in the `/sdcard/odk/layers` folder. The folder name will be used to identify your offline map in the user interface so choose a user-friendly name for it. For example, if your MBTiles file is *my_tiles.mbtiles* and it includes information about settlements in the mapped area, you may want to call your folder `/sdcard/odk/layers/All settlements/`.

Note: MBTiles files placed directly in the `/sdcard/odk/layers` folder will not be detected! Placing it in a subdirectory with a friendly name is required.

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*.

- 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*

CHAPTER

FIVE

TIPS AND BEST PRACTICES

5.1 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.

5.1.1 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.

5.1.2 Managing forms with ADB

Forms are stored on the device in `/sdcard/odk/forms/`.

Loading blank forms

```
$ adb push local/path/to/form.xml /sdcard/odk/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 /sdcard/odk/forms/form.xml
```

Downloading forms to your computer

To download a completed form or form instance from the computer:

```
$ adb pull /sdcard/odk/forms/form.xml
```

5.1.3 Developer tasks and troubleshooting with ADB

Downloading Collect databases

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

```
$  adb pull /sdcard/odk/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.

Recording video

```
$ 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:

```
$ 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:

   ```
   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 Collect issue on GitHub 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.

```
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:

https://developer.android.com/studio/debug/bug-report.html

5.2 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.

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.

5.3 Improving Location Performance

When you need to specify location in any survey form, you may get a warning message which says: **Sorry, location providers are disabled** and a screen will open asking you to enable location providers.

example_locat...
Location
Start GeoPoint

Location
Mode
Improve accuracy
Recent location requests
Google
Google Play services
Maps

5.3.1 Enable/Disable location providers

To turn your device's location providers on/off:

1. Open your device's Settings app.

2. Tap *Location*.

Tip: If you don't see "Location", tap *Security & Location* and then *Location*.

3. Turn Location on or off.

5.3.2 Change location mode

You can also choose your location mode based on accuracy, speed, and battery use. Android can help improve location accuracy by triangulating your location off nearby Wi-Fi access points or cell towers. Combined with data available from your ailing GPS signal, turning on Wi-Fi will greatly improve location accuracy under most conditions.

In the Location section, tap on *Mode*.

Fig. 5.1: Location providers disabled.

Fig. 5.2: Location providers enabled.

Then pick:

- **High accuracy**: This mode uses GPS, Wi-Fi, mobile networks, and other sensors to get the highest-accuracy location. It uses Google's Location service to help estimate your device's location faster and more accurately.
- **Power saving** or **Battery saving**: This mode uses sources that use less battery, like Wi-Fi and mobile networks. It uses Google's Location service to help estimate your device's location faster and more accurately.
- **GPS only** or **Device only**: This mode uses only GPS. It doesn't use Google's Location service to provide location information. It can estimate your device's location slower and use more battery.

5.3.3 Improve location accuracy

You can also improve accuracy of location.

Tap on *Improve accuracy* in Location section.

Then enable:

- **Wi-Fi scanning**: Allow apps and services to scan for Wi-Fi networks automatically, even when Wi-Fi is turned off.
- **Bluetooth scanning**: Allow apps and services to scan for and connect to nearby devices automatically via Bluetooth, even when Bluetooth is turned off.

Note: Google has added Bluetooth scanning support to further improve location accuracy with Android 6.0 Marshmallow.

For more details on location services, see this guide.

5.4 Android security recommendations

5.4.1 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:

- 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.

Settings
NFC and sharing
More networks
Device
Sounds and notifications
Display and wallpaper
Lock screen
Themes
Multi window
Notification panel
Motions and gestures

Lock screen
Screen security
Screen lock
Secured with pattern
Show information
Pattern type
Camera shortcut
Make pattern visible
Vibration feedback

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.

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.

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.

5.4.2 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.

5.4.3 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.

2. Then tap *Backup & Reset* in *Personalisation* section.

3. Now switch off the option to *Back up my data*.

5.4.4 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.*

5.4.5 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.

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.

Lock screen
Secured with pattern
Show information
Pattern type
Camera shortcut
Make pattern visible
Vibration feedback
Lock automatically
Lock instantly with power key

12:47 p.m.
Lock automatically
5 seconds
15 seconds
30 seconds
1 minute
2 minutes
5 minutes
10 minutes
30 minutes
CANCEL

5.4.6 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.

Sounds and notifications
Vibrate when ringing
Ringtone
Vibrations
Other sounds
Notifications
Do not disturb
Notifications on lock screen
Application notifications
Notification access

4:22 a.m.
Sounds and notifications
Vibrate when ringing
Ringtone
Notifications on lock screen
Show content
Hide content
Do not show notifications
CANCEL
Application notifications
Notification access
1 application is allowed to read notifications.

5.4.7 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.

5.4.8 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.

Settings
System
Language and input
Date and time
Safety assistance
Accessories
Battery
Storage
Security
Help
About device

4:44 a.m.
About device
Download updates manually
Last checked on: 04/09/2017
Downloading via mobile networks may result in additional charges. If possible, download via a Wi-Fi network instead.
Download updates automatically
Check for firmware updates automatically. Updates will be downloaded automatically only when the device is connected to a Wi-Fi network.
Scheduled software updates
Off
Status
Show status of battery, network, and other information.
Legal information
Report diagnostic info
Device name
Galaxy A5

5.5 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:

5.5.1 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.

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 *RELAUCH*

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*.

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.

5.5.2 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*.

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*.

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*.

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...*

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

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*.

14 Now you would be able to see your virtual device in Android Virtual Device Manager. Click on 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 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:

```
$ 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.

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\
```

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
```

On Windows:

```
> where emulator
```

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

```
$ 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.

   ```
   $ 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.

   ```
   $ 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 sys-img tag to use for the AVD.
   - `-k` specifies package path of the system image for the AVD.

See also:

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.

5.6 Related Topics

- *Intro to Forms in Open Data Kit*
  - *Example form questions in Collect*
- *Encrypted Forms*
- Integrating with other Android Apps
  - *Launching ODK Collect from External Apps*
  - *Launching External Apps from Collect Forms*

CHAPTER

SIX

ODK AGGREGATE

ODK Aggregate is an open source Java application that stores, analyzes, and presents *XForm* survey data collected using *ODK Collect* or other *OpenRosa-compliant applications*. It supports a wide range of data types, and is designed to work well in any hosting environment.

With Aggregate, data collection teams can:

- *Host* blank *XForms* used by ODK Collect or other OpenRosa clients
- *Store and manage XForm submission data*
- Visualize collected data using *maps* and *simple graphs*
- *Export and publish data in a variety of formats*

Aggregate can be hosted on *Google App Engine*, *Amazon Web Services*, or *your own local or cloud server*. There's also a *pre-configured virtual machine image* that is ready to deploy on any computer.

6.1 Learn more about ODK Aggregate

- *Setting Up ODK Aggregate*
- *Using ODK Aggregate*
- *Getting Started With ODK*

CHAPTER

SEVEN

SETTING UP ODK AGGREGATE

Before you get started

If you just want to explore ODK Aggregate, try the demo server.

7.1 Planning Your Aggregate Deployment

ODK Aggregate can be deployed to:

- *Google App Engine*
- *Amazon's EC2 cloud services*
- *any web server running Tomcat with MySQL or PostgreSQL*.

There is also a fully set-up *virtual machine* that can be run in nearly any environment.

We recommend using Google App Engine or the ODK Aggregate VM before attempting an EC2 or Tomcat deployment. Once you have tried Aggregate together with *ODK Collect* and familiarized yourself with their use, you can consider alternative hosting platforms.

You can also go without Aggregate altogether and use *ODK Briefcase*.

This document provides general advise for thinking through your deployment decisions.

7.1.1 Things to Consider

Internet access

Google App Engine and Amazon Web Services both require internet access. If you don't have consistent internet access, Briefcase may be more appropriate.

Tomcat deployments can operate without internet access. In such an environment, Collect would only be able to upload finalized forms after it connects to the network containing the Tomcat deployment.

Computer skills

Tomcat deployments (including deployment to Amazon Web Services) have a steep learning curve and require technical aptitude. At a minimum you will be:

- changing network configuration
- selecting and using a website hosting service or specifying and configuring your own server and network router(s)
- installing software
- ensuring that your site has proper power-failure and data-backup systems in place

If this level of systems administration skill is not available, you will have more success using Google App Engine.

Ongoing support

Tomcat deployments require periodic backups of your data. If data security is a concern, you should have a system administrator or database administrator periodically review logs and look for malicious activity.

Availability

Google App Engine provides highly available servers and data storage. Tomcat deployments with similar availability will be expensive to operate unless your organization already has its own information technology department. The less downtime you can tolerate, the more expensive a Tomcat deployment will be.

On the other hand, high availability is not an issue for many deployments. Most users of Collect download blank forms once and rarely update those forms over the course of a study. Surveyors upload finalized forms to Aggregate infrequently and opportunistically. If that is your situation, you likely do not need a server that is as highly available as Google App Engine provides.

Dataset size

Google App Engine can store a virtually unlimited amount of data — well in excess of a million submissions.

However, in deployments with data sets exceeding 7,000 submissions, the *data export feature* will stop working. To correct this, you will need a custom deployment with a larger virtual machine. This problem affects both Google App Engine and Tomcat deployments.

On Google App Engine, a larger server will incur higher billing costs. Additionally, for datasets of over 100,000 records, it is likely that performance will be better when using MySQL or PostgreSQL, rather than Google App Engine's data store. You also have more optimization opportunities when running your own database servers than are available through Google's cloud services.

Data locality and security

Google App Engine servers may be located anywhere in the world.

Depending on the sensitivity of the data and specific storage rule, regulations, or restrictions of your country or organization, the server infrastructure may not have all necessary locality guarantees or security precautions.

In some circumstances, you might be able to use *Overview* to achieve compliance. You should research and comply with applicable laws and regulations before storing data on Google App Engine.

See also:

Google Cloud Services Terms of Service.

Billing

Google App Engine has 24-hour activity quotas that typically enable free use of ODK Aggregate during evaluation and small pilot studies.

You may be able to run a full study within these activity thresholds provided you:

- collect fewer than 2000 responses
- access the site only during the work day
- can be flexible about when you upload and access data

Otherwise, you will need to set up a billing account with Google.

Note: These usage limits are well within what is needed to try out ODK, to determine if it would be a good fit for your data collection needs.

Because of this, and its ease of setup, we recommend using Google App Engine in test deployments and pilot studies, even if you will likely need a custom deployment later on.

Open source

The ODK software is free, open source, and available for use without charge.

It is important to recognize that the open source software model does place additional responsibilities on the users of that software.

Unless you pay for assistance when technical support is needed, you will be required to take the initiative to research and find answers, and to perform technical support tasks yourself.

Finally, unless you and others contribute back to Open Data Kit through involvement in the community and contributions to the project, this software will become irrelevant and obsolete.

See also:

Learn more about participating in ODK

7.1.2 Minimizing App Engine fees

On App Engine, the major driver of cost is Datastore Reads. These add up quickly:

- Viewing a page of form submissions incurs *at least* one Read for each submission.
- Each multiple-choice question in a form incurs an additional Read on every displayed submission.
- An additional read is incurred for every 200 questions in your survey.
- Each image incurs at least 10 reads.
- The default view shows 100 submissions.
- The form submissions display refreshes every six seconds.

For example, if your survey has 500 questions (q), with a repeat group containing an additional 300 questions, the typical survey has 4 filled-in repeats (rpt), and 100 submissions (s) are shown on each page load (pl), then the cost to display the Submissions tab is a minimum of 1100 Reads (R) with each refresh of the Submissions tab.

$$100s/pl \times (500q/s \times \lfloor 1R/200q \rfloor + 4rpt/s \times 300q/rpt \times \lfloor 1R/200q \rfloor) = 1100\ R/pl$$

At this rate, the free quota would be exceeded within 5 minutes!

And this hypothetical survey did not contain any select-one or select-multiple questions, or any audio, video or image captures, all of which would require more Reads.

Therefore, to reduce datastore reads:

- *restrict access to the Aggregate website*
- do not keep the browser window open on the submissions tab

- *export or publish your data*, and do your analysis in a different tool
- use *ODK Briefcase* instead of Aggregate to generate CSV files

It is generally more efficient to use Briefcase to generate CSV files than to use Aggregate, as Briefcase will use the locally cached data to generate the CSV files.

With larger datasets, there are two modes of operation:

- Aggregate retains the full dataset.

  In this mode, it is slightly more efficient to Pull data to your local computer then immediately Push it back up. This sets some internal tracking logic within Briefcase so that the next Pull is somewhat more efficient, as the Push only verifies that what you have locally matches the content on Aggregate, rather than re-uploading all of it.

- Aggregate retains only a portion of the dataset.

  In this mode, you periodically purge older data collection records and never Push data up to Aggregate, as that would restore the purged data.

7.2 Installing Aggregate

Tip: *Google App Engine* is the recommended hosting platform, especially for organizations new to ODK Aggregate.

Other installation options are documented here for your information, but we strongly encourage you to use Google App Engine unless you have a compelling reason to do otherwise.

7.2.1 Installing on Google App Engine

Note: Google App Engine is the recommended deployment platform for Aggregate.

There is no charge to set up an Aggregate server on App Engine, and lightly used instances will usually incur no charges.

Before you begin...

- Make sure Java 8 or higher is installed on your system.
- You will need a Gmail account to use App Engine. This Gmail account will be the owner of the Google Cloud Platform project under which your App Engine will execute.
- Read the Google Cloud Platform term of service.

- Download the ODK Aggregate installer. Select the latest release for your operating system.
  - You can *verify the download* using *SHA256 signatures*.

1. Go to Google Cloud Platform and click on *Console* in the top right corner.

2. Sign in with the Gmail/Google account you wish to use.

3. If you have never configured a Google Cloud Platform project, click on *Create an Empty Project*.

4. If you have configured a Google Cloud Platform project before, you may see a listing of all your projects and a button labeled *Create Project*. Click that button.

   If instead you see the console for one of your existing projects, click on the project name at the top of the window. In the example image below, the older project is *Project-edu*.

- Upon clicking the project name, a window appears with a + symbol. Click on it to create a new project.

5. Name your project.

   On the project-creation popup dialog, type in a project name that makes sense to you. If you want to edit the project ID, which will be a part of your project URI, click the small *Edit* link. When you are done, click *Create*.

Tip: You may want to use a project id that combines your organization name and the name of your data collection group or project.

6. After few seconds, you will see a notification in the top right corner of the window. Click on the notification icon and select the notification message **Create Project: project-name**. An empty project screen will open.

7. Click on the menu icon ( ) to the left of *Google Cloud Platform* in the upper left side of the screen, and select *App Engine* from the dropdown menu.

8. Open the *Select a language* menu and select *Java*.

9. Select your preferred datacenter location and click *Next*.

Google will then configure the server. This may take a moment.

10. When this completes, you will be directed to begin a tutorial to install a sample application. Choose *Cancel Tutorial* and confirm that you want to not perform that tutorial.

11. Launch the ODK Aggregate installer on your computer. (*See download info here.*)

    Linux

    macOS

    Windows

    Before launching, change the installer's permissions to enable running it as a program:

    (a) Right click on the file.

    (b) Select *Properties* → *Permissions*.

    (c) Check *Allow executing file as program*.

    Or, from a terminal, go to the directory where you downloaded the installer and change permissions:

```
$ chmod 554 "ODK Aggregate vN.N.N linux-installer.run"
```

    (Use actual name of the file, which will be different.)

    (a) Unzip the downloaded file before running the installer within it.

    (b) When you attempt to run the installer, macOS will prevent it. Go to → *System Preferences* → *Security & Privacy* to enable running the installer.

    On launch, you may need to approve running an unsigned installer.

The installer will guide you through configuring ODK Aggregate for App Engine. Click on the *Forward* button each time you complete a step to move ahead.

12. Accept the license agreement.

13. Select a parent directory under which an `ODK Aggregate` directory will be created to contain the configured software. Click on the *folder* icon to choose a directory.

14. Select *Google App Engine* as the platform for the Aggregate server.

15. Enter a name for your ODK Aggregate instance.

Note:

- The Aggregate instance name will be displayed to your users when they log into Aggregate using their username and password.
- The instance name does not need to be the same as the Project Name you set in Google App Engine. However, it might be helpful to use the same name.

Tip: Including the name of your organization in the instance name can help users confirm that they have contacted the correct website.

16. Enter a superuser name in the next window.

Note:

- The superuser will have full permissions on the system.
- The password for this user will be set to `aggregate` initially.
- Only this user will be allowed to log onto the system when Aggregate is run for the first time.

- Upon first logging in, the superuser should change the password and complete the configuration of Aggregate by specifying additional users and what permissions they will have on the system.

17. Enter the ID of the project you created on the Google Cloud platform.

The installer will configure Aggregate and launch an upload tool.

Tip: Depending on your Java version, security settings may prevent the upload tool from running. A possible workaround is to add `file://` to the Exception Site List.

18. Enter the Gmail account that you specified when setting up the App Engine project and click the *Get Token* button.

Your default browser will open a Google dialog screen asking you to choose a Gmail account. Select the account you specified earlier when setting up App Engine, and then allow *Google App Engine appcfg* to view and manage your App Engine instances and datastores. Click *Allow*. This will take you to a screen with instructions to copy a code.

At the same time, the install wizard should display a popup dialog box.

Tip: If the popup dialog does not appear, relaunch the upload tool:

Linux

macOS

Windows

(a) Close the upload tool.

(b) Open a terminal.

(c) `cd` to the directory you specified earlier.

(d) run `uploadAggregateToAppEngine.sh`

(a) Close the upload tool.

(b) Open a Finder window.

(c) Navigate to the directory you specified earlier.

(d) Run `uploadAggregateToAppEngine.app`

(a) Close the upload tool.

(b) Open a file explorer window.

(c) Navigate to the directory you specified earlier.

(d) Double-click **`ODKAggregateAppEngineUpdater.jar`**

Re-enter the email address, and click *Get Token* again. The popup dialog should now appear.

19. Copy the code from the browser into the installer's popup dialog and click *OK*.

Tip: The text box on Google's site is not as wide as the code. Be sure to copy the entire code.

20. If everything went well, you should see a status message letting you know the `Action Succeeded`.

Tip:

- If the output does not look like that, you may have waited too long between getting the code and pasting it into the tool. Click *Delete Token* and try again.
- If you see a failure message in the output window, it is likely that you have several different Gmail accounts and Google has gotten confused during the token-issuing process. In you suspect this is the case, click *Delete Token* and try again:

  (a) When the browser window opens, before selecting an account, copy the URL.

  (b) Open a Private Browsing or Incognito Window in your browser.

  (c) Paste the URL into the private window.

(d) Proceed with the other steps as above.

21. Click *Upload ODK Aggregate*.

Clicking on *Upload ODK Aggregate* will generate a long list of progress messages in the Output window. You will see a number of warnings and errors. Don't worry, this is expected.

For reference, here is a list of few of those errors:

```
 listBackends : Warning: This application uses Backends, a deprecated␣
↪feature that has been replaced by Modules, which offers additional␣
↪functionality. Please convert your backends to modules as described at:␣
↪https://developers.google.com/appengine/docs/java/modules/converting.

 listBackends! : WARNING: Error posting to URL: https://appengine.google.
↪com/api/backends/delete?backend=background&app_id=project-123-181306&
 listBackends! : 500 Internal Server Error

 listBackends : Unable to list backends: Error posting to URL: https://
↪appengine.google.com/api/backends/list?app_id=project-123-181306&
 listBackends : 500 Internal Server Error

 deleteBackendBackground : Warning: This application uses Backends, a␣
↪deprecated feature that has been replaced by Modules, which offers␣
↪additional functionality. Please convert your backends to modules as␣
↪described at: https://developers.google.com/appengine/docs/java/modules/
↪converting.

 deleteBackendBackground!: WARNING: Error posting to URL: https://
↪appengine.google.com/api/backends/delete?backend=background&app_
↪id=project-123-181306&
 deleteBackendBackground!: 400 Bad Request
```

```
deleteBackendBackground : Unable to delete backend: Error posting to URL:␣
↪https://appengine.google.com/api/backends/delete?backend=background&app_
↪id=project-123-181306&
deleteBackendBackground : 400 Bad Request
```

22. Finally, you should see the message `status : Action Succeeded!`.
23. Once the installer has run and uploaded the ODK Aggregate configuration to App Engine, return to the Google Cloud Platform console.
24. Open your Aggregate server from your project's screen, by selecting → *App Engine* and clicking on the project's URI.

25. *Log In* with the superuser username that you specified in the installer (the initial password for this username will be `aggregate`), and access the site administration screens for your server.

26. Go to *Site Admin -> Permissions* to change your password. You can also add additional users.

7.2.2 Installing on Tomcat (Local or Cloud)

Apache Tomcat is an open source Java web server that can be used to serve ODK Aggregate.

This document guides you through the installation and initial setup of a self-hosted instance of ODK Aggregate, running on Apache Tomcat with a MySQL or PostgreSQL. "Self-hosted" could mean on your own hardware or on a cloud-based server such as Amazon Web Services. (Though, for AWS in particular, we have *additional instructions*.)

Warning: Self-hosting, whether local or cloud-based, requires expertise in server maintenance, security, systems administration, and networking. If your organization does not have this expertise, we recommend using *Google App Engine*.

Note:

- Local hosting implies that you are taking ownership of the off-site back up and restoration of your data and are documenting the steps necessary to return your systems to operation in circumstances that might include a full hardware failure or the destruction of your facility. You must also plan for the security of your data and systems.
- Local hosting requires that you configure your network routers. It is recommended

to seek assistance from your local computer technical support community before proceeding. The set-up of the ODK Aggregate web server and database are very easy in comparison.

Considerations before you begin

Availability

Decide the availability of your server depending on how frequently you want to update and upload forms. If you do need a high-availability server, you need to talk to your Internet Service Provider (ISP) as to their availability guarantees.

Data Loss

Your tolerance to data loss will impact your backup schedule and your server hardware. Invest in a storage system based on your tolerance to data loss. Seek technical assistance for these requirements. If you cannot tolerate any data loss, or less than 24 hours of data loss, you should invest in a RAID storage array with battery-backed controller cards. If you can tolerate a day or longer interval of data loss, be sure you have a periodic tape or other means of backup for your system that matches or is shorter than the data loss interval.

Dataset Size

The quantity of data you intend to collect will affect the size of the machine required to host the ODK Aggregate instance and of your database server. For most use cases, the default size should be fine. If you are collecting more than 6000 submissions, you may need to increase the JVM size. Note that the maximum size of the JVM is limited by the size of the physical memory on your machine.

Secure and Protected Data

If you need to prevent eavesdroppers from seeing your data as it is transmitted to your ODK Aggregate instance, you need to do one of the following:

- only connect to ODK Aggregate from within your organization's network (when the ODK Collect devices are on your premises)
- obtain an SSL certificate and install it on your Tomcat server (a certificate is required to secure transmissions over *https*)
- use *Encrypted Forms*

Encrypted forms can be used in conjunction with either of the first two suggestions.

If you are not using encrypted forms and are handling sensitive data, a computer security specialist should review your system and your security procedures.

Database Systems

ODK Aggregate works with any of these database servers:

- MySQL
- PostgreSQL
- Microsoft SQL Server
- Azure SQL Server

Most deployments will use either MySQL or PostgreSQL, as they are the two most prevalent open source relational database servers.

MySQL is the most popular, so you will likely find more qualified professionals to install, administer, and debug it. However, you should consider PostgreSQL if you plan to:

- collect geographic data
- use forms with a very high number of questions (over 200)

PostgreSQL has better built-in support for geographic data, and MySQL's tables have a row size limit that will affect performance for very large forms.

Note: Use of an SSL and *https* is recommended for any deployment accessed from the internet.

Installation procedure

1. Install Tomcat on your server.
   (a) Install Java 8 or higher on your system.

   > **Note:** You generally need to launch installers with *Run as administrator* privileges (available under the right-click menu). Accept all the defaults.

   (b) Add the installed Java bin directory to the PATH variable.
   (c) Download and install Tomcat 8

- If using the Windows installer, change to use port 80 for the HTTP/1.1 port. If you are going to set up an SSL certificate, change the HTTPS/1.1 port to 443. Use all other defaults.
- Verify that Tomcat 8 is running by opening a browser on this server to `http://localhost/`. You should see the Apache Tomcat administration page. If you didn't request port 80 during the install, you will need to specify the port you chose (`http://localhost:port/`). If you didn't configure a port, the default port is 8080 (and 8443 for HTTPS).
- **Linux Installs**
  - To ensure that the proper Java settings are found by the web server, you may need to specify the `-E` flag when restarting the webserver.

    ```
    $ sudo apt-get install tasksel
    $ sudo tasksell install tomcat
    $ sudo apt-get install java8-jdk
    ```

  - Now open `/.bashrc` with your editor and add `export JAVA_HOME = /usr/lib/jvm/java-8-openjdk-amd64` at the bottom of that file. Change this to whatever path is appropriate for your Java installation.

    ```
    $ sudo -E /etc/init.d/tomcat8 restart
    ```

    The **E** flag on the last command is critical. It forces Ubuntu to reload the environment settings for the service, causing it to pick up the new *JAVA_HOME* setting.
- Apply or change the administrator password for Tomcat; the administration functions should be secured.
- ODK Aggregate v1.4.13 and higher are supported on Tomcat 8.0; these newer releases should also work, without modification on other webservers.
- Prior to ODK Aggregate v1.4.13, we only supported Tomcat 6. Tomcat 7, Tomcat 8, Glassfish and Jetty require additional configuration steps to run ODK Aggregate v1.4.12 and earlier. All of these webservers require configuration settings to enable cookies under HTTPS.
  - **For Tomcat 7**: Edit `context.xml` (under Tomcat 7's conf directory) to have the attribute 'useHttpOnly' set to false.

    ```
    <Context useHttpOnly="false">
    ```

  - **For Tomcat 8**: My ODK Aggregate file is installed as `/var/lib/tomcat8/webapps/ODKAggregate.war`. The following content needed to be placed in the file `webapps/ODKAggregate/META-INF/context.xml`

(this is within the expanded content of the war file, once the Tomcat 8 server has exploded it).

```
<Context path="" useHttpOnly="false" />
```

– **For Glassfish 4**: Add `glassfish-web.xml` under ODK Aggregate's WEB-INF directory with the content:

```
<?xml version="1.0" encoding="UTF-8"?>
  <glassfish-web-app>
    <session-config>
      <cookie-properties>
       <property name="cookieHttpOnly" value="false" />
      </cookie-properties>
    </session-config>
  </glassfish-web-app>
```

– **For Jetty**: Add `jetty-web.xml` under ODK Aggregate's WEB-INF directory with the content:

```
<?xml version="1.0"  encoding="ISO-8859-1"?>
<!DOCTYPE Configure PUBLIC "-//Jetty//Configure//EN" "http://
↪www.eclipse.org/jetty/configure.dtd">

<Configure class="org.eclipse.jetty.webapp.WebAppContext">
  <Get name="sessionHandler">
    <Get name="sessionManager">
      <Set name="secureCookies" type="boolean">true</Set>
    </Get>
  </Get>
</Configure>
```

2. Configure your server and network devices so that laptops or Android devices connecting to the internet from an external access point can access your server.

   If your organization has a network or systems administrator, contact them for assistance. The steps for this are:

   - configure your server firewall to allow access
   - make your server visible on the internet (optional)
   - establish a DNS name for the server

3. Obtain and Install an SSL certificate if you need secure `https` access.

4. Select and Install your database server.

   ODK Aggregate works with any of these database servers:

   - MySQL

- PostgreSQL
- Microsoft SQL Server
- Azure SQL Server (requires Java 8)

- For MySQL, download and install MySQL Community Server 5.7 or higher from MySQL download site. Be sure to set a root password for the database. Stop the MySQL database server, then configure the database (via the `my.cnf` or the `my.ini` file) with these lines added to the [mysqld] section:

```
character_set_server=utf8
collation_server=utf8_unicode_ci
max_allowed_packet=1073741824
```

  and restart the MySQL databaseserver. Then, download the *MySQL Connector/J*, unzip it, and copy the `mysql-connector-java-x.x.x-bin.jar` file into the Tomcat server's libs directory. After copying it into that directory, you should stop and restart the Tomcat server. The *max_allowed_packet* setting defines the maximum size of the communications buffer to the server. The value used in the snippet above is 1GB, the maximum value supported. For ODK Aggregate 1.4.11 through 1.4.7, and 1.2.x, the maximum media (e.g., image or video) attachment is limited to the value you set for max_allowed_packet minus some unknown overhead – e.g., a storage size of something less than 1GB. For ODK Aggregate 1.4.6 and earlier (excluding 1.2.x), the maximum media attachment is unlimited and the setting for max_allowed_packet does not need to be specified. For ODK Aggregate 1.4.12 and later, the max_allowed_packet value should be set to a value greater than 16842752 (this is the minimum value that should be used: 16MB plus 64kB); with that setting, media attachments of unlimited size are once again supported. If you are upgrading to a newer ODK Aggregate, you must continue to use the setting you already have, or 16842752, whichever is greater. If you experience problems uploading large attachments, change this setting to its maximum value, 1073741824. Finally, if you are using MySQL 5.7 or later, some of releases expire all database passwords after 360 days. Please verify the behavior of your version of MySQL and either change the password expiration policy or create a calendar reminder to change the password before it expires. For ODK Aggregate, you will need to re-run the installer to specify the new password.

- For PostgreSQL, download and install the appropriate binary package from PostgreSQL download site. Be sure to set the password for the postgres (root) user and set the default character set and collation sequence.

- For either database, you should ensure that the default character set is configured to be UTF-8 and that the collation sequence (dictionary order) is set appropriately for your circumstances. If it isn't, any non-Latin characters may display as question marks. Refer to the character set and collation sections of your database's documentation for how to do this.
- For Microsoft SQL Server or Azure SQL Server, you should configure these with UTF-8 character sets and to use Windows authentication. When using Windows authentication, the user under which the webserver executes must be granted permissions to access the SQL Server instance. The install wizard for ODK Aggregate will produce a `Readme.html` file that contains additional information on how to complete the configuration of the database and webserver service.

5. Download and install ODK Aggregate. Select the latest Featured release for your operating system.

Note: The installer will guide you through configuring ODK Aggregate for Tomcat and MySQL/PostgreSQL/SQLServer. The installer will produce a WAR file (web archive) containing the configured ODK Aggregate server, a `create_db_and_user.sql` script for creating the database and user that ODK Aggregate will use to access this database, and a `Readme.html` file with instructions on how to complete the installation.

Tip:

- When asked for the fully qualified hostname of the ODK Aggregate server, you should enter the DNS name you established above. The install also asks for a database name, user and password. The user should not be root (MySQL) or postgres (PostgreSQL). ODK Aggregate will use this user when accessing this database (and it will only access this database). By specifying different databases and users, you can set up multiple ODK Aggregate servers that share the same database server, store their data in different databases, and operate without interfering with each other.
- If you are upgrading to a newer version of ODK Aggregate, as long as you specify the same database name, user and password, you do not need to re-run the `create_db_and_user.sql` script (it only needs to be executed once).

7.2.3 Installing on AWS (Cloud)

This document provides instructions on setting up ODK Aggregate using Apache Tomcat and MySQL on a Linux micro-instance on the Amazon Web Services EC2 infrastructure.

Additional Resources

- For more on the general set-up of Tomcat on AWS, see Cat in the cloud Apache Tomcat Series.
- For an introduction to hosting on AWS, see Amazon's getting-started guides.
- For details on setting up ODK Aggregate on Tomcat, but not in an AWS environment, see *Installing on Tomcat (Local or Cloud)*.

Before you get started...

Get an Amazon Web Services account at http://aws.amazon.com/ec2/.

> **Note:**
>
> - You will need to confirm a working phone number.
> - If you want to use a U.S. number but are presently overseas, you can use a Google Voice number routed to Google Chat.

1. Go to the AWS/EC2 management console and note your region (shown in the upper-left).
2. Launch a new instance with the *Launch Instance* button prominently displayed on the EC2 console home screen.

   Accept the default behavior and use the quick-launch wizard.
3. For the launch configuration, choose the *Amazon Linux AMI: EBS-Backed (64-bit)* option.

   This is one of the instance types that you can run on a micro-instance as part of their free tier.

   Leave everything else with the default settings, including the instance type. The instance type will default to `t2.micro` which is a small, limited instance that can be run for free. For a price, you can upgrade the instance type later if you need better performance.

   > **Note:** The exact AMI name and ID will depend on your region.

4. Create a new key pair, download the private key, and keep the private key safe.

This will be your only method of communicating with your new instance and **you will not be allowed to download it again**.

5. After creating the instance, add security rules for allowing both HTTP and HTTPS.

   Choose the *Security Groups* tab and click on the auto-created security group associated with your new instance. This might have been called `launch-wizard-1`. If you're not sure, you can go to the *Instances* tab to see which *Security Group* is listed for the new instance.

   In the properties pane at the bottom, click on the *Inbound tab*, select `HTTP` from the *create a new rule* drop-down, then click *Add Rule*. Do the same for `HTTPS`. Then click *Apply Rule Changes*.

   Tip: To avoid potential problems with MTU settings and packet loss, also add a rule to allow *All ICMP*. Add a rule for both IPv4 and IPv6. After you create the new rule, click *Apply Rule Changes*.

   Note: Tomcat defaults to listening on nonstandard ports `8080` and `8443`. In this guide we provide instructions to use the standard HTTP and HTTPS ports instead.

   However, if you want to leave Aggregate on the non-standard ports, you can certainly do so. In that case, simply add two additional security rules to allow access via `8080` and `8443`.

6. Switch to the *Instances* tab, click on your instance, and note its Public DNS Address in the properties pane below.

   This is the default address that you will use to access your instance.

7. Allocate an *elastic IP* and domain name.

   This step is not required, but it allows you to access your Aggregate instance using a human readable URL set by you. So we recommend it.

   Navigate to Elastic IPs and click *Allocate New Address*. Associate it with your new instance. Note your new IP. Also, if possible, configure DNS to route one or more names to this address. You can then use this IP and/or name to access your instance (and can forget the Public DNS Address assigned by AWS).

   Note: The IP is free so long as you keep it associated with a running instance. If you stop your instance and do not release the IP address for others to use (in essence, wasting it), then Amazon will begin charging you for holding the unused address.

Tip: When you set up Aggregate below, you will need to configure it with the domain name you will use to access it. Thus, it is best if you configure the domain name first.

8. Connect to your instance.

   Go to the *Instances* tab and select *Connect* from the Instance Actions drop-down (alternatively, you can right-click on the instance and choose Connect). From here AWS presents you with several options.

   - The easiest is to connect using their Java SSH client. If you choose that option, you just have to specify the location of your private key file (created above) and AWS launches an in-browser SSH client to connect to your instance.
   - You can also select to connect with a stand-alone SSH client. If you choose this option, AWS will provide extremely helpful instructions, including an SSH command that you can cut and paste into your local command window. It will also inform you that you may need to update the permissions on your local private key file in order for the ssh client to run properly, and it will even give you the command to run (for example, **`chmod 400 xxx.pem`**).
   - Once you connect, you will probably be told that there are new security updates to install. You can run **`sudo yum update`** to install these updates, as it advises.

9. Transfer files to your instance.

   When you login using `ssh`, you will be in the (empty) `ec2-user` home directory. You will want to be able to transfer files between here and your local directory. You have several options.

   - If you're using the command-line ssh, you can also use the command-line scp to copy files. The syntax is similar to ssh, but of course you also need to specify the source and destination file paths.
   - An easier option is to use an FTP client like FileZilla. (You can use any FTP client, as long as it supports SFTP).
     - To configure FileZilla to connect to your instance, go into *Edit→Settings/Preferences→Connection→SFTP* and add your private key to FileZilla's keystore (it will offer to convert the key format, which you should accept). Then, go into Site Manager and create a new site. The host should be the IP, name, or Public DNS for your instance, the port can be blank, the protocol should be **SFTP – SSH File Transfer Protocol**, the login type should be Normal, and the user should be **ec2-user**. Everything else should be left at the defaults, including the password (which will be blank). When you connect, the default directory will be the ec2-user's home directory, but you can also navigate to other directories.

10. Install Tomcat 6.

While logged into your ec2 instance:

```
$ sudo yum install tomcat6
```

This installs configuration files into `/etc/tomcat6` and other files into `/usr/share/tomcat6`. Log files go into `/var/log/tomcat6`.

11. Install MySQL.

While logged into your ec2 instance:

```
$ sudo yum install mysql mysql-server
```

Open `/etc/my.cnf`. In the `[mysqld]` section, add:

```
character_set_server=utf8
collation_server=utf8_unicode_ci
max_allowed_packet=1073741824
```

Note: Whatever your preferred editor, when updating files on the server, the easiest tool is probably vim. This will open the file directly in your terminal.

```
$ vi /etc/my.conf
```

Vim can be a little tricky to use at first. There are many tutorials online. You may find this interactive Vim introduction helpful.

12. Run MySQL.

```
$ sudo service mysqld start
```

13. Install and transfer ODK Aggregate files.

(a) First, install ODK Aggregate on your local computer (not on your AWS instance).

Note: During set-up, it's important to specify that this will be a MySQL installation, and it is also very important that you specify the correct domain name or IP address that will be used to access your Aggregate server. Ideally, this will be a specific domain name that you have already mapped to an elastic IP (and can re-map later if you change the IP).

(a) The installation will create a file named `create_db_and_user.sql`. Upload this to your ec2-user home directory.

(b) The installation will also create a file named `ODKAggregate.war`. Rename this to `ROOT.war` and upload it to the `/usr/share/tomcat6/webapps` folder. If you

receive a *Permission Denied* error, you might need to execute **`chmod -R 755`** or something similar for the webapps folder.

(c) After `ROOT.war` has been copied to the server, you need to make sure Tomcat has permission to use it. Run **`sudo chown tomcat ROOT.war`** and **`sudo chgrp tomcat ROOT.war`** in the webapps directory to ensure this is the case.

2. Configure MySQL.

   (a) On your AWS instance, run `/usr/bin/mysql_secure_installation` to set a root password and secure your MySQL installation.

   (b) Run **`mysql -u root -p`** to log in to MySQL (specifying the password you just set), and type **`source ~/create_db_and_user.sql`**. This will create the ODK user and database. Type **`quit`** on the mysql prompt to quit from MySQL.

   (c) Run **`sudo /sbin/chkconfig --levels 235 mysqld on`** to auto-start MySQL whenever your instance boots up.

3. Configure Tomcat.

   (a) Download the MySQL Connector/J from the MySQL download page. Unzip it and transfer the `mysql-connector-java-x.x.x-bin.jar` file up to your instance's `/usr/share/tomcat6/lib` directory.

   (b) Edit `/etc/tomcat6/server.xml` in order to customize settings.

   (c) Assuming that you want to run Aggregate on the standard HTTP port (80) and HTTPS port (443):

   - Change `<Connector port="8080" protocol="HTTP/1.1"` to `<Connector port="8080" proxyPort="80" protocol="HTTP/1.1"` (that is, add the `proxyPort` attribute).
   - If you are using SSL, also change `<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"` to `<Connector port="8443" proxyPort="443" protocol="HTTP/1.1" SSLEnabled="true"`.
   - Execute the following commands to have Linux forward to the ports on which Tomcat listens:

```
$ sudo /sbin/iptables -t nat -I PREROUTING -p tcp --dport␣
↪80 -j REDIRECT --to-port 8080
$ sudo /sbin/iptables -t nat -I PREROUTING -p tcp --dport␣
↪443 -j REDIRECT --to-port 8443
$ sudo /sbin/service iptables save
```

   - If you have an SSL certificate for HTTPS support:
     - Make sure that the `<Connector port="8443"` part of the configuration file is not commented out. If it is, un-comment it.

  - Upload your SSL keystore file and the certificate(s) to the server.
  - Install it as instructed. (If you buy from RapidSSL, for example, they provide you with Tomcat installation instructions. For example, you may need to download a special P7S certificate file, then install it on the server with **`keytool -import -alias YOURALIAS trustcacerts file xxxxx.p7s -keystore xxxxx.keystore`**.)
  - In the `<Connector port="8443"` part of the configuration file, specify the location of your keystore file and password (for example: `keystoreFile="/…/xxxxx.keystore" keystorePass="changeit"`).

- Start Tomcat with **`sudo service tomcat6 start`**.
- Configure Tomcat to auto-start when the instance boots with **`sudo chkconfig --level 345 tomcat6 on`**.

4. Login and test.

   At this point, you should be able to login to your AWS-hosted Aggregate instance by going to its domain name or IP address in your web browser (with or without HTTPS, depending on your set-up).

   Note: For your first login, you will need to login with the Google account you specified during the Aggregate installation process. Then you can add additional users from the *Site Admin* tab.

   Once you have confirmed that your Aggregate instance is working, you can back it up by creating an image of the instance (an AMI).

   To do this, go to the *Instances* tab in the AWS-EC2 console, then select the *Create Image (EBS AMI)* Instance Action for your instance.

5. Further set-up for production servers.

   - You will want to create a system to monitor and manage the log files in `/var/log/tomcat6`.
   - You will also want to create a system for regular back-ups and a plan for how to restore them when needed. This will be needed to safely back up the MySQL database, which may be in-use at any given time.

Note:

- The micro instance is only free for 12 months from AWS sign-up, and that you may exceed the free quotas on disk space or network bandwidth before that point (see).

- You may at some point need to upgrade your instance to a standard instance if the micro instance is not providing enough performance.

7.2.4 ODK Aggregate Virtual Machine

The ODK Aggregate VM is a fully-configured install of Aggregate that you can run on any computer. It requires very little setup, works well without Internet connectivity, and gives you complete control over your data collection campaign.

See also:

Planning Your Aggregate Deployment

7.3 Upgrading Aggregate

7.3.1 Need for Upgrading

It is important to upgrade to newer ODK Aggregate versions as they come out. You don't need to do this immediately, but this should be something you do at least once a year.

There are several reasons for this.

Security vulnerabilities

The ODK developers are constantly upgrading the libraries we use with newer, safer, versions. The older your software, the greater the number of vulnerabilities in it.

Hosting revisions

If you are using Google App Engine as we recommend, your hosting environment is being continuously updated.

Google AppEngine is a managed environment, unlike AWS or other "bare-box" hosting services. Google is continuously updating features and removing support for older features in this environment.

The Aggregate development team is committed to supporting Google App Engine, so we update our application as the platform changes. If you do not upgrade gradually as new versions come out, your installation may stop working or cease to have a viable upgrade path.

Performance improvements

As we find performance issues and address them, the Aggregate gets faster and cheaper to run.

Enhanced capabilities

ODK JavaRosa, the underlying form rendering library is updated about annually. Those updates add new functions, features, and occasionally data types. New features are slowly added to ODK Aggregate, too.

To use these newer features, keep your Aggregate installation up to date.

7.3.2 Determining your Aggregate version

If you are able to log onto the server as a Site Administrator, the ODK Aggregate version is displayed at the top of the *Preferences* sub-tab under the *Site Admin* tab.

If you are unable to log onto your server, you will need to search for the version in the application logs. To do that:

1. Open a browser to https://appengine.google.com.
2. Choose the project id for your ODK Aggregate server by clicking on the project drop-down in the top left corner.

3. Search *logs* in the search box and select *logging*.

4. In the filter text box paste this text : *afterPropertiesSet* and hit enter.

5. Expand the list of logs and find the log which shows the version of Aggregate. It will of the following format:

```
13:24:54.806 org.opendatakit.common.security.Realm afterPropertiesSet: Version:␣
↪v1.4.15 Production (Realm.java:51)
```

Note: In the 1.0 Production release, the Version appears on the *Site Admin* Tab, under the *Preferences* sub-tab. If no version information is displayed even after you log in, then you are running an extremely old version of ODK Aggregate; in that case follow this cheat

sheet:

- *Beta 1 and earlier*: *Permissions* tab looks nothing like the tab in the Production release.
- *Beta 2*: *Permissions* tab looks like the Production release; Forms tab does not show the number of media files associated with a form.
- *Beta 3*: The number of media files associated with a form is displayed; publishing to Google is broken.
- *Beta 4*: There are *Form Management* and *Site Admin* tabs; publishing to Google is broken.
- *RC1*: csv exports apply the active Filter to the exported columns; publishing to Google is broken.
- *RC2*: *Submissions* tab is paginated (shows 500 records per page by default); publishing to Google works.
- *RC3*: kml exports apply the active Filter to the exported data. All data is exported. It no longer truncates to the first 500 records.
- *RC4*: Submissions for forms with non-repeating groups are now properly reconstructed into Briefcase. Date fields are broken.

Publishing to Google is broken prior to RC2 (due to a change in Google infrastructure).

7.3.3 General steps for upgrading

1. Disable all submissions to ODK Aggregate, in the *Form Management* tab.
2. Use *ODK Briefcase* to pull a copy of all data to your computer.
3. Search upward in the *release notes* below, starting from your current ODK Aggregate version to locate the most recent ODK Aggregate version that does not require any manual upgrade steps.
4. Log onto your server to confirm that it is still functioning.
5. Download the next ODK Aggregate version, perform the manual upgrade steps if any and upgrade to that version (and perform any post-install upgrade steps, if any).
6. Log onto your server to confirm that it is still functioning.
7. Repeat the steps 3-6 until you have upgraded to the current version.
8. Enable submissions to ODK Aggregate via the *Form Management* tab.

Tip: You need to know the exact *instance name* that was used in prior installs for your username and password to continue to work. If you add a space or change capitalization or spelling, the passwords will be invalid (you just need to re-run the installer with the correct string to correct the problem).

7.3.4 Release notes

Aggregate v1.4.15

1. Update javarosa to the new opendatakit-javarosa-2.3.0.jar.
2. Fix: rev 210 sync protocol. User permissions were incorrectly being computed and filtered. This prevented resetting the server with new content from the device (but syncing with existing content worked fine).

Note: The rev 210 sync protocol is incompatible with anything prior to rev 210.

Upgrade notes

- Do not upgrade if you are using ODK 2.0 rev 208 or earlier.
- See *v1.4.13* upgrade notes.

Aggregate v1.4.14

1. Update javarosa to the new opendatakit-javarosa-2.2.0.jar
2. Fix for JSON publisher correctly handling UTF-8
3. Added SHA-1 library to browser so that Basic Auth can now be configured and will now work (this capability is not exposed in the wizard-based installer/configurer; existing passwords need to be changed before they can be used in a Basic Auth configuration)
4. Revised sync protocol and database schema for ODK 2.0. This is incompatible with anything prior to rev 210 (not yet available).
5. Microsoft SQLServer configurations can now run on Linux and MacOSX boxes.
6. Updated to the latest AppEngine SDK.
7. Updated build environment to a mixed Java 7 and Java 8 environment.

Upgrade notes

- Do not upgrade if you are using ODK 2.0 rev 208 or earlier.
- See *v1.4.13* upgrade notes.

Aggregate v1.4.13

1. Add support for Microsoft SQL Server database and Azure SQL Server
2. Updated to expect Tomcat 8 and JDBC 4 libraries. This extensively impacted the datastore layer.
3. Rewrote the Google Sheets publisher to use a newer API.
4. Changed the PAUSED publisher state to expose a restart button. Fixed bug that prevented ABANDONED button from working. When a publisher enters the Paused state, this button triggers ODK Aggregate to resume from that state. This happens automatically every 8 minutes or so, but having a resume button simplifies integration testing. i.e., when a publisher is in the Paused state, you do NOT need to click this button – but if you are testing a publisher to your own server, it can be useful to have.
5. Fixes for ODK 2.0 datatype form.
6. NaN +Infinity and -Infinity are now handled within Google AppEngine. These cannot be supported on systems using MySQL and SQL Server (a fundamental limitation of those databases). On PostgreSQL systems, if you alter your tables to use REAL instead of DECIMAL(38,0) columns, it will be able to store and process these special values.
7. Updated a multitude of jars (libraries).
8. If this is a first-time install using PostgreSQL, please see the upgrade steps below for the additional commands you need to execute to complete the PostgreSQL configuration. These commands are missing from the `create_db_and_user.sql` but are required for PostgreSQL operations.

Upgrade notes

- If publishing to Google Sheets, you must go to the API Manager tab and enable the Google Sheets API.
- Tomcat installs now require Tomcat 8 or higher. The war file produced by the installer will work as-is on those systems – no need for any manual modifications to get things to work (though MySQL still requires downloading and copying the MySQL Connector/J, and SQL Server also has a manual configuration step).

- If using an older MySQL database, please upgrade to MySQL 5.7 or higher; note that some MySQL releases expire all database passwords after 360 days. Please verify the behavior of your version of MySQL and either change the password expiration policy or create a calendar reminder to change the password before it expires. For ODK Aggregate, you will need to re-run the installer to specify any new password. For more information, see the MySQL documentation. e.g., MySQL password expiration policy
- The ODK 2.0 sync protocol in this release is compatible with the ODK 2.0 Rev 200, 202, 204, 206 and 208 tools. Note that you cannot just upgrade from a server earlier than 1.4.10; you have to delete everything and start over.
- See *v1.4.12* upgrade steps if upgrading from an earlier ODK Aggregate version.
- If using an older PostgreSQL database, please upgrade to at least 9.4. For all PostgreSQL servers, you must execute an additional command to grant permissions to the schema on the server. This is a change from the earlier 9.1 install and the additional command is missing from the `create_db_and_user.sql` script. The commands to be executed in the pgAdmin query window are as follows, with your_database, your_schema and database_username replaced with the values you specified when you ran the installer:

```
\c "your_database";
alter schema "your_schema" owner to "database_username";
```

Aggregate v1.4.12

1. Update to installer with new language and links to documentation.
2. Revise upload tool for Google AppEngine deployments to more-consistently display the token-entry dialog.
3. Add .csv-based downloading and bulk updating of configured users and their capabilities.
4. Change MySQL ODK 1.x and ODK 2.0 data table constructions to use MEDIUMBLOB columns for media attachments. Fix several issues around accessing attachments. See Upgrade notes below and MySQL configuration instructions in *installing Aggregate on Tomcat* for configuration that is required for MySQL.
5. Tweak Google AppEngine configuration to reduce frontend usage hours and thereby reduce costs.
6. Performance improvements to Google AppEngine task locks (primarily for ODK 2.0 support).
7. Add `/users/list` URL that returns the list of all configured users and the roles they have been granted. Access to this URL requires authentication and is restricted to

ODK Aggregate usernames and Google accounts (anonymousUser access is forbidden). If the authenticated user does not have Tables Super-user, Administer Tables, or Site Administrator permissions, a singleton list is returned that contains only information about their own identity and its capabilities.

8. Changes to support row-level filtering in ODK 2.0.
9. Implement a paginated view of ODK 2.0 data rows; 100 records per page.
10. Upgrade to Google AppEngine 1.9.42 SDK.
11. The ODK 2.0 "datatypes" example form and table (in the app-designer repository) fail to upload to ODK Aggregate. This will be addressed in a future release.

Upgrade notes

- For MySQL installations, please make sure you have this server configuration (if this is a new database install, you can reduce *max_allowed_packet* to *16842752*):

```
character_set_server=utf8
collation_server=utf8_unicode_ci
max_allowed_packet=1073741824
```

- For ODK 1.x uses, no special upgrade steps are required.
- If you are using ODK 2.0 features, you must visit the *Permissions* sub-tab under the *Site Admin* tab and click *Save Changes* to complete the upgrade to this release. You do not need to modify anything on that page, just click the button.
- The ODK 2.0 sync protocol in this release is compatible with the ODK 2.0 Rev 200, 202, 204, 206 and 208 tools. Note that you cannot just upgrade from a server earlier than 1.4.10; you have to delete everything and start over.
- See *v1.4.11* upgrade steps if upgrading from an earlier ODK Aggregate version.

Aggregate v1.4.11

1. Add a mutex around ODK 1.x form submission creation and updating. This should eliminate submission data corruption.
2. If only ODK 2.0 privileges are assigned, hide the Submissions tab.
3. Fix: advisory memcache interaction to delete the entry when rolling back the datastore update.
4. Add **Tables Super-user** as a configurable user capability. Users with this capability will be able to modify the table-level and row-level privileges on ODK 2.0 rev 206

data tables. These privileges are advisory and do not provide strong access or revision control.

5. Add `/roles/granted` URL that returns the roles granted to an authenticated username or google account. This will be used by ODK 2.0 rev 206 tools to determine and save the capabilities of the configured user on a device. Those capabilities are then used to apply advisory access controls on the tables and rows.

Upgrade notes

- For MySQL installations, please make sure you have this server configuration:

```
character_set_server=utf8
collation_server=utf8_unicode_ci
max_allowed_packet=1073741824
```

- For ODK 1.x uses, no special upgrade steps are required.
- If you are using ODK 2.0 features, you must visit the *Permissions* sub-tab under the *Site Admin* tab and click *Save Changes* to complete the upgrade to this release. You do not need to modify anything on that page, just click the button.
- The ODK 2.0 sync protocol in this release is compatible with the ODK 2.0 Rev 200, 202 and 204 tools. You cannot just upgrade from a server earlier than 1.4.10; you have to delete everything and start over.
- See *v1.4.10* upgrade steps if upgrading from an earlier ODK Aggregate version.

Aggregate v1.4.10

1. On AppEngine, add advisory memcache interactions inside database mutex implementation (TaskLockImpl) to detect and thereby reduce race conditions that could lead to database corruption. These changes do not alter the fundamental mutex mechanism. They may reduce the likelihood of a mutex failure.
2. Fix ODK 2.0 sync: table-specific pre-loaded instance attachments under `config/assets/csv/tableid/instances/...` were not being identified as table-specific.
3. Fix ODK 2.0 sync: add mutex around app-level file, table-level file and row-level attachment REST interactions. Eliminates the possibility of joint-updating that could corrupt the app-, table- or row-level manifests.
4. Fix ODK 2.0 sync: change row-level file attachment bulk-update to not refetch the manifest after adding each file attachment. This had caused 1% failure rate on row-level attachment syncing when there were large numbers of file attachments (30-60) for each row.

5. Fix ODK 2.0 sync: when authorization fails (permission denied), return UNAUTHORIZED response rather than DENIED. This reports an authorization failure on the client vs. a protocol error.

Upgrade notes

- For MySQL installations, please make sure you have this server configuration:

```
character_set_server=utf8
collation_server=utf8_unicode_ci
max_allowed_packet=1073741824
```

- For ODK 1.x uses, no special upgrade steps are required.
- The ODK 2.0 sync protocol is compatible with the ODK 2.0 Rev 200, 202 and 204 tools.
- For ODK 2.0 uses, if you have pre-loaded datasets (via tables.init) that have row-level attachments (e.g., geotagger example dataset), you should:
  1. Remove these incorrectly-categorized files from the server (anything under `assets/csv/table_id/instances/...` on the *Manage App Level Files* sub-tab)
  2. Deploy your device configuration to a clean device
  3. Reset App Server to correctly upload these files so that they appear under the *Manage Table Level Files* sub-tab.

Aggregate v1.4.9

1. Add XPath 3.1 math functions (e.g., exp, log, sin, cos, etc.)
2. Fix AppEngine adaptation layer for Apache HttpClient so that Oauth2 authentications work change version string in `ODKAggregateAppEngineUpdater.jar` to be a date stamp.
3. Fix ODK 2.0 interfaces to support ODK Aggregate username / password for authentication.
4. Fix ODK 2.0 Sync so that a delete-table request can be repeated until successful or Not Found (404) when it times out.

Upgrade notes

- For MySQL installations, please make sure you have this server configuration:

```
character_set_server=utf8
collation_server=utf8_unicode_ci
max_allowed_packet=1073741824
```

- For all other installations, there are no special upgrade steps required.
- The ODK 2.0 sync protocol is compatible with the ODK 2.0 Rev 200, 202 and 204 tools.

Aggregate v1.4.8

1. Submission Filters are once again working.
2. Update javarosa library with fixes for:

- fix date, time, dateTime handling of time zones (in collaboration with SurveyCTO)
- fix itemset choice lists – various issues.
- fix jr:choice-name() context resolution so that it can work with relative paths and repeat groups (SurveyCTO contribution).
- add CONTROL_FILE_CAPTURE for future arbitrary-file-attachment handling
- use consistent UTF-8 treatment when writing and reading files
- better detection and handling of binary-file-format changes across versions
- build with Java 7

3. Add support for geotrace and geoshape to Export-to-KML (for ODK 1.x).
4. Update Apache HTTP Client libraries to 4.5.2; this should support SNI protocol interactions of external publishers (ODK 1.x - untested).
5. Change Google App Engine update mechanism to use new wizard update tool.
6. Change Google App Engine code to use EAR / module format.
7. Widespread jar update.
8. Correct bug in csv-export library shared with ODK 2.0 tools for exporting datasets in those tools (does not impact Export as CSV in the 1.x toolchain).
9. Add new bulk attachment-POST APIs for ODK 2.0 sync protocol.
10. Rework ui tests to use Chrome for the Selenium web testing.
11. Rework all of the eclipse projects and add documentation for setting up a tomcat eclipse debug environment.
12. Rework the Google App Engine maven projects to use the EAR / module format.

13. Update maven plugins and tools to newer versions.

Upgrade notes

- For MySQL installations, please make sure you have this server configuration:

```
character_set_server=utf8
collation_server=utf8_unicode_ci
max_allowed_packet=1073741824
```

- For all other installations, there are no special upgrade steps required.
- The ODK 2.0 sync protocol is compatible with the ODK 2.0 Rev 200 and higher tools.

Aggregate v1.4.7

1. Submission Filters are broken in this release. When submissions rows are corrupted, consistently report the information necessary for end-users to delete or repair the corrupted submission.
2. Correct ODK 2.0 delete-table functionality so that it does not generate errors when it is not running on AppEngine. The code had been performing an unguarded explicit cast to an AppEngine-only implementation class.

Upgrade notes

- For MySQL installations, please make sure you have this server configuration:

```
character_set_server=utf8
collation_server=utf8_unicode_ci
max_allowed_packet=1073741824
```

- For all other installations, there are no special upgrade steps required.

> **Warning:** Google disabled functionality used by this installer and earlier versions to upload ODK Aggregate to Google App Engine. See *here* for the work-around.

Aggregate v1.4.6

1. Submission Filters are broken in this release.

2. Fix for Google Sheets publisher-creation problem caused by April 20th deprecation of the Google Docs API. We were making a single call to that obsolete API to create a blank Google Sheets document.
3. Update to javarosa-2015-04-17 jar. That jar adds OpenStreetMap support to javarosa. Removal of the *Sign in with Google* mechanisms that used OpenID.
4. Widespread update to jars, custom repository configuration, and the elimination of several custom-built jars because of the elimination of OpenID and the maturity of Maven as a version repository.

Upgrade notes

No special upgrade steps required.

Aggregate v1.4.5

1. Update to javarosa-2015-01-10 jar.
2. Update to jQuery 1.11.1
3. Update the ODK 2.0 Data model and Sync protocol (incompatible with device releases: rev 122 and earlier).
4. Fix: support performing mark-as-complete on encrypted submissions (requires ODK Briefcase v1.4.5 or higher).
5. Fix: add server preference to ignore partially inserted/deleted submissions. Logs them but ignores them so that you can access all other rows in your dataset. Disabled by default. By default, all actions fail upon encountering any malformed submission. You should not ignore these failures but should correct them as soon as is practical.
6. Attempted fix: for *Log In* issue – insert sleep to give Google a chance to propagate clearing of session cookie. Tweak the webpage resize/layout calculations to be more efficient.
7. Fix: make incomplete deletions and insertions more recoverable.
8. Performance: change MySQL table creation to not declare primary keys and just use ordinary indices for the primary key.
9. Security: Support Enketo-express (allow non-https communications with Enketo server). When the communications are not secure, this change discloses the Enketo access token to eavesdroppers.
10. Security: Filter out forbidden characters in redirect string to prevent XSS attacks.
11. Security: Add clickjacking prevention header as detailed here: https://www.owasp.org/index.php/Clickjacking_Defense_Cheat_Sheet.

Upgrade notes

If you were testing out ODK Tables, you should delete all ODK Tables files and data tables before upgrading, as the database schema has changed. The server no longer works with ODK Sync 2.0 rev 122 (or earlier releases).

Aggregate v1.4.4

1. Installer now asks for an ODK Aggregate username for the super-user (not a Google e-mail account). Default password is aggregate.
2. Banner displayed if super-user's password has not been changed from aggregate.
3. Fix: column name generator bug (upload of form definition failed)
4. Fix: add more detailed error messages when a submission is corrupted
5. IE6 and IE7 are no longer supported. Upgrade your browser.
6. Allow user to specify the ODK 2.0 App Name.
7. Allow anonymous access to ODK 2.0 Sync APIs.
8. Enforce *Administer Tables* access for ODK 2.0 Sync APIs that alter server configuration.
9. For ODK 2.0 Sync APIs, remove JBoss Resteasy; use Apache Wink instead.
10. Extensive version updates to supporting software libraries.
11. Update `CONFIGURE.txt` instructions for maven builds. Define ANT scripts for external dependencies.

Upgrade notes

Upgrades require several manual interventions:

- You must have Java 7 installed - the GAE 1.9.7 SDK used within the installer now requires that version of Java. If you are upgrading from ODK Aggregate 1.4.3, you have already done this.
- If you were testing out ODK Tables, you should delete all ODK Tables data tables before upgrading, as the database schema has changed.
- You must flush the session cookies on the server. Session cookies are used to identify the logged-in users of the system. The security software versions were updated in this release, causing the older cookies to become invalid. See below for the upgrade steps.

After performing the above steps, the upgrade steps after those are as follows:

1. Open a browser and go to your Google AppEngine dashboard.
2. Click through to your application id. Then click on *Memcache Viewer* under the *Data* heading in the left sidebar. Keep this window open.
3. Now, run the installer and deploy ODK Aggregate 1.4.4 to this application id.
4. After it has deployed, click on *Flush Cache* in the dashboard window's Memcache Viewer screen.
5. Follow the instructions *here* for changing the password of the super-user username.

Other than the need to flush this cache, and the need to delete any ODK Tables data before upgrading, this should be a seamless upgrade from ODK Aggregate 1.4.x.

Aggregate v1.4.3

1. Updated javarosa jar.
2. Add stub Tomcat/MySQL Eclipse project with readme.
3. Update selenium test environment and various 3rd party jars.
4. Update the GAE SDK inside the installer to 1.9.0.
5. New ODK Tables sync protocol and UI. Breaks ODK Tables Alpha 2 sync.
6. New sync protocol exchanges SAVEPOINT_TYPE, FILTER_TYPE, FILTER_VALUE.

Upgrade notes

Upgrades require a manual intervention:

1. You must have Java 7 installed - the GAE 1.9.0 SDK used within the installer now requires that version of Java.
2. If you were testing out ODK Tables, you should delete all ODK Tables data tables before upgrading, as the database schema has changed.

Otherwise, this should be a seamless upgrade from ODK Aggregate 1.4.

Aggregate v1.4.2

1. Fixes to ODK Tables sync protocol for the combined release. Due to interactions with Google AppEngine, the ODK Tables Synchronization protocol does not currently work on Google AppEngine servers. You must *install a local server or a VM image* in order to use that mechanism.

Upgrade notes

If you were testing out ODK Tables, you should delete all tables before upgrading, as the database schema has changed.

Otherwise, this should be a seamless upgrade from ODK Aggregate 1.4.

Aggregate v1.4.1

1. You can now use Enketo browser-based Webforms to fill-in and publish submissions directly into ODK Aggregate. This feature was developed and donated by SDRC India. To enable Enketo integration, go to the *Preferences* sub-tab under *Site Admin* tab and click on *Enketo API Configuration*.
2. Fix the Z-ALPHA JSON publisher and the JSON File export to emit an array of zero or more objects, one object per submission, with proper treatment of embedded quotes, etc and confirmed that the output passes JSLint.
3. Fix the CVS File export functionality to double-up all occurrences of double-quotes in a field before surrounding that field with double quotes (per RFC 4180).
4. Clean up date and time handling in REDCap publisher and enforce GMT time zone interpretation when rendering date and time strings.
5. Various GWT interfaces have been changed to use concrete ArrayList types (reduces code size).
6. New permissions have been added in support of ODK Tables – Synchronize Tables and Administer Tables and for most ODK Tables interactions, the user is required to have Synchronize Tables permissions.
7. Extensive changes to ODK Tables sync protocol and database structures. There will be further changes in the next update.
8. Fix sizing calculations and CSS so that the ODK logo does not get clipped or overwritten.

Upgrade notes

If you were testing out ODK Tables, you should delete all tables before upgrading, as the database schema has changed.

Otherwise, this should be a seamless upgrade from ODK Aggregate 1.4.

Aggregate v1.4

1. Changed behavior: simple JSON publisher now POSTs an application/json entity body; added option for how to treat binary content. Incompatible change; see Upgrading section if you were using the Z-ALPHA JSON Server
2. Changed representation: ODK Tabled storage schema has been revised. Incompatible change; see Upgrading section if you used your server to upload or download data to ODK Tables. Syncing with ODK Tables is broken in this release. Only v1.3.4 on Google AppEngine works with the ODK Tables alpha (we are in the middle of changing table schemas).
3. Fix: file exports were not properly writing UTF-8 character sets.
4. Fix: PostgreSQL failures on some forms (column names must be less than 63 characters)
5. Upgrade Aggregate's Google publishers to use the updated Oauth2 libraries (e.g., Google Fusion Tables, Google Spreadsheet). Publishing from Google Apps domains does not work (it never has).
6. Add support for publishing data to Google Maps Engine.
7. Rework Spreadsheet and Fusion Table publishers to use Google libraries (consistent with the new Google Maps Engine publisher)
8. Additional active-paused state in publisher to extend the delay in publishing attempts (and reduce quota usage) if the destination publisher is reporting an error.
9. Improved how failures during form definition uploads are rolled back to increase the likelihood that the database is restored to a clean state.

Upgrade notes

- You may need to clear your browser cache to complete the upgrade. If the browser screen flickers after upgrading, first clear your browser cache and reload the page.
- If you were using the Z-ALPHA JSON Server, you must delete all instances of that publisher before upgrading. After upgrading, the updated publisher will send a single application/json entity body to the external server instead of a multi-part form containing that entity.
- If you were using ODK Tables with ODK Aggregate, you need to delete all the ODK Tables data on ODK Aggregate before upgrading.

Aggregate v1.3.4

1. Fix for the v1.3.3 fix for Google AppEngine – The original fix caused the creation of new publishers to Google Spreadsheets to fail, export to files to fail, form deletions to fail, and purge-sent-submissions actions to fail. This fix should rectify those issues.

Aggregate v1.3.3

1. Fix for Google AppEngine – entity keys (unique identifiers assigned by Google AppEngine and used internally by ODK Aggregate) may now contain slashes. Submissions that have been assigned an entity key containing a slash were breaking the 'SubmissionKey' parsing used when publishing, retrieving images, accessing repeat groups, or retrieving submissions using ODK Briefcase.

Aggregate v1.3.2

1. Expose the ODK Tables data and management tabs.
2. When installing for first time, ODK Aggregate will not require you to log in. Access restrictions are not altered when upgrading. This only affects new deployments. issue 710 (on Google Code) - upon an initial install, configure anonymousUser with Data Collector and Form Manager (and Data Viewer) permissions.
3. Watchdog sweep interval shortened to 30 seconds in fast-publishing mode (from 60 seconds). Ensure watchdog is scheduled to be fired when there are records remaining to be published.
4. Fix publisher failure to Google Spreadsheets and Fusion Tables by prepending 'n' to element names beginning with digits.
5. Fix publishing failure with Fusion Tables when a form with repeat groups has submissions without any repeats.
6. Fix publisher-creation failure that can cause cycling UI refresh.
7. Update javarosa library, adding format-date-time().
8. Update to selenium 2.33.0 to resolve Firefox ESR 17.0.7 failures.

Aggregate v1.3.1

1. Change watchdog to run more frequently if there is an active publisher. Provide a *disable* button on the Site Admin / Preferences page to restore older behavior (to conserve GAE quota).

Warning: Following issues arise while using this version:

- Form upload fails for some forms on MySQL with stack exhaustion.
- Fix to simple JSON publisher had caused instability when used.
- Popups don't show centered in screen when displayed on top of scrolling regions.
- Forms with repeat groups cannot be versioned.
- Rows-per-page value keeps getting reset on refresh.

Aggregate v1.3

1. Wholesale transition from OAuth 1.0 to OAuth 2.0, breaking all publishers.
2. Installer now supports migrated AppEngine instances (for the Master-Slave -to- High-Replication Datastore migration).
3. Google Fusion Tables publisher now provides the links to the tables of all the repeat groups, the top-level record, and a left-outer-join view of the first repeat group and top-level record. This gives a 'flat' view of the data.
4. Google Spreadsheets and Google Fusion Tables publishers are now using OAuth 2.0 for authentication. This breaks all existing publishers (you need to republish). For more details on this, see OAuth2-service.
5. Google Maps v3 API is now used for the visualization features.
6. Added a *Published Through* and an *Owner* column to the *Published Data* table to communicate the progress of the publisher and who is receiving the data.
7. JSON file export now exports multiple-choice values as a JSON array of string values, rather than a space-separated string.
8. MySQL media attachments are stored as BLOB types, allowing the default MySQL configuration to work.
9. forms are now listed alphabetically
10. New additions:
    - Alpha release REDCap (XML) publisher
    - Alpha release Simple JSON publisher
    - Alpha release Ohmage JSON publisher

Warning: Following issues arise while using this version:

- Google Spreadsheet publisher failed badly if name was blank.
- Extra comma in JSON file export (in repeat groups).
- Arbitrary intermingling of http and https requests are problematic
- Filters are not saved unless display metadata is checked.
- Title of existing form cannot be changed.

Upgrade notes

After the upgrade, ODK Aggregate needs to be configured with OAuth 2.0 credentials on the Site Admin / Permissions page. Once configured, you will then be able to create new publishers for your data (it is not possible to resume or restore publishing to the original publishers).

To avoid having to create new publishers that re-publish already-published data, follow these steps before upgrading:

1. Go to the *Form Management* tab.
2. Uncheck *Accept Submissions*.
3. Verify that all submissions appearing on the *Submissions* tab have been successfully published to Fusion Tables and Google Spreadsheets.
4. At this point, because ODK Aggregate is not accepting any new submissions, your surveyors are unable to send filled-in forms and we can be assured that no data is in transit during the upgrade process.
5. Deploy ODK Aggregate 1.3.0.
6. Go to the Publishers page, and create replacement publishers using *Stream New Submissions ONLY*.
7. Now go to Fusion Tables or Spreadsheets and copy the data from the v1.2 tables into the newly-created publisher tables.
8. Check *Accept Submissions* under the *Form Management* tab.
9. At this point, new submissions will stream into the new publishers and you have manually copied the old data into the new publisher, so these new publishers will now have all of your data.

Tip: The database tables for the new publishers and older publishers do not overlap, so if you roll back to the ODK Aggregate 1.2 release, you will not see the new publishers, but

the earlier 1.2 publishers will 'reappear.' If you want to, after the upgrade, in MySQL or PostgreSQL, you can drop the unused old tables:

- `_server_preferences` (has been replaced by `_server_preference_properties`)
- `_form_service_cursor` (has been replaced by `_form_service_cursor_2`)
- `_fusion_table` (has been replaced by `_fusion_table_2`)
- `_fusion_table_repeat` (has been replaced by `_fusion_table_repeat_2`)
- `_google_spreadsheet` (has been replaced by `_google_spreadsheet_2`)
- `_google_spreadsheet_repeat` (has been replaced by `_google_spreadsheet_repeat_2`)

Aggregate v1.2

1. Updated javarosa library with cascading select support (as with KoBo Collect).
2. Add a *Delete* button to the Exports list to enable deleting the generated files.
3. Exported files using filters now export the metadata if displayed by the filter.
4. Improved Map visualization display and pop-ups (showing images)
5. Improved filter, export and publishing pop-ups.
6. Update to use LONGBLOB and LONGSTRING on MySQL (new tables only).

Note: The use of LONGBLOB and LONGSTRING requires a configuration change in the MySQL server. The server requires the transmission packet size to be configured large enough to hold the largest LONGBLOB or LONGSTRING you will ever send to the server. See this for details.

Alternatively, after creating your tables, you can use the MySQL `ALTER TABLE` command to change the LONGBLOB field to a BLOB field. This was the pre-1.2 setting, and will be the 1.2.1 setting for image fields, returning the system to use 65kB image chunks and avoiding the need to change the server configuration. If you do this, you will need to stop and restart your ODK Aggregate server for the change to be detected and take effect.

---

7. Cache thumbnail images for 1 hr for improved performance and lower AppEngine datastore usage.
8. Form definition files and media attachments can now be altered and those changes uploaded to the ODK Aggregate server. The server still maintains only one version of the form, and all alterations must not affect the number of questions in the form or change the data type of any field (e.g., from int to decimal or string, etc.).

9. Whenever a form or any of its media files are modified, the version attribute in the top-level element (where the form id is defined) must be changed.

Tip: Version attributes are recommended to be of the form "yyyymmddnn", e.g., 2012060400 – the last two digits are the form iteration within the given day. They must be integer values and the new value must compare lexically greater than the prior value (this means, for example, since "9" compares lexically greater than "10", you cannot update a version from 9 to 10 – but you could upgrade from "09" to "10").

10. There is a 15-minute grace period for uploading revisions after which the version must be incremented (e.g., incremented to 2012060401).
11. Fix odd start-up failures on Google AppEngine.

Upgrade notes

Existing 1.0 installations can upgrade to the 1.2 release, but, once upgraded, if you use the new form-versioning feature, these installations cannot downgrade from 1.2 to the earlier 1.0 releases.

MySQL and PostgreSQL require special upgrade instructions. The `_filter_group` table has a new column. If you are running ODK Aggregate 1.0 or 1.1, you will need to issue an `alter table` command on this table to add this column.

For MySQL:

```
ALTER TABLE `_filter_group` ADD COLUMN `INCLUDE_METADATA` char(1) NULL;
```

For PostgreSQL:

```
ALTER TABLE "_filter_group" ADD COLUMN "INCLUDE_METADATA" boolean NULL;
```

Depending upon your database management tool, you may need to qualify the table name with the schema.

General steps for v1.1 to v1.0.4

For any Aggregate version from v1.1 to v1.0.4, no additional upgrade steps are required if none of your forms use **odk:length** to alter the maximum string length.

Otherwise, for ONLY those forms that use **odk:length**:

1. Download the forms that use odk:length using the ODK Briefcase application.
2. Delete the forms on Google AppEngine (this may take several minutes or hours if you have many submissions).

3. Upload the forms from ODK Briefcase back onto your Google AppEngine instance.

odk:length has always been respected on MySQL and PostgreSQL; there are no additional steps to be performed on those systems.

General steps for version v1.0.3 to v1.0

For any Aggregate version from v1.0.3 to v1.0, if you are running RC4, RC3, RC2, or RC1 (on Google's AppEngine or with MySQL or PostgreSQL), there are no special upgrade steps. It should just work.

If you are using Beta 4, follow these upgrade instructions:

On GAE: No additional steps required. It should just work.

On MySQL or PostgreSQL: The persistent results tables that hold the exported csv and kml files have changed (adding support for emitting csv files filtered by the active filter).

1. Stop Tomcat.
2. In your database's administration tool, connect to the database and:
   - `drop table _persistent_results;`
   - `drop table _persistent_result_file_bin;`
   - `drop table _persistent_result_file_ref;`
   - `drop table _persistent_result_file_blb;`
3. Copy the new WAR to the `/webapps` directory.
4. Start Tomcat.

For all other Alpha or Beta releases, you must either use a new appspot instance or delete all data in your appspot instance. If you are using MySQL or PostgreSQL with an Alpha or Beta release, you should start with an empty database.

General steps to upgrade from Alpha and Beta releases

On GAE

1. If running builds prior to Beta 2 on Google AppEngine cloud services, any forms with Decimal data will need to be deleted and reloaded. Decimal data had been represented as strings and is now represented properly as double-precision numbers in Google AppEngine.
2. Disable writes to the datastore (via Application Settings / Disable Datastore Writes)
3. Deploy to GAE.

4. Delete all the records in these kinds (using Datastore Viewer):
   - opendatakit._granted_authority_hierarchy
   - opendatakit._registered_users
   - opendatakit._user_granted_authority
5. Disable your application via Application Settings / Disable Application.
6. Enable writes.
7. Enable your application.

On MySQL or PostgreSQL

1. Stop Tomcat.
2. In your database's administration tool, connect to the database and:
   - `drop table _granted_authority_hierarchy;`
   - `drop table _registered_users;`
   - `drop table _user_granted_authority;`
3. copy the new WAR to the `/webapps` directory.
4. Start Tomcat.

Warning: The primary upgrade impact is the loss of all registered users and their privileges (what the manual steps above do). Beta 3 changes the user configuration. Registered users are now either Gmail accounts or ODK Aggregate usernames. ODK Aggregate now supports only anonymous access and/or registered users.

7.3.5 Migration from ODK Aggregate 0.9x to 1.x

Note:

- Aggregate 0.9.x has a 1000-record limit to the number of records it can display (this is by design); it does, however, still retain all records i.e. it will hold as many records as you've uploaded and does not loose or ignore the 1001st record, etc; in general, any webserver will have a limit to what it can display interactively (at some point it will run out of memory or time out).
- On Aggregate 0.9.x, we provide the Briefcase applet (one of the links on the top row) to allow you to extract your data from the server into a local CSV file for local

processing. That should retrieve all data within the server, and should work regardless of the number of submissions you have uploaded.

- Aggregate 1.0 has far fewer display restrictions and has implemented display paging on the submissions display page so you can page through all the submissions on the server, rather than just the first 1000 (and you get to set the page size, as well).
- Aggregate 1.x has stricter requirements on the form definition than 0.9.x, so you will likely need to edit your form definitions before they can be successfully uploaded into Aggregate 1.x.
- Migration will necessitate creating a new AppEngine application id and migrating your ODK Collect devices to use that new AppEngine URL. This allows you to keep the old Aggregate 0.9.x instance should you need to return to it (e.g., if there is an error discovered after data migration), and it gives you time to validate the accuracy of the data that was migrated. It is simply safer to do the migration this way rather than overwrite your older instance.

Since Google is now charging for AppEngine usage, the lowest-cost sequence for the AppEngine migration is as follows:

1. Upgrade ODK Aggregate 0.9.x to the latest 0.9.x release. If you have any 0.9.x release, you can upgrade to the latest 0.9.x release without any changes (the newer updates are backward compatible). Get the latest Aggregate release from the ODK Downloads page here.
2. Create a new application id for the 1.x instance.
3. Download the latest installer for ODK Aggregate 1.x, run it, and deploy to the new application id. For installation process, see the *Aggregate Installation Guide*.
4. Manually download the form definitions (as XML) from Aggregate 0.9.x. Browse to your Aggregate 0.9.x instance and choose the *View XML* button on the main forms page. Then, from the resulting page, choose the *Download XML* button. Repeat for every form you have on the system.

Tip: You may wish to specify a new folder for the downloads of your forms so that you can easily locate and edit them.

---

5. Manually upload the form definitions to Aggregate 1.x, editing and fixing the form definitions as needed. First sign into your Aggregate server. Go to the *Forms List* sub-tab under *Form Management* tab. Click the *Add New Form* button. Select the form to upload, and click on *Upload Form*. Repeat with the remaining forms.

Warning: Following issues may arise in form upload process:

- Your form definition may be using *xmlns="myformid"* to define the Identifier (Aggregate 0.9.x) now referred to as the Form Id (Aggregate 1.x) of the form. In Aggregate 1.x, we enforce that all xmlns assignments conform to URL syntax. This is a w3c standard and a requirement of some Xml parsers and editing systems. Since most users have just a simple short alphanumeric id for their forms, the fix is to change from using xmlns to using id to define your "form id". E.g., use *id="myformid"* instead. This variant allows "myformid" to be any alphanumeric string with additional non-space characters (e.g., dashes and underscores).
- Aggregate 1.0.1 warns you if you have fields in the form that do not have defined data types. I.e., they lack any <bind> expressions or their <bind> expression does not specify a type="datatype" expression, with datatype being one of the known data types. This is a concern because Aggregate can filter results based on the values of individual columns. The ranking order for the string "111" would place it before (less than) "13", but that would be incorrect if this field were actually an integer value. The warnings are to catch omissions like this so that filters can work properly and so that data can be properly published to Fusion Tables.

6. Configure the Aggregate 1.x instance to accept submissions from ODK Collect. If you are running ODK Collect 1.1.5, you will need to configure Aggregate 1.x to grant the anonymousUser the Data Collector privilege. This allows unauthenticated users to submit data to Aggregate 1.x. If you are running ODK Collect 1.1.7 or higher, you can choose to define one or more local usernames and passwords, and configure ODK Collect 1.1.7 to identify itself with one of those usernames and passwords when accessing the server. Steps to do either of this are as follows:
   (a) Go to the *Permissions* sub-tab under the *Site Admin* tab.
   (b) If you are creating new local user(s), type the username(s) in the text box in the Add Users section of the page and click *Add*. You can add multiple users by separating the usernames with spaces or commas. The table above will update with the added users. Ensure that the Account Type is *ODK* for each of these users.
   (c) Check the *Data Collector* check box beside each username that will be used by ODK Collect. If you have ODK Collect 1.1.5, check the check box beside the anonymousUser.
   (d) Click *Save Changes* to apply your changes.
   (e) Then, for each username you created, you must click *Change Password* and set the password for that username.
7. Configure all your ODK Collect clients to use the new Aggregate 1.x instance. On both ODK Collect 1.1.5 and 1.1.7, ensure that the Server URL points to the new instance

and begins with *https* (for AppEngine instances). For MySQL or PostgreSQL systems, specify *https* only if the server has an SSL certificate. For ODK Collect 1.1.7, if you are requiring authentication, enter one of the configured usernames and passwords into the phone.

8. *Install* the latest ODK Briefcase application.
9. Run ODK Briefcase, choosing to *pull all forms* from your 0.9.x instance to the local server. You will need enough space to store these files.
10. For any form definitions you have changed, manually overwrite the downloaded form definition within the Briefcase directory with the one updated for Aggregate 1.x. The ODK Briefcase Storage Area has a `/forms` directory under it, and, in that, it has directories for each form downloaded from Aggregate 0.9.x. The form definition is within that directory and must have the same name (with a file extension of .xml) as the directory containing it.
11. Run ODK Briefcase, choosing to *push all forms* from your local server to the new Aggregate 1.x instance.
12. Validate that your Aggregate 1.x instance has all the data you expect, and the values are complete and correct (by, e.g., publishing to Fusion Tables and comparing this dataset to the corresponding dataset produced by 0.9.x).
13. Once you are comfortable with the operations of the Aggregate 1.x instance, you should go to the Aggregate 0.9.x dashboard and delete that application id.

1. Click on the *Application Settings* link on the left sidebar under the Administration heading.
2. Click on the *Disable Application..* button on this page.
3. Choose to disable it, and then, when prompted, choose to delete the application.

This is more cost-effective than deleting the forms (and data) from the old instance and trying to reuse it as that would incur usage charges. Deleting the entire instance accomplishes that without running up any charges.

Tip: If you have to do this migration mid-survey and need to be able to incrementally pull data from ODK Aggregate v0.9.x, you should follow these steps:

1. Pull data from ODK Aggregate v0.9.x
2. Push data to ODK Aggregate v1.x
3. Pull data from ODK Aggregate v1.x
4. Repeat periodically.

The extra pull of data down from ODK Aggregate v1.x sets a flag within ODK Briefcase that identifies the record as being "complete" (having all of its attachments). This does

not prevent ODK Briefcase from downloading all the data each time from ODK Aggregate v0.9.x, but it does enable it to avoid repeatedly uploading this data to ODK Aggregate v1.x.

7.3.6 Downgrade steps

On April 12, 2016, Google disabled the login mechanism used by the older update scripts that were packaged with these installers.

To use these older installers:

1. Download and run an ODK Aggregate 1.4.8 (or newer) installer. On the final screen, uncheck the *Launch upload tool (3-10 minutes)* checkbox and click *Finish*.
2. Go into the directory where the installer placed the files, go into the `ODK Aggregate` directory, and rename the `ODKAggregate` directory inside it to *NewRemoval*.
3. Download and run the pre-1.4.8 ODK Aggregate installer. On the final screen, uncheck the *Launch upload tool (3-10 minutes)* checkbox and click *Finish*.
4. Go into the directory where the pre-1.4.8 installer placed its files, go into the `ODK Aggregate` directory, and copy the `ODKAggregate` directory from there to the installer directory created by the newer installer (effectively replacing the configuration produced by the newer installer with the configuration produced by the older one).
5. In the directory created by the newer installer, if on:
   - *Windows*: double-click the `ODKAggregateAppEngineUpdater.jar` and use that to update your App Engine instance.
   - *Mac OSX*: double-click the `uploadAggregateToAppEngine.app` file in that directory.
   - *Linux*: open a bash shell, navigate to this directory, and run the `uploadAggregateToAppEngine.sh` file.

Note: If you are reverting from a 1.4.8 or later release to a 1.4.7 or earlier release, you must manually delete the background module of your App Engine using the Google Cloud Platform administration web pages.

7.4 Backing Up Aggregate

7.4.1 Recovering from Briefcase

Use *ODK Briefcase* to back up all forms and submissions on Aggregate. You can *pull forms from your Aggregate server* into your local machine using Briefcase. In particular, Briefcase's *command line interface* makes this easier.

7.4.2 Recovering database from MySQL dump

1. Stop running Tomcat.
2. *Upgrade to the latest version of Aggregate*.
3. Finally, restore it from MySQL dump. An SQL dump of a database is a common method to safely store away a snapshot of the database for archival purposes or to migrate data between database instances, e.g. between two major system releases. The content of a SQL dump is a large collection of SQL commands in ASCII. Running the script will recreate the database in the same state as it was when the dump was created. The primary tool to consider for making an ASCII dump is mysqldump, which includes a wide variety of options.

```
$ mysqldump [ options ] [ dbname ]
```

Some of the useful options are:

- `-h hostname` or `--host=hostname` specifies host to connect to.
- `-p portnr` or `--port=portnr` specifies port to connect to.
- `-u user` or `--user=user` specifies user id.
- `-d database` or `--database=database` specifies database to connect to.

To take a backup of database:

```
$ mysqldump database > backup-file.sql;
```

To restore a database:

```
$ mysql database < backup-file.sql;
```

To copy a database from one server to another

```
$ mysqldump --opt database | mysql --host=remote_host -C database
```

remote_host indicates a remote server where you want to take backup.

Note: Creation of the dump respects your credentials, which means you only can dump the tables you have access to.

7.4.3 Backup and recovery on Google App Engine

Create backup

1. Open a browser to Google Cloud Platform and click on *Console* in the top right corner.

2. Sign in with a Gmail account which you used for Aggregate installation.

3. Choose the project id for your ODK Aggregate server by clicking on the project drop-down in the top left corner.

4. Click on the menu icon (three horizontal bars) to the left of *Google Cloud Platform* in the upper left side of the screen and then select *Datastore* from the menu. Click on *Admin* in the dropdown.

5. Enable Cloud Datastore Admin access by clicking on *Enable Datastore Admin*.

6. Then, click on *Open Datastore Admin*.

Tip: It is recommended to disable writes during creation of backup. To disable writes, click on *Disable writes* on the Admin page.

Warning: Your Aggregate server may become unstable when you disable writes. It will be fine when you enable writes again.

7. Select the entity kinds that you wish to back up and then click on *Backup Entities*.

opendatakit._form_data_model	Stats not available	Stats not available	Stats not available	Stats not available
opendatakit._form_info	Stats not available	Stats not available	Stats not available	Stats not available
opendatakit._form_info_fileset	Stats not available	Stats not available	Stats not available	Stats not available
opendatakit._form_info_submission_association	Stats not available	Stats not available	Stats not available	Stats not available
opendatakit._form_info_xform_bin	Stats not available	Stats not available	Stats not available	Stats not available
opendatakit._form_info_xform_blb	Stats not available	Stats not available	Stats not available	Stats not available
opendatakit._form_info_xform_ref	Stats not available	Stats not available	Stats not available	Stats not available
opendatakit._granted_authority_hierarchy	22	381 Bytes	8 KBytes	67 KBytes
opendatakit._misc_tasks	Stats not available	Stats not available	Stats not available	Stats not available
opendatakit._registered_users	1	519 Bytes	519 Bytes	5 KBytes
opendatakit._security_revisions	4	267 Bytes	1 KByte	8 KBytes
opendatakit._server_preferences_properties	2	344 Bytes	689 Bytes	6 KBytes
opendatakit._user_granted_authority	2	361 Bytes	723 Bytes	6 KBytes

Backup Entities Delete Entities

8. A backup form will be displayed.

Datastore Admin: Backup to Google Cloud Storage

Backup name:
datastore_backup_2017_10_25

Selected namespace: --All--

Operation will be performed using the selected queue: default

Backup storage destination: Google Cloud Storage

Google Cloud Storage bucket name (More info?) project-edu-181114.appspot.c

Backup Entities or Cancel

Note:

- A backup name is supplied and it includes a datestamp. You must change this value if you make more than one backup per day because a backup is not made if a backup of the same name already exists.
- The default queue is used for the backup job; you can use this in most cases. If you use a non-default queue for backup/restore, you can only specify the target **ah-builtin-python-bundle** in **queue.yaml**. You cannot use any other targets. To know more about queues, see this.
- Select Google Cloud Storage as the backup storage location.

In the bucket name box, enter *your-project-id.appspot.com*. You can alternatively preface the bucket name with **/gs/**, for example, /gs/[BUCKET_NAME].

Note: Buckets are containers where your backup will be stored. You can also create buckets for your project.

Now click on *Backup Entities* to start the backup jobs.

9. A job status page is displayed. Click on *Back to Datastore Admin* to see the backup status.

You can abort a backup by selecting a backup from the list of pending backups and clicking on *Abort*.

Warning: When you abort a backup job, App Engine attempts to delete backup data that has been saved up to that point. However, in some cases, some files can remain after the abort. You can locate these files in the location you chose for your backups in Google Cloud Storage and safely delete them after the abort completes. The names of such files

start with the following pattern: **datastore_backup_[BUCKET_NAME]**.

Click on *Info* to get more information about the backup. On the info page, click *Back to Datastore Admin* to return to the main Cloud Datastore Admin screen.

Tip: After the backup is complete, if you disabled Cloud Datastore writes, re-enable them by going to Admin page and clicking on *Enable writes*.

Restoring data from Backup

1. Go to the Admin page as described in the *creation of backup*.

Tip: Disable Cloud Datastore writes for your application. It's normally a good idea to do this to avoid conflicts between the restore and any new data written to Cloud Datastore. To disable writes, click on *Disable writes* on the Admin page.

2. Click on *Open Datastore Admin*.

3. In the list of available backups, select the backup that you want to restore from and click on *Restore*. You can click on *Info* to get more information about the backup. To delete a backup, select the backup and click on *Delete*.

4. In the advisory page that is displayed, notice the list of entities with checkboxes. By default, all of the entities will be restored. Uncheck the checkbox next to each entity that you don't want to restore. Click on *Restore* at the bottom of the page to start the restoration.

Note: In the advisory page, notice that the default queue, with its pre-configured performance settings, is used for the restore job. Change this to another queue that you have configured differently if you need different queue performance characteristics, making sure the queue chosen does not have any target specified in **queue.yaml** other than **ah-builtin-python-bundle**. To know more about queues, see this.

5. A job status page is displayed. Click on *Back to Datastore Admin* to see the status of the restore.

Restore from backup Job Status

The following jobs were launched by MapReduce.

Each job's status will only be available for as long as the job takes to complete. Once the job is complete, it will remove the status objects from your datastore. Click a job's id for status information.

Restore from backup job with id 1567092070921236207917708 1EC0A5 kicked off.

Back to Datastore Admin

Active Operations

Description	Jobs
Restoring opendatakit._form_info_xform_bin, _STRING_FIELD_LENGTHS_, _COST_LOGGING_, opendatakit._form_data_model, opendatakit._granted_authority_hierarchy, _ah_SESSION, opendatakit._security_revisions, opendatakit._form_info_submission_association, opendatakit._form_info, opendatakit._user_granted_authority, opendatakit._server_preferences_properties, opendatakit._registered_users, opendatakit._backend_actions, opendatakit._misc_tasks, opendatakit._form_info_fileset, opendatakit._form_info_xform_blb, opendatakit._form_info_xform_ref from backup: datastore_backup_2017_10_25	(0 steps completed, 1 active) Job 1567092070921236207917708 1EC0A5

Completed Operations

Description	Completed steps	Status	Status info
Backup: datastore_backup_2017_10_251234	1 steps	Completed	
Backup: datastore_backup_2017_10_25123	1 steps	Completed	
Restoring opendatakit._form_info_xform_bin, _STRING_FIELD_LENGTHS_, _COST_LOGGING_, opendatakit._form_data_model, opendatakit._granted_authority_hierarchy, _ah_SESSION, opendatakit._security_revisions, opendatakit._form_info_submission_association, opendatakit._form_info, opendatakit._user_granted_authority, opendatakit._server_preferences_properties, opendatakit._registered_users, opendatakit._backend_actions, opendatakit._misc_tasks, opendatakit._form_info_fileset, opendatakit._form_info_xform_blb, opendatakit._form_info_xform_ref from backup: datastore_backup_2017_10_25	1 steps	Completed	
Backup: datastore_backup_2017_10_25	17 steps	Completed	
Restoring opendatakit._form_info from backup: datastore_backup_2017_10_25	1 steps	Completed	
Restoring opendatakit._form_info from backup: datastore_backup_2017_10_25	1 steps	Completed	
Restoring opendatakit._form_info from backup: datastore_backup_2017_10_25	1 steps	Completed	

Tip: After the restore is complete, if you disabled Cloud Datastore writes, re-enable them by going to Admin page and clicking on *Enable writes*.

Note:

- If you back up your data using Google Cloud Storage, you can restore backups to applications other than the application used to create the backup. To restore backup data from a source application to a target application, see this guide.
- Google has new beta service for exporting and importing. Only Cloud Platform projects with billable accounts can use the export and import functionality.

CHAPTER

EIGHT

USING ODK AGGREGATE

8.1 Managing Forms in Aggregate

You can add, download and delete forms, export data into useful formats, publish data into another service, manually upload submissions and view the published data. ODK Aggregate provides all these options under the *Form Management* tab.

8.1.1 View blank form list

Click on *Forms list* tab to see a list of all your forms.

From here, you can add new forms, delete forms, download forms as well as restrict submissions for a form.

Blank form info in list

In addition to the functional columns described below, the form list displays the following information about each form.

- *Form Id* is the unique name for the form.
- *Media Files* is the count of media files associated with the form.
- *User* is the user who uploaded the form.

8.1.2 Adding New Forms

Under the *Form list* tab, you will see *Add New Form* button to upload a new form definition to ODK Aggregate.

When you click on it a box will open asking for details of the form. *Form Definition* is required and *Media File(s)* is optional. Choose the `.xml` file that will be used. You can also choose the appropriate media files for the form.

8.1.3 Viewing and downloading forms

From the *form list*, click on form's *Title* to view the XForm (the actual XML file). You can then download the file by clicking on *Download XML* in the Form XML Viewer.

8.1.4 Disabling or enabling downloading of a form

To disable or enable the ability of Collect or other clients to download forms, toggle the checkbox in the *Downloadable* column of the *blank forms list*.

8.1.5 Disabling or enabling submission of a form

To disable or enable the ability of Collect or other clients to submit completed instances of a form, toggle the checkbox in the *Accept Submissions* column of the *blank forms list*.

8.1.6 Deleting a blank form

Click on the *Delete* button for the form in the *blank forms list*.

8.1.7 Exporting form data

Click on *Export* option in the form list to export form into useful formats and choose the format in which you want to export data. You can also choose a filter which you have saved for the form to export only a certain subset of form. Details on *exporting data* are given in the *data transfer* section.

8.1.8 Publishing form data

Click on *Publish* option in the form list to publish the form into another service. You can choose where you want to publish data and which data you want to publish. Details on *publishing data* are given in the *data transfer* section.

Viewing published data

You can get a view of the published data you have created for a particular form by clicking on *Published Data*.

- Select the form corresponding to the published data in the *Form* dropdown.
- Read the message that appears and click on *Purge Published Data*.
- *Created By* shows the email of the user who created the published file.
- *Status* can be *ACTIVE* (the file is ready to view) or *ESTABLISHED* (something went wrong in the process of exporting.)
- *Start Date* shows the time when you finished filling out the *Publish* form.
- *Action* is based on your selection of upload only, stream only, or both in the *Publish* form.
- *Type* shows the type you choose to publish your data to.
- *Owner* shows the owner of the published data.
- *Name* is the place where you published your data. If the type was a Google Fusion Table, click on the link to view the Fusion Table.
- Select delete box in the *Delete* column if you want to delete your published file.

8.1.9 Managing submissions manually

You can manually upload submissions for a form and check incomplete submissions under the *Submission Admin* tab.

Submitting forms directly

To manually upload form submissions, click on *Manually upload submission data*.

Note: Submissions in the Collect app are located under the /odk/instances directory on the phone's sdcard. This directory will contain subdirectories with names of the form in the format `formID_yyyy-mm-dd_hh-MM-ss`. Within each of these subdirectories are the submis-

sion data file (`formID_yyyy-mm-dd_hh-MM-ss.xml`), and zero or more associated media files (images, audio, video) associated with this submission.

Note: If you upload a submission, but fail to upload all media attachments, it places the submission in the incomplete submissions bucket. While it resides there, it won't be published to external servers or downloadable via ODK Briefcase.

Removing form submissions

To remove a form submission, select the form in the *Form* dropdown and click on *Purge Submission Data*.

Incomplete form submissions

To see a list of incomplete submissions for a particular form under the *Incomplete Submissions* list.

8.2 Working with Submitted Data in Aggregate

You can view the data submitted from ODK Collect here. You can filter the submissions, visualize them using pie chart, bar graph or map, export the submissions into useful formats and publish the submitted data into another service. You can also view all the exported submissions. ODK Aggregate provides all these options under the *Submissions* tab.

8.2.1 Filtering Submissions

Submissions from ODK collect can be filtered to view or hide a specific subset of data by creation of filters. Filters give you the ability to see a subset of your data. You can have a single filter as well as multiple filters. If you have multiple filters applied at once, then you have a filter group. You can create and apply filters by using the options under the *Filter Submissions* tab.

You can create a single or multiple filters depending on the subset of data you want to view or hide. Creating a filter like *Display Rows where column Gender EQUAL male* specifies that you want to get a list of all rows where gender column has value as male i.e you want to obtain information about all male people in your data. Unless the filter is saved, it is

temporary. You can save a filter to make it permanent. Any filter can be deleted if it is no longer needed.

Various options under this tab can be used as follows:

- Click on *Add Filter* to add filter to the data. In the *Create filter to* dropdown, *Display* or *Hide* will specify whether you will be selecting data to show or hide and *Rows* or *Columns* will specify whether you will be working with the rows or columns of the table.

If you select *Rows* specify a condition you want to apply in the *where* box.

If you selected *Columns* specify the columns you wish to display or hide in the *titled* box.

- Click on *Save* to save the filter or filter group for future use. Clicking on *Save As* allows you to give a name to the filter or filter group.
- Click on *Delete* to delete a filter or filter group.

- You can check the *Display Metadata* checkbox to display or hide metadata.

Note: Metadata provides information about the submissions. There will be information such as date submitted, if the data is complete, version numbers, and id numbers.

8.2.2 Visualizing Submissions

ODK Aggregate provides a quick means for basic data visualization. This Visualize functionality is meant to provide a quick means to view early data results in meaningful ways but is not meant to provide complex data analysis functionality. You can view your data in bar graph, pie chart or on a map.

In both Pie chart and bar graph visualization you can either count the number of times a unique answer occurs in a specified column or calculate sum of values in one column grouped by a value in another column. You can choose a column that you want to map in map visualization.

Click on *Visualize* to visualize the submitted data. Select bar graph, pie chart or map in the *Type* dropdown. Further options are described as follows:

- If you choose Pie Chart, choose whether you would like to count or sum data:
  - If you select *Count* option, then select the column in which you want to apply this.
  - If you select *Sum* option, then select the column of values that you want to add and another column that you want to use to group the numbers.
  - Then click on *Pie It* to get the Pie Chart.

- If you choose Bar Graph, you have the same options as that in case of Pie Chart. Choose the option you want to use and then click on *Bar It* to get the Bar Graph.

- If you choose Map, select a column that you want to map in the *GeoPoint to Map* dropdown. Click on *Map It* to get the map. You can click on a point to view a balloon with the other information supplied in the table.

8.2.3 Exporting Submissions

Click on *Export* option to export submitted data into useful formats and choose the format in which you want to export data. You can also choose a filter which you have saved earlier to export only a certain subset of data. Details on *exporting data* are given in the *data transfer* section.

8.2.4 Publishing Submissions

Click on *Publish* option to publish the submitted data into another service. You can choose where you want to publish data and which data you want to publish. Details on *publishing data* are given in the *data transfer* section.

8.2.5 Viewing Exported Submissions

You can to view the list of exported files under the *Exported Submissions* tab.

- *File Type* specifies whether file is `.csv` or `.kml` or `.json` file.
- *Status* will state whether the file being made is in progress, or is now available for viewing.
- *Time Completed* shows the time when the *Export* task is complete and the file is ready.
- Click on the link in *Download File* to see your exported file.
- Select delete box in the *Delete* column if you want to delete your exported file.

8.3 Administering Aggregate

Click the *Log In* link in the upper right corner of the screen to be presented with the Log onto Aggregate screen. Choose the *Sign in with Aggregate password* button and enter the super-user username you specified within the installer. The initial password for this account is *aggregate*. When you log in, *Site Admin* will be visible to you.

Tip:

- When signing in with this method, if you do not enter the password correctly, you may need to close all your browser windows and quit your browser before you can try again.
- If the instance name of the server changes (the installer asks for this name), then the passwords for all ODK Aggregate usernames will be cleared (preventing their use) and the super-user username's password will be reset to aggregate and the above message will also be displayed. In this case, you should log in, change the super-user's password, and change the passwords for all of your ODK Aggregate usernames.
- In April 2015, the use of Google e-mail accounts for accessing the site (via Sign in with Google) stopped working (Google turned off that functionality). If you have an existing site running an old version of ODK Aggregate that does not have ODK Aggregate usernames configured for website access (and offers a Sign in with Google sign-in choice), you will need to upgrade to regain access to it.

Note: Beginning with ODK Aggregate 1.3.2, upon the initial installation of the server, it is configured to allow unauthenticated (*anonymousUser*) submissions from ODK Collect and unauthenticated browser access to the submissions and forms management functionality of ODK Aggregate.

8.3.1 Permissions

If you have not yet changed your super-user password to something other than *aggregate*, the server will display **This server and its data are not secure! Please change the super-user's password!** at the top of the web page.

Please visit the *Permissions* sub-tab under the *Site Admin* tab to change this user's password.

You can specify additional usernames with browser access to the server under the *Permissions* sub-tab. Privileges are as follows:

- **Data Collector**: able to download forms to ODK Collect and submit data from ODK Collect to ODK Aggregate.

Note:

- Only ODK accounts and the anonymousUser can be granted Data Collector rights.
- The anonymousUser must be granted Data Collector rights to accept submissions from unidentified sources (e.g., from ODK Collect 1.1.5 and earlier, or from ODK Collect 1.1.7 and later if not authenticating).

- **Data Viewer**: able to log onto the ODK Aggregate website, filter and view submissions, and generate csv or kml files for download.
- **Form Manager**: all the capabilities of a Data Viewer plus the abilities to upload a form definition, delete a form and its data, and upload submissions manually through the ODK Aggregate website.
- **Site Administrator**: all the capabilities of a Form Manager plus the ability to add users, set passwords, and grant these capabilities to other users.

Remember to click *Save Changes* to make these changes take effect. You can also edit the privileges for current users.

You can upload a `.csv` file of users and their capabilities as well as download the current file.

8.3.2 Preferences

In the *Preferences* sub-tab under *Site Admin* tab, you can manage:

- Google API credentials: These credentials are used when publishing into Google services. For details on this, see Aggregate OAuth2 info.

- **Enketo credentials**: These credentials are used for Enketo webforms integration. To link Enketo with Aggregate, see this.
- **ODK 2.0 App name**
  - *ODK Tables Synchronization Functionality* - check this to enable ODK Tables functionality i.e. able to download, upload and alter data records within ODK Tables as restricted by table-access privileges granted to the user.
- **Aggregate features**: These settings affect the operations of the server.
  - *Disable faster background actions* - check this to reduce AppEngine quota usage.
  - *Skip malformed submissions* - check this to ignore corrupted submissions.

8.4 Getting Data Out of Aggregate

ODK Aggregate supports three primary mechanisms for data transfer:

- *Exporting*: one time snapshot
- *Publishing*: continuous streaming of submissions
- *APIs*: programmatic access to the data

Since ODK supports complex data structures such as question grouping, repeating questions, and multimedia, compromises have been made for each mechanism in regards to these data structures. This document explains each mechanism and what each supports.

8.4.1 Exporting

The easiest way to get data from Aggregate is by using its 'Export' feature. Export allows the user to manually export all of the data (or filtered data) at any time into one of the following formats:

CSV (comma separated values)

CSV files are a text only, tabular representation of the data. Multimedia files are represented by including standard web links to the files. Repeats are also represented with links to the underlying data. Grouping information is not preserved. Metadata is only preserved if a filter is created with the metadata.

To download CSV files of forms with repeats, consider using ODK Briefcase instead of ODK Aggregate's export functionality. Instead of repeats that are represented with links to the underlying data, ODK Briefcase will export a set of CSV files, one for each repeating group. ODK Briefcase will also export any multimedia as files (e.g., pictures will get exported as JPEGs). The only metadata that is preserved is the submission date in the CSV is preserved, but the XML files that ODK Briefcase downloads has all form metadata.

Note: 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.

JSON

JSON is a text only representation of the data in a key:value format. Multimedia files are represented by including standard web links to the files. Repeating data is preserved, but grouping information not related to repeats is not. Metadata is only preserved if a filter is created with the metadata.

KML (Keyhole Markup Language)

KML is a text only representation of the data similar to XML, but used for mapping applications like Google Earth. Multimedia files are represented by including standard web links to the files hosted on the server. Pictures will appear embedded in popup windows (when pushpins are clicked) in programs that render .kml files. Repeats and grouping information is not preserved. Metadata is only preserved if a filter is created with the metadata.

Table 8.1: **Export Data Summary**

Format	Groups	Repeats	Multimedia	Metadata
CSV(Aggregate)	not preserved	links to	links to	preservable
CSV(Briefcase)	not preserved	split into separate CSVs	exported as files	only submission date
JSON	not preserved	preserved	links to	preservable
KML	not preserved	not preserved	links to	preservable

8.4.2 Publishing

Aggregate provides mechanism for either bulk publishing data to another service, or for streaming incoming data to a service as it is received, or both.

Warning: Under certain failure conditions, the downstream service can receive multiple copies of a given submission. This is known, expected, behavior.

Tip:

- Duplicates typically occur if the downstream service is slow to respond or acknowledge a request.
- It is your responsibility to detect and eliminate these duplicates should they occur (they will always have exactly the same information in all fields).

Note:

- If ODK Aggregate aborts its publishing attempt before it gets an acknowledgment, it will re-send the submission a short time later. If the downstream server successfully processed the first request, the re-send of that same submission can cause a duplicate record of it to appear in the downstream system. The instance ID of the submission will appear as the metainstanceid column in Google Sheets and as the *meta-instance-id* column in Google Fusion tables.
- When the downstream server fails to respond or responds with an error code, ODK Aggregate first delays a re-send for 60 seconds, and, if that also fails, it then backs off its publishing attempts, and will re-send at either 7.5-minute or 15-minute intervals

until the downstream service successfully responds, after which ODK Aggregate will resume its normal publishing behavior.

Aggregate currently supports publishing data to the following services or in the following formats:

Google Fusion Tables

Fusion Tables is an experimental data visualization web application to gather, visualize, and share larger data tables. Multimedia files are represented by including standard web links to the files. Repeating groups of questions are split into separate tables. All metadata is preserved. Fusion Tables is hosted in Google's cloud infrastructure. We know of no row limit, but one may exist. Fusion Tables also has an API that can be used to export/publish data.

Warning: Non-repeating grouping information is not preserved. This implies that if your form has a non-repeating group *purchaser* and a second non-repeating group *supplier* and within each of these groups, you have a *name* field, then when you publish to Fusion Tables, there will be two columns called *name*. Unfortunately, that badly confuses Fusion Tables, and generally results in data not transferring successfully into Fusion Tables.

The solution is to this problem is to design your forms with field names such as *purchaser_name* and *supplier_name*. In other words, use unique names throughout your form.

OAuth2 Service Account Required

Publishing data to Google Fusion Tables requires an oauth2-service.

Google Spreadsheets

Spreadsheets is Google's cloud-hosted spreadsheet solution. Multimedia files are represented by including standard web links to the files. Repeats are also represented with links to the underlying data. All metadata is preserved. Spreadsheets has a 400,000 cell limit, above which errors will be reported. Spreadsheets also has an API that can be used to export/publish data.

Warning: As with Google Fusion tables, *non-repeating grouping information is not preserved*.

OAuth2 Service Account Required

Publishing data to Google Spreadsheets requires an oauth2-service.

JSON Server

JSON preserves grouping and repeat structures. The user can choose to let multimedia files be represented as web links or embedded as base64 encoded strings. All metadata is preserved.

Google Maps Engine

Maps Engine is a cloud-hosted service to easily create and share maps. Multimedia files are represented by including standard web links to the files. Grouping information is preserved in the variable names. Repeating groups are forbidden and submissions must have at least one geopoint. Maps Engine also has an API that can be used to export/publish data.

RedCap Server

RedCap servers ignore grouping and repeating information. Multimedia files are sent as binary form elements in POST.

Ohmage JSON Server

Ohmage JSON preserves grouping and repeat structures. Multimedia files are sent as binary form elements in POST.

For more details on JSON Server, RedCap Server and Ohmage JSON Server see Aggregate Publishers Implementation Details

Table 8.2: **Publish data Summary**

Service	Groups	Repeats	Multimedia	Metadata
Google Fusion Tables	not preserved	split into separate tables	links to	preserved
Google Spreadsheets	not preserved	links to	links to	preserved
JSON Server	preserved	preserved	links to or embedded as base64	preserved
Google Maps Engine	preserved	forbidden	links to	not preserved
RedCap Server	not preserved	not preserved	binary elements in POST	
Ohmage JSON Server	preserved	preserved	binary elements in POST	

8.4.3 APIs

ODK Aggregate has public APIs defined for sending data to and from its various components. This section gives references to those API implementations that can be used for connecting new applications to the ODK ecosystem.

Briefcase CLI

ODK Briefcase 1.4.4 introduces a new Command Line Interface (CLI) to automate downloading forms from ODK Aggregate (or ODK Collect) and exporting the forms to an Excel-compatible format like CSV. For more details, see *Working with command line on Briefcase*.

Briefcase-Aggregate API

ODK Briefcase uses an API that external applications can use to pull all data from and push data to ODK Aggregate.

Interfaces used during pull actions are:

- OpenRosa Form Discovery API
- download forms and media files using URLs form discovery
- `view/submissionList` to obtain a chunk of submission keys
- `view/downloadSubmission` to download an individual submission

Interfaces used during push actions are:

- `formUpload` to upload a form and its media files to ODK Aggregate
- `view/submissionList` to obtain a chunk of submission keys
- OpenRosa Form Submission/Overwrite API (with extensions)

For more details, see Briefcase Aggregate API.

OpenRosa API

Collect and Aggregate communicate using a standard set of API calls defined in the *OpenRosa* specification, which can also be used by alternative clients.

Direct database connection

If you are running on a Tomcat server, you have access to the underlying MySQL or PostgreSQL tables. The structure of those tables are documented at Aggregate Database Structure.

> **Warning:** This is a fragile way to pull data from Aggregate because table structure could change between versions. Moreover, changing any of this data could corrupt your Aggregate install.

8.4.4 Accessing Media

Many of the export and publishing options provide a URL to the media (image, audio or video) without providing the content itself. To enable the viewing (following) of this link without requiring a log-in:

- Go to the *Permissions* sub-tab under the *Site Admin* tab.
- Check the checkbox for: *Allow anonymous retrieval of images, audio and video data (needed for GoogleEarth balloon displays).*
- Click the *Save Changes* button.

This allows anyone to view the media files on your server. Even though you are granting anyone access to this information, it is still quite secure because the users would need to have a valid URL.

8.5 Visualizing Geographic Data

This guide helps the users to visualize the collected geodata uploaded on ODK Aggregate server. There are various options you can choose from, like Google Fusion Tables and Google Earth.

Tip:

- If you're looking to do some quick visualizations of a small dataset and aren't too worried about the inherent risks of experimental tools, then you should use *Google Fusion Tables*.
- If you're serious about the long-term use of geodata and have large dataset use *Google Earth*.

Prerequisites

1. Make sure you have set up your *ODK Aggregate server*.
2. You should know how *ODK Collect* works.
3. You should be familiar with form designers like *ODK Build*, *XLSForm* to create your location based forms and upload it to ODK Aggregate server.

8.5.1 Google Fusion Tables

Google Fusion Tables is an experimental data visualization web application provided by Google for data management. Fusion tables can be used for visualizing dataset on a map.

Follow these steps to view your data using fusion tables:

1. Login to your ODK Aggregate server.
2. In the *Submissions* tab, under *Filter Submissions*, select the form you want to use.

3. If you are using Google Cloud Platform make sure you have enabled relevant APIs like **Google Drive API**, **Fusion Tables API**, **and Google Maps Javascript API** otherwise you may get *RequestFailureException error*.

4. To enable the APIs go to Google Cloud Platform and click on *Console* in the top right corner. Click on the three horizontal bars in the top left corner, now select *APIs & services* from the menu and then select *Library* option.

5. You will be able to see the API Library page.

6. Search the APIs in the search bar.

7. Click on *Enable*.

8. Now return to ODK Aggregate server and click on the *Publish* button.

9. In *Publish to* field, choose the platform from the drop-down menu to which you want to publish your data, choose Google Fusion Tables. In the *Data to Publish* field, Select the option from the drop-down menu to define what you want to upload:

- *Upload Existing Submission Data Only* will only upload the existing data you already have submitted to your ODK Aggregate server.
- *Stream New Submission Data Only* can be used if you want new submissions to be automatically entered into Google Fusion Tables.
- *BOTH Upload Existing and Stream New Submission Data* will upload your existing submissions into Fusion Tables and continue automatically adding new submissions to your table.

Choose *BOTH Upload Existing and Stream New Submission Data* and click on *Publish*.

Tip: If you choose *BOTH Upload Existing and Stream New Submission Data* option, you would be able to create a link between your data in Google Earth and Google Fusion Tables, for more information, refer this link.

---

10. You will need to enter an email address and that email address will be granted access to the documents.

11. Check the inbox of the email address that you entered and there would be an email from **odk-oauth2-publishing**, click on *Open*.

12. After clicking on *Open* you would be redirected to Google Fusion tables website. Here you would be able to see your submissions.

13. Select *Map of location* and you should see your data points displayed on a Google Map.

Tip: If the points aren't displaying where you expect them to display, visit Help . You can also customize pop-up balloons in Google Fusion Tables using *Change info window* button under the *Feature map* option.

8.5.2 Google Earth

Google Earth is a virtual globe that accesses satellite and aerial imagery, and other geographic data over the internet to represent the Earth as a three-dimensional globe

Follow these steps to view your data using Google Earth:

1. Login to your ODK Aggregate server.
2. In the *Submissions* tab, under *Filter Submissions*, select the form you want to use and click on *Export*

3. Choose the type as `KML file` and click on *Export*.

4. You don't need to change the default options, but you can select the *title* according to your choice. Click on *Export*
5. You will be redirected to *Export Submissions* page where you can download your kml file.

6. After downloading go to Google Earth website or you can download it from here. Wait for a while as Google Earth takes time to load.

7. Click on and to enable KML import click on *Settings*.

8. In the Settings window, enable KML file import and click on *SAVE*.

9. Click on *IMPORT KML FILE* and import the downloaded kml file.

10. Now you will be able to see your data in an organized manner on Google Earth.

Tip:

- On Google Earth, you can choose your map style, add additional points, lines and polygons to add more information for the enhancement of map. You can also try out this tutorial on how to annotate Google Earth.

8.6 Getting Help in Aggregate

Aggregate provides three kinds of help accessible by pressing one of three buttons in the upper righthand corner.

The red question mark will give you instructions for the tab you are currently viewing. When you click the button, a help panel will appear at the bottom of the screen. To hide the help panel, simply click the red question mark again.

The green book will give you the most comprehensive help. When you click the button, a popup will appear providing detailed information as well as video instructions.

The blue balloon increases the amount of detail that appears describing the button's functionality when you hover over most buttons.

CHAPTER

NINE

TIPS AND BEST PRACTICES

9.1 Form Uploader

Use ODK Form Uploader to easily upload a blank form and its media files to ODK Aggregate.

Note:

- Adding blank forms through the ODK Aggregate website is limited to an overall form and media size of 10MB; beyond that, you have to perform multiple uploads of the form definition file with different subsets of the media files in order to fully upload the blank form and its media attachments.
- If you have many media attachments or more than 10MB of media attachments, Form Uploader provides a quick and easy means to upload forms into ODK Aggregate.

9.1.1 Installation

1. Make sure latest version of Java is installed on the computer you plan to use. If it is not, download and install it.
2. Download the latest version of ODK Form Uploader
3. Double click the file to start. If that fails, try running `java -jar path_to_jar` from the command line.

Warning:

- On Windows use `java -jar "path_to_jar"`

9.1.2 Usage

1. Choose the form definition file (**.xml**) to upload.
2. For Installation and setup of ODK Aggregate, follow the steps mentioned *here*.
3. Specify the ODK Aggregate server. When specifying the ODK Aggregate server, either leave the username blank if the *anonymousUser* is granted Form Manager permissions, or specify an ODK Aggregate user (Account Type '**ODK**') that has been granted Form Manager permissions. ODK Form Uploader does not work with Google accounts (Account Type '**Google**').
4. Specify the URL and Password.
5. Click *Upload Form*.

Note:

- ODK Form Uploader will look for a matching media directory and upload the form definition file and all media files to the ODK Aggregate server.
- For example, when the form definition file is named `myForm.xml`, that form definition and all files in the matching media directory `myForm-media` will be uploaded. Note that the media directory should not contain any nested directories.

9.2 Aggregate Limitations

This is a listing of known limitations and potential "gotchas" users of Aggregate may encounter.

9.2.1 Pushing Data to Aggregate on Google App Engine

If Aggregate is *installed on Google App Engine*, using the default datastore as described in our documentation, a combination of request time limits and datastore implementation create the following issues.

Simultaneous push requests will block each other and may time out

Within ODK Aggregate, there is a global mutex (*TaskLock* across all server instances, mediated by the datastore layer) in the server when inserting submissions. Having multiple push requests occurring simultaneously will cause them to block on the mutex, chewing up their 60-second request limit, as they get processed in single file no matter how many server instances are spun up.

The solution to this is: **Serialize your push requests.**

Time limit may be exceeded on low-bandwidth connections

The 60-second request limit can be very commonly exceeded over low-bandwidth connections, and even text-only submissions can be impacted on satellite connections. That is why ODK Collect splits submissions into multiple 10MB submission requests. The timer starts upon receipt of the first byte, so a slow connection can account for a sizeable portion of those 60 seconds. The same applies for sending a response. The processing on the server is generally negligible in relation to the transmission times.

Note:

- The above two limitations, the global mutex and the in-memory copies/full-packet-assembly, are a result of implementing on top of AppEngine and its Datastore.
- A server that used database transactions and that used streaming servlet 3.0 functionality would have less trouble with concurrent requests.

9.2.2 Media held in memory

When a form submission is uploaded, and when blank forms are downloaded, all the associated media files are held in memory at the same time, twice. For forms with a lot of media files, this can consume a lot of memory.

The previous section already suggested serializing form submission uploads. This is not absolutely critical for form downloads, but you should probably manage how many form download requests are being handled concurrently, in order to avoid memory problems.

9.2.3 Uploading blank forms with media exceeding 10MB

Adding blank forms through the ODK Aggregate website is limited to an overall form+media size of 10MB. Beyond that, you have to perform multiple uploads of the form definition file with different subsets of the media files in order to fully upload the blank form and its media attachments.

An easier solution is use:doc:*Form Uploader <form-uploader>*.

9.2.4 Issues with older version of Aggregate

Aggregate < 1.4.8 used deprecated technology

Aggregate 1.4.7 and earlier use a deprecated backend technology. Google may terminate support for that at any time with little warning. You should consider upgrading.

Aggregate 1.4.15 changed sync protocol

Aggregate 1.4.15 fixed the ODK 2.0 rev 210 sync protocol. Prior to this, user permissions were incorrectly being computed and filtered. This prevented resetting the server with new content from the device (but syncing with existing content worked fine). The rev 210 sync protocol is incompatible with anything prior to rev 210.

Basic Auth broken prior to Aggregate 1.4.14

OpenRosa servers are *required to implement at least one authentication protocol* as outlined in RFC2617.

- a subset of RFC2617 Digest Authentication
- Basic Authentication.

Aggregate v1.4.14 added an SHA-1 library so that Basic Auth is possible. Prior to this, Basic Auth was not possible.

Note: Basic Auth is not exposed in the setup wizard. Additionally, it requires that default passwords be changed.

9.3 Reducing Data Corruption and Boosting Performance on App Engine

If you have many form definitions on your server, you may get better performance and reduce the likelihood of data corruption if you increase the size of your server.

9.3.1 What is causing data corruption?

Data corruption is generally caused by the premature termination of an action (for example, saving of a submission) because it took longer than the allotted time.

The likelihood of data corruption is correlated with the following factors:

- quantity of form definitions on the server
- the size of the individual submissions
- the number of devices simultaneously submitting data
- the speed of the network.

Increasing the web server size enables it to handle larger workloads faster, which can reduce the likelihood of hitting this time limit, thereby avoiding data corruption.

9.3.2 Increasing web server size

To change Google App Engine configuration, you must edit the configuration files produced by the installer and re-run the uploader script to push the changes to Google's servers.

There are two server settings that can be changed:

Web Server Size

The web server handles all browser interactions and all data-submission and form-download requests from ODK Collect and ODK Briefcase. Increasing the size of the web server should reduce the likelihood of data corruption if it is not caused by slow network speeds.

To change the Google App Engine web server size:

1. Go to the folder you specified when you ran the installer.
2. Within that folder, navigate to `ODKAggregate/default/WEB-INF/appengine-web.xml` and open that file.
3. The file looks like this:

```
<appengine-web-app xmlns="http://appengine.google.com/ns/1.0">
  <application>opendatakit-simpledemo</application>
  <module>default</module>
  <version>1</version>

      <instance-class>F2</instance-class>
...
```

   To change the size of the web server, replace **F2** with a different instance class size.

   There are several different instance classes available. Select from among the instance classes beginning with the letter **F**. See instance classes for their descriptions or search for *Google AppEngine instance classes standard environment* on the web.

4. Re-run the upload tool within the ODKAggregate folder.

- Windows: double-click `ODKAggregateAppEngineUpdater.jar`
- Mac: double-click `uploadAggregateToAppEngine.app`
- Linux: double-click `uploadAggregateToAppEngine.sh`

5. Once you have uploaded these changes to App Engine, your server will be running on the instance size that you have specified.

Background Server Size

App Engine deployments use a "background" copy of the website to process long-running actions like generating CSV and KML files for export and for publishing all accumulated data to an external server.

If you experience difficulty exporting to CSV or KML, the size of that server will also need to be updated.

1. Go to the folder you specified when you ran the installer.
2. Within that folder, navigate to `ODKAggregate/background/WEB-INF` and open that file.
3. The file looks like this:

```
<appengine-web-app xmlns="http://appengine.google.com/ns/1.0">
   <application>opendatakit-simpledemo</application>
   <module>background</module>
   <version>1</version>

       <instance-class>B2</instance-class>
...
```

   To change the size of the server, replace **B2** with a different instance class size.

   There are several different instance classes available. Select from among the instance classes beginning with the letter **B**. See instance classes for their descriptions or search for *Google AppEngine instance classes standard environment* on the web as described in Web Server Size.

   (a) Re-run the upload tool within the ODKAggregate folder.

   - Windows: double-click `ODKAggregateAppEngineUpdater.jar`
   - Mac: double-click `uploadAggregateToAppEngine.app`
   - Linux: double-click `uploadAggregateToAppEngine.sh`

5. Once you have uploaded these changes to App Engine, your server will be running on the instance size that you have specified.

Note: For data corruption caused by slow network speeds, you might also be able to change more aspects of the App Engine configuration (specified in these files) to make your web server always-available and to replace it with a Bx instance that does not have an automatic request time limit (the documentation provided by Google is currently unclear on whether this is still possible with the new services constructions).

CHAPTER

TEN

ODK BRIEFCASE

ODK Briefcase is an open source form management tool that works with *ODK Collect* and *ODK Aggregate*.

With Briefcase you can:

- Pull blank form definitions and completed form instances *from Collect or Aggregate* to local storage.
- *Push blank form definitions and completed form instances* from local storage to an Aggregate server.
- *Export data from completed form instances* to *CSV* files.
- *Transfer data across upgrade-incompatible versions of Aggregate*.
- *Decrypt encrypted submissions*.
- Back up form data from an Aggregate server.
- Work offline.
- *Work from a command line*.

10.1 Learn more about Briefcase

- *Setting Up ODK Briefcase*
- *Using ODK Briefcase*

CHAPTER

ELEVEN

SETTING UP ODK BRIEFCASE

Note: The ODK Briefcase 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 or higher is installed on your system.

1. Download ODK Briefcase.

   You may wish to move the Briefcase file to your desktop, your Applications directory, or another location.

2. Open the file.

   - Double click the file icon.
   - Or, from the command line:

   ```
   $ java -jar {path/to/Briefcase}
   ```

3. Set the location for Briefcase Storage.

   The first time you open Briefcase, you will need to select a directory for storing forms and submission data. A new directory called `ODK Briefcase Storage` will be created under the directory you select.

   Tip: You can change the storage location later, from the *Settings* tab.

CHAPTER

TWELVE

USING ODK BRIEFCASE

12.1 Pulling forms from Aggregate

To download blank forms and completed form instances from an *Aggregate* server:

1. Open the *Pull* tab.
2. Select *Aggregate* in the *Pull data from* drop-down.
3. Click the *Connect* button and enter the URL and login credentials for your Aggregate server.

   If you have anonymous login enabled on Aggregate, no login credentials are needed here.

   To connect to the Aggregate Demo Server, the URL is https://opendatakit.appspot.com.
4. Select the forms you want to download and click *Pull*. The selected forms will be pulled to `ODK Briefcase Storage` on your local system.

   For each selected form, Briefcase will pull down:

   - The form definition file (that is, the blank XForm).
   - All media associated with the form.
   - Completed form instances.

   If you have previously pulled the form:

   - New instances will be downloaded to the same location as previously downloaded ones.
   - The form definition file and media files will not be re-copied.

   > **Warning:** If your local copy and the remote copy of the blank form definition file are different, the pull will be aborted.

Workaround

If the form definition has changed, but the changes only affect the question text and do not alter the structure of the collected data (or change the form ID or version), you can:

(a) In *Settings*, temporarily change the location of `ODK Briefcase Storage`.

(b) Pull data into to the new location.

(c) Manually copy the instances from the temporary location to your original storage location.

(d) Update *Settings* back to the original storage location.

12.2 Pushing forms to Aggregate

To upload blank forms and completed form instances to *Aggregate*:

1. Open the *Push* tab.
2. Select *Aggregate* in the *Push data to* drop-down.
3. Click the *Connect* button and enter the URL and login credentials for your Aggregate server.

   If you have anonymous login enabled on Aggregate, no login credentials are needed here.

   To connect to the Aggregate Demo Server, the URL is https://opendatakit.appspot.com.
4. Select the forms you want to upload and click *Push*. The selected forms will be pushed from `ODK Briefcase Storage` on your local system to the Aggregate server.

   For each selected form, Briefcase will upload:

   - If not on the server already:
     - The form definition file (that is, the blank XForm).
     - All media associated with the form.
   - Completed form instances.

Warning: If your local copy and the remote copy of the blank form definition file are different, the push will be aborted.

Workaround

If the form definition has changed, but the changes only affect the question text and do not alter the structure of the collected data (or change the form ID or version), you can:

(a) In *Settings*, temporarily change the location of `ODK Briefcase Storage`.

(b) Manually copy the form directory from your original storage location to the temporary location.

(c) Replace the local form definition file with a copy of the version from your Aggregate server.

(d) Push your form instances.

(e) Update *Settings* back to the original storage location.

Tip: By pulling data into the local Briefcase Storage location and then pushing data up to an Aggregate instance, Briefcase provides a mechanism to transfer data across upgrade-incompatible versions of Aggregate.

12.3 Pulling forms from Collect

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 `/sdcard/odk` before deleting them, and restore it after you are finished.

2. Create a zip archive of the entire `odk` directory.

   Tip: You'll need to use an app for this.

   One option is OI File Manager.

3. Connect your Android device to your computer using a USB cable and choose to mount it as a Media device.

4. Copy the zip file you created from the Android device to your local hard drive.

5. Once it is copied onto your local hard drive, unzip the file.

6. In Briefcase, open the *Pull* tab.
7. Select *Custom Path to ODK Directory* in the *pull data from* drop-down.
8. Select the unzipped `odk` folder.
9. Click *Pull*.
10. On the Android device, open ODK Collect and delete the filled-in forms.

Tip:

- You can use the *Custom path to ODK Directory* any time you want to pull forms from custom location.
- You can confirm that the forms have been successfully pulled into Briefcase by confirming a successful pull status or by verifying the data appearing in a *CSV export file*.

Warning: Briefcase cannot discriminate between duplicate form instances. After you pull completed forms into Briefcase, it is important that you delete them from Collect. Otherwise, the next time you pull in forms, you will create duplicates.

Note: ODK Briefcase does not support pushing blank forms to ODK Collect. Instead, *manually load the forms on your ODK Collect device*.

12.4 Export forms to CSV

1. Open the *Export* tab.
2. Choose an Export Location.
3. If exporting *Encrypted Forms*, identify the location of your *PEM file*.
4. If you wish, select Start and End dates to specify a limited date range to export.
5. Select the forms to export.

   If you are selecting and exporting more than one form, you may need to individualize your export settings (export location, PEM file, start date, end date). To do this, click the gear icon ( ) next to the form name.
6. Click *Export*.

12.5 Working with the command line

New in version 1.4.4: A CLI was added.

12.5.1 Pulling form data from Aggregate

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

12.5.2 Pulling form data from Collect

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

```
$ java -jar {path/to/briefcase-jar-file} --form_id {form-id} --storage_directory
↪{path/to/briefcase-storage-location} --odk_directory {path/to/unzipped-odk-
↪file}
```

12.5.3 Exporting form data to CSV

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

12.5.4 Getting CLI help

To get help about the command line operation:

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

CHAPTER

THIRTEEN

BRIEFCASE AND AGGREGATE

ODK Briefcase and *ODK Aggregate* have complementary and overlapping features. For some deployments, it makes sense to use one or the other. In other case, it makes sense to use them together.

This page compares Briefcase and Aggregate, and explains their differences.

13.1 Briefcase

With Briefcase you can:

- Pull blank form definitions and completed form instances *from Collect or Aggregate* to local storage.
- *Push blank form definitions and completed form instances* from local storage to an Aggregate server.
- *Export data from completed form instances* to *CSV* files.
- *Transfer data across upgrade-incompatible versions of Aggregate*.
- *Decrypt encrypted submissions*.
- Back up form data from an Aggregate server.
- Work offline.
- *Work from a command line*.

Note: Briefcase cannot be used to Push forms directly to Collect. However, this is not needed as you can *do this with adb*.

13.2 Aggregate

With Aggregate you can:

- Host blank form definitions to be downloaded by Collect.
- Accept finalized form instances from Collect.
- Review and visualize submitted form data.
- Export submitted form data to *CSV*, *KML*, and *JSON*.
- Publish data to Google Fusion Tables or Google Sheets.
- Work offline.

13.3 Notable Differences

- Aggregate supports export to KML and JSON. Briefcase does not.
- Aggregate can host blank form definitions for Collect users to download over the internet. Briefcase cannot.
- Aggregate can receive form data from Collect over the internet. Briefcase cannot.
- When moving forms from Collect to Aggregate, form instances are ignored if they have not been finalized. When using Briefcase, all form instances are transferred.
- Aggregate supports simple visualizations. Briefcase does not.
- Briefcase has a CLI, while Aggregate does not. In addition to using Briefcase from a terminal, the CLI can also be scripted. No comparable scripting interface exists for Aggregate.
- Briefcase is a small and simple desktop application requiring very little setup and no maintenance. Aggregate is a full-fledged server backed by an enterprise database.

CHAPTER

FOURTEEN

INTRO TO FORMS IN ODK

Forms used in the ODK ecosystem are XML documents following the ODK XForms specification, a subset of the W3C XForms specification. (Example XForms are available for reference.)

Most ODK tools use the ODK Javarosa library to render forms. Form transfer between ODK components is governed by the *OpenRosa* API.

Because of the complexity of the XForms format, we do not recommend coding XForms directly in XML. The recommended process is:

1. Begin with one of the *JavaRosa-compliant build tools*.
2. Edit the XML only if necessary.
   - Before editing an XForm directly, you need to be familiar with the ODK XForm specification.
3. Use *ODK Form Validate* to check that the edited XForm is well-formed and fully compliant.

14.1 JavaRosa-compliant build tools

Most ODK users design their forms in Excel, following the XLSForm specification. To convert XLSForms to XForms, you can use:

- ODK's online XLSForm converter
- XLSForm Offline for Mac or Windows
- XLSForm for Windows
- *pyxform*, a Python XLSForm converter with a command-line tool

Tip: If you are comfortable with using the command-line, Pyxform is the most efficient XLSForm converter.

14.1.1 Non-XLSForm Javarosa builders

For simple forms, *ODK Build* is an online drag-and-drop form designer.

CHAPTER

FIFTEEN

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 Open Data Kit, try *XLSForm*.

You may also be interested in form design tools from other organizations:

- Vellum
- Kobo
- Enketo
- PurcForms.

15.1 Using ODK Build

Go to http://build.opendatakit.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.

15.1.1 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...*.

Note: Forms which are saved to your local machine have extension `.odkbuild` and only these forms can be loaded into ODK Build.

15.1.2 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.

15.1.3 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.

15.1.4 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.

15.1.5 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.

```
<instance>
  <data id="build_example1_1508999324">
    <meta>
      <instanceID/>
      <instanceName/>
    </meta>
    <s_name/>
    <sid/>
  </data>
</instance>
```

Fig. 15.1: Instance name as **sid-s_name: 12345-Ankita**

15.1.6 Add, remove and display new languages

You can add a new language for your form by clicking on *Manage Translations...* 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.

CHAPTER

SIXTEEN

XLSFORM

XLSForm (formerly XLS2Xform) is a tool to simplify the creation of forms. Forms designed with Excel can be converted to *XForms* that can be used with ODK tools.

Note:

- When this tool was renamed to XLSForm there were many changes to the syntax. The new version is mostly backwards compatible with the old syntax.
- XLSForm works with ODK Collect version *1.1.7* or later.

16.1 Using the Application

- To design your form, you can refer to the XLSForm form design documentation and check out the sample Excel file.
- Once your xls form is ready, you can submit it for conversion here.

16.2 Other XLSForm converters

- For Windows users, a standalone (non-web-based) version of this tool (without the error report) is also available on our downloads page here. If you use that tool, you should also download ODK Validate (from here) and run the generated form through that tool to generate an error report.
- Nafundi's XLSForm Offline is an app for Windows and Mac that can convert and validate forms. It is always available, very fast, and easy to use.
- *pyxform* is a Python library. It works offline and can be used on the command line to convert forms.

- XLSForm is also available through Ona.io which requires internet connection and also makes it easy to share converted forms and collected data.

Note:

- Enketo previews generated from this converter will not include media.
- Forms do not have to be uploaded to *Aggregate* before they are used. They can be manually added to *Collect*. Simply place them in the `/odk/forms` folder on your Android device's SD card.
- For a simpler, more user friendly form designer, try Build. For more powerful form designers, try Kobo or enketo or PurcForms. We also have sample forms (Examples forms are not available for all the features) and form design documentation. And in particular, design guidelines if you wish to design forms manually.

CHAPTER

SEVENTEEN

FORM WIDGETS

This document is a list of available ODK *Collect form widgets* (question types), with:

- example images from the ODK Collect app
- example Excel spreadsheet rows for creating form widgets with *XLSForm*
- example *ODK XForm* XML snippets

17.1 Basic Form Widgets

This section shows examples of all the form widgets types, with no additional options displayed.

17.1.1 String Input

A simple text input.

XLSForm Rows

Table 17.1: survey

type	name	label
text	name	What is your name?

XForm XML

```
<bind nodeset="/sample-xlsform/name" type="string"/>

<input ref="/sample-xlsform/name">
  <label>What is your name?</label>
</input>
```

17.1.2 Single Select

A radio-button single choice from several options.

XLSForm Rows

Table 17.2: survey

type	name	label
select_one fruit_list	fruits	What is your favorite fruit?

Table 17.3: choices

list_name	name	label
fruit_list	apple	apple
fruit_list	pear	pear
fruit_list	peach	peach
fruit_list	strawberry	strawberry
fruit_list	pineapple	pineapple
fruit_list	orange	orange
fruit_list	grape	grape
fruit_list	plum	plum

The **list_name** in the **choices** table matches the string *after* `select_one` in the **type** column of the **survey** table.

XForm XML

```
<bind nodeset="/sample-xlsform/fruits" type="select1"/>

<select1 ref="/sample-xlsform/fruits">
  <label>What is your favorite fruit?</label>
  <item>
    <label>apple</label>
    <value>apple</value>
  </item>
  <item>
    <label>pear</label>
    <value>pear</value>
  </item>
  <item>
    <label>peach</label>
    <value>peach</value>
  </item>
  <item>
    <label>strawberry</label>
    <value>strawberry</value>
  </item>
  <item>
    <label>pineapple</label>
    <value>pineapple</value>
  </item>
  <item>
    <label>orange</label>
    <value>orange</value>
  </item>
  <item>
    <label>grape</label>
    <value>grape</value>
  </item>
  <item>
    <label>plum</label>
    <value>plum</value>
  </item>
</select1>
```

17.1.3 Multi-select

A checkbox-list selection of multiple items. (The example includes a *hint*.)

XLSForm Rows

Table 17.4: survey

type	name	label	hint
select_multiple veggie_list	veggies	Which vegetables do you enjoy?	Select all that apply.

Table 17.5: choices

list_name	name	label
veggie_list	broccoli. broccoli	
veggie_list	carrot	carrot
veggie_list	spinach	spinach
veggie_list	tomato	tomato
veggie_list	bell_pepper	bell pepper
veggie_list	asparagus	asparagus

- The **list_name** in the **choices** table matches the string *after* `select_one` in the **type** column of the **survey** table.
- The content of **name** cannot have spaces.

XForm XML

```
<bind nodeset="/sample-xlsform/veggies" type="select"/>

<select ref="/sample-xlsform/veggies">
  <label>Which vegetables do you enjoy?</label>
  <hint>Select all that apply.</hint>
  <item>
    <label>broccoli</label>
    <value>broccoli</value>
  </item>
  <item>
    <label>carrot</label>
    <value>carrot</value>
  </item>
  <item>
    <label>spinach</label>
    <value>spinach</value>
  </item>
  <item>
    <label>tomato</label>
    <value>tomato</value>
  </item>
  <item>
    <label>bell pepper</label>
    <value>bell_pepper</value>
  </item>
  <item>
    <label>asparagus</label>
    <value>asparagus</value>
  </item>
</select>
```

17.1.4 Integer

A whole number entry input. Integer widgets will not accept decimal points.

XLSForm Rows

Table 17.6: survey

type	name	label
integer	age	What is your age in years?

XForm XML

```
<bind nodeset="/sample-xlsform/age" type="int"/>

<input ref="/sample-xlsform/age">
  <label>What is your age in years?</label>
</input>
```

17.1.5 Decimal

A numerical entry input that will accept decimal points.

XLSForm Rows

Table 17.7: survey

type	name	label
decimal	weight	Weight in kilograms.

XForm XML

```
<bind nodeset="/sample-xlsform/weight" type="decimal"/>

<input ref="/sample-xlsform/weight">
  <label>Weight in kilograms.</label>
</input>
```

17.1.6 Geopoint

A single set of GPS coordinates. The example includes a *hint*.

4:15
Sample XLSFo...
Current location.
You might have to turn on your GPS.
Start GeoPoint

4:18
Loading Location
Please wait. This could take a few minutes.
Cancel
Save GeoPoint

XLSForm Rows

Table 17.8: survey

type	name	label	hint
geopoint	current_location	Current location.	You might have to turn on your GPS.

XForm XML

```
<bind nodeset="/sample-xlsform/current_location" type="geopoint"/>

<input ref="/sample-xlsform/current_location">
  <label>Current location.</label>
  <hint>You might have to turn on your GPS.</hint>
</input>
```

17.1.7 GeoTrace

A line or polygon of GPS coordinates tracking actual device movement. The user can specify one of two location-recording modes:

- Manual Mode — The user taps the device to place a marker as desired while moving.
- Automatic Mode — The app creates a marker on a regular time interval (default: 20 seconds) as the user moves.

4:38
Wait for lock, then tap add marker button to start.
Zoom to...
Zoom to current location
Zoom to saved feature
Cancel
Southern Ocean
Google

4:38
Wait for lock, then tap add marker button to start.
Google

4:39
Wait for lock, then tap add marker button to start.
Center for the Arts &
Religion at the GTU
Select GeoTrace Mode
Manual Mode
Automatic Mode
Cancel
Start
Google

4:40
Wait for lock, then tap add marker button to start.
Center for the Arts &
Religion at the GTU
Select GeoTrace Mode
Manual Mode
Automatic Mode
20
Seconds
Cancel
Start
Observatory
Hill
Google

1:12
Wait for lock, then tap add marker button to start.
Google

4:41
Wait for lock, then tap add marker button to start.
Save GeoTrace As
Save as Polygon
Save as Polyline
Cancel
Google

XLSForm Rows

Table 17.9: survey

type	name	label
geotrace	trace_example	Where have you been?

XForm XML

```
<bind nodeset="/sample-xlsform/trace_example" type="geotrace"/>

<input ref="/sample-xlsform/trace_example">
  <label>Where have you been?</label>
</input>
```

17.1.8 GeoShape

A user-entered series of GPS coordinates, forming a polygon.

5:03
Sample XLSFo...
Select an area.
Start GeoShape

5:03
Long press to place marks.
Zoom to...
Zoom to current location
Zoom to saved feature
Cancel
Google

5:03
Long press to place marks.
Google

5:05
Long press to place marks.
Google

XLSForm Rows

Table 17.10: survey

type	name	label
geoshape	shape_example	Select an area.

XForm XML

```
<bind nodeset="/sample-xlsform/shape_example" type="geoshape"/>

<input ref="/sample-xlsform/shape_example">
  <label>Select an area.</label>
</input>
```

17.1.9 Date

A date selector.

8:24
Sample XLSFo...
What is your birthday?
Select date
No date selected

8:27
Sample XLSFo...
2017
Tue, Aug 8
August 2017
S M T W T F S
1 2 3 4 5
6 7 8 9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31
Cancel
OK

XLSForm Rows

Table 17.11: survey

type	name	label
date	birthday	What is your birthday?

XForm XML

```
<bind nodeset="/sample-xlsform/birthday" type="date"/>

<input ref="/sample-xlsform/birthday">
  <label>What is your birthday?</label>
</input>
```

17.1.10 Time

A time selector. Captures only a specific time-of-day, not a date and time. For date and time, see *Datetime*.

8:36
Sample XLSFo...
What time do you usually wakeup?
Select time
No time selected

8:38
Sample XLSFo...
What time do you usually wakeup?
Select time
7
37
8
:
38
AM
9
39
PM
Cancel
OK

XLSForm Rows

Table 17.12: survey

type	name	label
time	wakeup	What time do you usually wakeup?

XForm XML

```
<bind nodeset="/sample-xlsform/wakeup" type="time"/>

<input ref="/sample-xlsform/wakeup">
  <label>What time do you usually wakeup?</label>
</input>
```

17.1.11 Ethiopian Calendar

An Ethiopian-calendar. The main calendar used in Ethiopia. Users can enter a date according to the Ethiopian calendar system, and also see the corresponding Gregorian (standard) calendar date.

Sample F...
Ethiopian Calendar
Select date
No date selected

Sample F...
Ethiopian Calendar
Select date
21 Meskerem 2009
22 Tikimt 2010
23 Hidar 2011
22 Tikimt 2010 (Nov 01, 2017)
CANCEL
OK

XLSForm Rows

Table 17.13: survey

type	name	label
date	ethiopian-calendar	Ethiopian Calendar

XForm XML

```
<bind nodeset="/data/ethiopian-calendar" type="date" />

<input ref="/data/ethiopian-calendar" appearance="ethiopian">
  <label> Ethiopian Calendar </label>
</input>
```

17.1.12 Datetime

A datetime selector. For date only, see *Date*. For time only, see *Time*.

8:46
Sample XLSFo...
When was the last time you ate?
Select date
No date selected
Select time
No time selected

8:49
Sample XLSFo...
2017
Tue, Aug 8
August 2017
S M T W T F S
1 2 3 4 5
6 7 8 9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31
Cancel
OK

8:50
Sample XLSFo...
When was the last time you ate?
Select date
Aug 08, 2017
Select time
No time selected

8:51
Sample XLSFo...
When was the last time you ate?
Select time
7
50
8
51
AM
9
52
PM
Cancel
OK

XLSForm Rows

Table 17.14: survey

type	name	label
dateTime	previous_meal	When was the last time you ate?

XForm XML

```
<bind nodeset="/sample-xlsform/previous_meal" type="dateTime"/>

<input ref="/sample-xlsform/previous_meal">
  <label>When was the last time you ate?</label>
</input>
```

17.1.13 Image

An image collector. The user can choose to select an image stored on the device or take a new picture. The example includes *hint* text.

9:03
Sample XLSFo...
Please take a self portrait.
A "selfie."
Take Picture
Choose Image

XLSForm

Table 17.15: survey

type	name	label	hint
image	selfie	Please take a self portrait.	A "selfie."

XForm XML

```
<bind nodeset="/sample-xlsform/selfie" type="binary"/>

<upload mediatype="image/*" ref="/sample-xlsform/selfie">
  <label>Please take a self portrait.</label>
  <hint>A &quot;selfie.&quot;</hint>
</upload>
```

17.1.14 Audio

An audio recording collector.

XLSForm Rows

Table 17.16: survey

type	name	label
audio	name_pronounce	Please record your name.

XForm XML

```
<bind nodeset="/sample-xlsform/name_pronounce" type="binary"/>

<upload mediatype="audio/*" ref="/sample-xlsform/name_pronounce">
  <label>Please record your name.</label>
</upload>
```

17.1.15 Video

A video collector. The example includes *hint* text.

9:24
Sample XLSFo...
Please record a video of yourself blinking.
Three times is probably sufficient.
Record Video
Choose Video
Play Video

XSLForm Rows

Table 17.17: survey

type	name	label	hint
video	blinking	Please record a video of yourself blinking.	Three times is probably sufficient.

XForm XML

```
<bind nodeset="/sample-xlsform/blinking" type="binary"/>

<upload mediatype="video/*" ref="/sample-xlsform/blinking">
  <label>Please record a video of yourself blinking.</label>
  <hint>Three times is probably sufficient.</hint>
</upload>
```

17.1.16 Note

A note to the user, accepting no input. This example includes *hint* text.

XSLForm Rows

type	name	label	hint
note	note_1	This is an example note.	The text displays, but there is no input.

XForm XML

```
<bind nodeset="/sample-xlsform/note_1" readonly="true()" type="string"/>

<input ref="/sample-xlsform/note_1">
  <label>This is an example note.</label>
  <hint>The text displays, but there is no input.</hint>
</input>
```

17.1.17 Barcode

Captures the text from a barcode using the device camera. Supported barcode formats are described [here](https://github.com/zxing/zxing/#supported-formats). Non-printing control codes are removed.

The flash can be used as a light source when scanning barcodes in a poorly-lit environment.

Note: Barcode scanning is built into ODK Collect versions 1.7.0 and greater. Versions of ODK Collect prior to 1.7.0 require the [ZXing app](https://play.google.com/store/apps/details?id=com.google.zxing.client.android) to be installed.

12345678

1:01
Sample XLSFo...
Scan any barcode.
Replace Barcode
671860013525

XSLForm Rows

Table 17.18: survey

type	name	label
barcode	barcode_example	Scan any barcode.

XForm XML

```
<bind nodeset="/sample-xlsform/barcode_example" type="barcode"/>
<input ref="/sample-xlsform/barcode_example">
  <label>Scan any barcode.</label>
</input>
```

17.1.18 Acknowledge

An acknowledgment prompt with a single checkbox. In *Aggregate*, a completed acknowledgment is stored as the string response `OK`.

XLSForm Rows

Table 17.19: survey

type	name	label
acknowledge	ack_sample	I acknowledge this acknowledgment.

XForm XML

```
<bind nodeset="/sample-xlsform/ack_sample" type="string"/>

<trigger ref="/sample-xlsform/ack_sample">
  <label>I acknowledge this acknowledgment.</label>
</trigger>
```

17.2 Widget Formatting with Appearance

In your *xlsform* sheet, you can optionally specify an appearance column. This will affect widget display and functionality in *Collect*.

17.2.1 Text Widgets

Default Appearance

XLSForm Rows

Table 17.20: survey

type	name	label
text	string_widget	String widget

XForm XML

```
<bind nodeset="/all-widgets/text_widgets/string_widget" type="string"/>

<input ref="/all-widgets/text_widgets/string_widget">
   <label>String widget</label>
</input>
```

String Number

A numerical input that treats the input as a string, rather than a number.

XLSForm Rows

Table 17.21: survey

type	name	label	appearance	hint
text	string_number_widget	String number widget	numbers	text type with numbers appearance

XForm XML

```
<bind nodeset="/all-widgets/text_widgets/string_number_widget" type="string"/>

<input appearance="numbers" ref="/all-widgets/text_widgets/string_number_widget">
  <label>String number widget</label>
  <hint>text type with numbers appearance</hint>
</input>
```

URL Widget

Provides a link which the user can open from the survey. Takes no input.

The URL to open is specified with default.

XLSForm Rows

Table 17.22: survey

type	name	label	appearance	hint	default
text	url_widget	URL widget	url	text type with url appearance and default value of http://opendatakit.org/	http://opendatakit.org/

XForm XML

```
<instance>
  <all_widgets>
```

```
    <text_widgets>
      <!-- URL is defined here. -->
      <url_widget>http://opendatakit.org/</url_widget>
    </text_widgets>
  </all_widgets>
</instance>

<bind nodeset="/all-widgets/text_widgets/url_widget" type="string"/>

<input appearance="url" ref="/all-widgets/text_widgets/url_widget">
  <label>URL widget</label>
  <hint>text type with url appearance and default value of http://opendatakit.
↪org/</hint>
</input>
```

External App String Widget

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.

XLSForm Rows

Table 17.23: survey

type	name	label	appearance	hint
text	ex_string_widget	Ex string widget	ex:change.uw.android.BREATHCOUNT	text type with ex:change.uw.android.BREATHCOUNT appearance (can use other external apps)

XForm XML

```
<bind nodeset="/all-widgets/text_widgets/ex_string_widget" type="string"/>

<input appearance="ex:change.uw.android.BREATHCOUNT" ref="/all-widgets/text_
↪widgets/ex_string_widget">
  <label>Ex string widget</label>
    <hint>text type with ex:change.uw.android.BREATHCOUNT appearance (can use␣
↪other external apps)</hint>
</input>
```

External Printer Widget

Connects to an external printer. See printing widget for complete details.

XLSForm Rows

Table 17.24: survey

type	name	label	appearance	hint
text	ex_printer_widget	Ex printer widget	printer:org.opendatakit.sensors.ZebraPrinter	text type with printer:org.opendatakit.sensors.ZebraPrinter

XForm XML

```
<bind nodeset="/all-widgets/text_widgets/ex_printer_widget" type="string"/>

<input appearance="printer:org.opendatakit.sensors.ZebraPrinter" ref="/all-
↪widgets/text_widgets/ex_printer_widget">
  <label>Ex printer widget</label>
```

```
  <hint>text type with printer:org.opendatakit.sensors.ZebraPrinter</hint>
</input>
```

17.2.2 Number Widgets

Default Integer Widget

XLSForm Rows

Table 17.25: survey

type	name	label	hint
integer	integer_widget	Integer widget	integer type with no appearance

XForm XML

```
<bind nodeset="/all-widgets/number_widgets/integer_widget" type="int"/>
```

```
<input ref="/all-widgets/number_widgets/integer_widget">
  <label>Integer widget</label>
  <hint>integer type with no appearance</hint>
</input>
```

External Integer Widget

Launches an external app and receives an integer input back from the external app. If the specified external app is not available, a manual input is prompted.

XLSForm Rows

Table 17.26: survey

type	name	label	appearance	hint
integer	ex_integer_widget	Ex integer widget	ex:change.uw.android.BREATHCOUNT	integer type with ex:change.uw.android.BREATHCOUNT appearance (can use other external apps)

XForm XML

```
<bind nodeset="/all-widgets/number_widgets/ex_integer_widget" type="int"/>

<input appearance="ex:change.uw.android.BREATHCOUNT" ref="/all-widgets/number_
↪widgets/ex_integer_widget">
  <label>Ex integer widget</label>
  <hint>integer type with ex:change.uw.android.BREATHCOUNT appearance (can use␣
↪other external apps)</hint>
</input>
```

Default Decimal Widget

XLSForm Rows

Table 17.27: survey

type	name	label	hint
decimal	decimal_widget	Decimal widget	decimal type with no appearance

XForm XML

```
<bind nodeset="/all-widgets/number_widgets/decimal_widget" type="decimal"/>

<input ref="/all-widgets/number_widgets/decimal_widget">
  <label>Decimal widget</label>
  <hint>decimal type with no appearance</hint>
</input>
```

External Decimal Widget

Launches an external app and receives a decimal number input back from the external app. If the specified external app is not available, a manual input is prompted.

XLSForm Rows

Table 17.28: survey

type	name	label	appearance	hint
decimal	ex_decimal_widget	Ex decimal widget	ex:change.uw.android.BREATHCOUNT	decimal type with ex:change.uw.android.BREATHCOUNT appearance (can use other external apps)

XForm XML

```
<bind nodeset="/all-widgets/number_widgets/ex_decimal_widget" type="decimal"/>

<input appearance="ex:change.uw.android.BREATHCOUNT" ref="/all-widgets/number_
↪widgets/ex_decimal_widget">
  <label>Ex decimal widget</label>
  <hint>decimal type with ex:change.uw.android.BREATHCOUNT appearance (can use␣
↪other external apps)</hint>
</input>
```

Bearing Widget

Captures a compass reading.

12:58
Loading Bearing
Direction: W
Bearing: 273.001
Cancel
Record Bearing

1:01
All widgets
Numerical widgets
Bearing widget
decimal type with bearing appearance
Replace Bearing
271.538

XLSForm Rows

Table 17.29: survey

type	name	label	appearance	hint
decimal	bearing_widget	Bearing widget	bearing	decimal type with bearing appearance

XForm XML

```
<bind nodeset="/all-widgets/number_widgets/bearing_widget" type="decimal"/>

<input appearance="bearing" ref="/all-widgets/number_widgets/bearing_widget">
  <label>Bearing widget</label>
  <hint>decimal type with bearing appearance</hint>
</input>
```

17.2.3 Image Widgets

Default Image Widget

XLSForm Rows

Table 17.30: survey

type	name	label	hint
image	image_widget	Image widget	image type with no appearance

XForm XML

```
<bind nodeset="/all-widgets/image_widgets/image_widget" type="binary"/>

<upload mediatype="image/*" ref="/all-widgets/image_widgets/image_widget">
  <label>Image widget</label>
  <hint>image type with no appearance</hint>
</upload>
```

Selfie Widget

Takes a picture using the front-facing ("selfie") camera. The "Choose picture" button is not displayed.

1:21
All widgets
Image widgets
Selfie widget
image type with selfie appearance
Take Picture

XLSForm Rows

Table 17.31: survey

type	name	label	appearance	hint
image	selfie_image_widget	Selfie widget	selfie	image type with selfie appearance

XForm XML

```
<bind nodeset="/all-widgets/image_widgets/selfie_image_widget" type="binary"/>

<upload appearance="selfie" mediatype="image/*" ref="/all-widgets/image_widgets/
↪selfie_image_widget">
  <label>Selfie widget</label>
  <hint>image type with selfie appearance</hint>
</upload>
```

Draw Widget

Provides the user a drawing pad and collects the drawn image.

1:29
All widgets
Image widgets
Draw widget
image type with draw appearance
Sketch Image

Reset
Save and Close
Set Color

1:37
All widgets
Image widgets
Draw widget
image type with draw appearance
Sketch Image

XLSForm Rows

Table 17.32: survey

header type	name	label	appearance	hint
image	draw_image_widget	Draw widget	draw	image type with draw appearance

XForm XML

```
<bind nodeset="/all-widgets/image_widgets/draw_image_widget" type="binary"/>

<upload appearance="draw" mediatype="image/*" ref="/all-widgets/image_widgets/
↪draw_image_widget">
  <label>Draw widget</label>
  <hint>image type with draw appearance</hint>
</upload>
```

Annotate Widget

Allows the user to take or select an image and then draw on it.

1:47
All widgets
Image widgets
Annotate widget
image type with annotate appearance
Take Picture
Choose Image
Markup Image

1:51
All widgets
Image widgets
Annotate widget
image type with annotate appearance
Take Picture
Choose Image
Markup Image

Reset
Save and Close
Set Color

1:56
All widgets
Image widgets
Annotate widget
image type with annotate appearance
Take Picture
Choose Image
Markup Image

XLSForm Rows

Table 17.33: survey

type	name	label	appearance	hint
image	annotate_image_widget	Annotate widget	annotate	image type with annotate appearance

XForm XML

```
<bind nodeset="/all-widgets/image_widgets/annotate_image_widget" type="binary"/>

<upload appearance="annotate" mediatype="image/*" ref="/all-widgets/image_
↪widgets/annotate_image_widget">
  <label>Annotate widget</label>
  <hint>image type with annotate appearance</hint>
</upload>
```

Signature Widget

Collects a signature from the user.

2:01
All widgets
Image widgets
Signature widget
image type with signature appearance
Gather Signature

XLSForm Rows

Table 17.34: table

type	name	label	appearance	hint
image	signature_widget	Signature widget	signature	image type with signature appearance

XForm XML

```
<bind nodeset="/all-widgets/image_widgets/signature_widget" type="binary"/>

<upload appearance="signature" mediatype="image/*" ref="/all-widgets/image_
↪widgets/signature_widget">
  <label>Signature widget</label>
  <hint>image type with signature appearance</hint>
</upload>
```

17.2.4 Media Widgets

The media widgets do not accept any appearance attributes. They are documented in *Basic Form Widgets* above:

- *Barcode*
- *Audio*
- *Video*

17.2.5 Date and Time Widgets

Default Date Widget

3:05
All widgets
2017
Fri, Aug 11
August 2017
Cancel
OK

3:06
All widgets
Date and time widgets
Date widget
date type with no appearance
Select date
Aug 11, 2017

XLSForm Rows

Table 17.35: survey

type	name	label	hint
date	date_widget	Date widget	date type with no appearance

XForm XML

```
<bind nodeset="/all-widgets/date_time_widgets/date_widget" type="date"/>

<input ref="/all-widgets/date_time_widgets/date_widget">
  <label>Date widget</label>
  <hint>date type with no appearance</hint>
</input>
```

Date, No Calendar

The no-calendar appearance displays a "spinner" type date selection. This is especially appropriate for selecting dates more than in the past.

XSLForm Rows

Table 17.36: survey

type	name	label	appearance	hint
date	date_widget_nocalendar	Date Widget	no-calendar	date type with no-calendar appearance

XForm XML

```
<bind nodeset="/all-widgets/date_time_widgets/date_widget_nocalendar" type="date
↪"/>

<input appearance="no-calendar" ref="/all-widgets/date_time_widgets/date_widget_
↪nocalendar">
   <label>Date Widget</label>
   <hint>date type with no-calendar appearanec</hint>
</input>
```

Month and Year Widget

Collects only a month and year.

XLSForm Rows

Table 17.37: survey

type	name	label	appearance	hint
date	date_widget_month_year	Date widget	month-year	date type with month-year appearance

XForm XML

```
<bind nodeset="/all-widgets/date_time_widgets/date_widget_month_year" type="date
↪"/>

<input appearance="month-year" ref="/all-widgets/date_time_widgets/date_widget_
↪month_year">
  <label>Date widget</label>
```

```
  <hint>date type with month-year appearance</hint>
</input>
```

Year Widget

Collects only a year.

XLSForm Rows

Table 17.38: survey

type	name	label	appearance	hint
date	date_widget_year	Date widget	year	date type with year appearance

XForm XML

```
<bind nodeset="/all-widgets/date_time_widgets/date_widget_year" type="date"/>

<input appearance="year" ref="/all-widgets/date_time_widgets/date_widget_year">
```

```
  <label>Date widget</label>
  <hint>date type with year appearance</hint>
</input>
```

Time Widgets

The time widget does not accept an appearance attribute.

See *Basic Form Widgets* for details on:

- *Time*
- *Datetime*

The *Datetime* widget accepts a no-calendar appearance. This changes the date selector to the "spinner" style shown in *Date, No Calendar*.

17.2.6 GPS Widgets

Default Geopoint

The default *Geopoint* widget collects the current GPS position (if available) from the device.

XLSForm Rows

Table 17.39: survey

type	name	label	hint
geopoint	geopoint_widget	Geopoint widget	geopoint type with no appearance

XForm XML

```
<bind nodeset="/all-widgets/geopoint_widgets/geopoint_widget_placementmap" type=
↪"geopoint"/>

<input ref="/all-widgets/geopoint_widgets/geopoint_widget">
  <label>Geopoint widget</label>
  <hint>geopoint type with no appearance</hint>
</input>
```

Geopoint Placement Map Widget

A geopoint with the appearance attribute placement-map allows the user to select a geopoint from a map.

XLSForm Rows

Table 17.40: survey

type	name	label	appearance	hint
geopoint	geopoint_widget_placementmap	Geopoint widget	placement-map	geopoint type with placement-map appearance

XForm XML

```
<bind nodeset="/all-widgets/geopoint_widgets/geopoint_widget_placementmap" type=
↪"geopoint"/>

<input appearance="placement-map" ref="/all-widgets/geopoint_widgets/geopoint_
↪widget_placementmap">
  <label>Geopoint widget</label>
  <hint>geopoint type with placement-map appearance</hint>
</input>
```

Geopoint with Map

The default *Geopoint* widget does not display a map to the user. With the added maps appearance attribute, a map of the recorded location is shown to the user. The user cannot select a different location on the map. (See *Geopoint Placement Map Widget* for a geopoint with a user-selected location.)

XLSForm Rows

Table 17.41: survey

type	name	label	appearance	hint
geopoint	geopoint_widget_maps	Geopoint widget	maps	geopoint type with maps appearance

XForm XML

```
<bind nodeset="/all-widgets/geopoint_widgets/geopoint_widget_maps" type="geopoint
↪"/

<input appearance="maps" ref="/all-widgets/geopoint_widgets/geopoint_widget_maps
↪">
  <label>Geopoint widget</label>
  <hint>geopoint type with maps appearance</hint>
</input>
```

Geoshape and Geotrace

The *Geopoint* and *GeoTrace* widgets do not accept any appearance attributes.

17.2.7 Single Select Widgets

Default Single Select

XLSForm Rows

Table 17.42: survey

type	name	label	hint
select_one opt_abcd	select_one_widget	Select one widget	select_one type with no appearance, 4 text choices

Table 17.43: choices

list_name	name	label
opt_abcd	a	A
opt_abcd	b	B
opt_abcd	c	C
opt_abcd	d	D

XForm XML

```
<bind nodeset="/all-widgets/select_one_widgets/select_one_widget" type="select1"/
↪>

<select1 ref="/all-widgets/select_one_widgets/select_one_widget">
  <label>Select one widget</label>
  <hint>select_one type with no appearance, 4 text choices</hint>
  <item>
    <label>A</label>
    <value>a</value>
  </item>
  <item>
    <label>B</label>
    <value>b</value>
  </item>
  <item>
    <label>C</label>
    <value>c</value>
  </item>
  <item>
    <label>D</label>
    <value>d</value>
  </item>
</select1>
```

Spinner Widget

Adding the minimal appearance attribute places the choices into a drop-down style menu.

XLSForm Rows

Table 17.44: survey

type	name	label	appearance	hint
select_one opt_abcd	spinner_widget	Spinner widget	minimal	select_one type with minimal appearance, 4 text choices

Table 17.45: choices

list_name	name	label
opt_abcd	a	A
opt_abcd	b	B
opt_abcd	c	C
opt_abcd	d	D

XForm XML

```
<bind nodeset="/all-widgets/select_one_widgets/spinner_widget" type="select1"/>

<select1 appearance="minimal" ref="/all-widgets/select_one_widgets/spinner_widget
↪">
  <label>Spinner widget</label>
  <hint>select_one type with minimal appearance, 4 text choices</hint>
  <item>
    <label>A</label>
    <value>a</value>
  </item>
  <item>
    <label>B</label>
    <value>b</value>
  </item>
  <item>
    <label>C</label>
    <value>c</value>
  </item>
  <item>
    <label>D</label>
    <value>d</value>
  </item>
</select1>
```

Autoadvance Widget

Advances immediately to the next question once a selection is made.

Video showing auto-advance after the questions are answered.

XLSForm Rows

Table 17.46: survey

type	name	label	appearance	hint
select_one opt_abcd	select_one_autoadvance_widget	Select one autoadvance widget	quick	select_one type with quick appearance, 4 text choices

Table 17.47: choices

list_name	name	label
opt_abcd	a	A
opt_abcd	b	B
opt_abcd	c	C
opt_abcd	d	D

XForm XML

```
<bind nodeset="/all-widgets/select_one_widgets/select_one_autoadvance_widget"␣
↪type="select1"/>

<select1 appearance="quick" ref="/all-widgets/select_one_widgets/select_one_
↪autoadvance_widget">
  <label>Select one autoadvance widget</label>
  <hint>select_one type with quick appearance, 4 text choices</hint>
  <item>
    <label>A</label>
    <value>a</value>
  </item>
  <item>
    <label>B</label>
    <value>b</value>
  </item>
  <item>
    <label>C</label>
    <value>c</value>
  </item>
  <item>
    <label>D</label>
    <value>d</value>
  </item>
</select1>
```

Select with Search Widget

A set of radio buttons with a search and filter function.

1:35
All widgets
Select one widgets
Select one search widget
select_one type with search appearance, 4 text choices
A
B
C

2:45
All widgets
Select one widgets
Select one search widget
select_one type with search appearance, 4 text choices
A
B
C
I have
I
I will

XLSForm Rows

Table 17.48: survey

type	name	label	appearance	hint
select_one opt_abcd	select_one_search_widget	Select one search widget	search	select_one type with search appearance, 4 text choices

Table 17.49: choices

list_name	name	label
opt_abcd	a	A
opt_abcd	b	B
opt_abcd	c	C
opt_abcd	d	D

XForm XML

```
<bind nodeset="/all-widgets/select_one_widgets/select_one_search_widget" type=
↪"select1"/>

<select1 appearance="search" ref="/all-widgets/select_one_widgets/select_one_
↪search_widget">
  <label>Select one search widget</label>
  <hint>select_one type with search appearance, 4 text choices</hint>
  <item>
    <label>A</label>
    <value>a</value>
  </item>
  <item>
    <label>B</label>
    <value>b</value>
  </item>
  <item>
    <label>C</label>
    <value>c</value>
  </item>
  <item>
    <label>D</label>
    <value>d</value>
  </item>
</select1>
```

Select with Autocomplete Widget

A set of radio buttons with a search and filter function. The search input has an autocomplete feature.

XLSForm Rows

Table 17.50: survey

type	name	label	appearance	hint
select_one opt_abcd	select_one_autocomplete_widget	Select one search widget	autocomplete	select_one type with autocomplete appearance, 4 text choices

Table 17.51: choices

list_name	name	label
opt_abcd	a	A
opt_abcd	b	B
opt_abcd	c	C
opt_abcd	d	D

XForm XML

```
<bind nodeset="/all-widgets/select_one_widgets/select_one_autocomplete_widget"␣
↪type="select1"/>

<select1 appearance="autocomplete" ref="/all-widgets/select_one_widgets/select_
↪one_autocomplete_widget">
  <label>Select one search widget</label>
  <hint>select_one type with autocomplete appearance, 4 text choices</hint>
  <item>
    <label>A</label>
    <value>a</value>
  </item>
  <item>
    <label>B</label>
    <value>b</value>
  </item>
  <item>
    <label>C</label>
    <value>c</value>
  </item>
  <item>
    <label>D</label>
    <value>d</value>
  </item>
</select1>
```

Default Single Select with Images

A set of radio buttons with text labels and accompanying images.

See *Including Images as Choices* to learn more about including images in surveys.

XLSForm Rows

Table 17.52: survey

type	name	label	appearance	hint
select_one abcd_icon	grid_widget	Grid select one widget		select_one type with no appearance, 4 image choices (a.jpg, b.jpg, c.jpg, d.jpg)

Table 17.53: choices

list_name	name	label	media::image
abcd_icon	a	A	a.jpg
abcd_icon	b	B	b.jpg
abcd_icon	c	C	c.jpg
abcd_icon	d	D	d.jpg

XForm XML

```
<bind nodeset="/all-widgets/select_one_widgets/grid_widget" type="select1"/>

<select1 ref="/all-widgets/select_one_widgets/grid_widget">
  <label>Grid select one widget</label>
  <hint>select_one type with no appearance, 4 image choices (a.jpg, b.jpg, c.jpg,
↪ d.jpg)</hint>
  <item>
    <label ref="jr:itext('/all-widgets/select_one_widgets/grid_widget/a:label')"/
↪>
    <value>a</value>
  </item>
  <item>
    <label ref="jr:itext('/all-widgets/select_one_widgets/grid_widget/b:label')"/
↪>
    <value>b</value>
  </item>
  <item>
    <label ref="jr:itext('/all-widgets/select_one_widgets/grid_widget/c:label')"/
↪>
    <value>c</value>
  </item>
  <item>
    <label ref="jr:itext('/all-widgets/select_one_widgets/grid_widget/d:label')"/
↪>
    <value>d</value>
  </item>
</select1>
```

Compact Single Select with Images

Image options are placed on a single line.

XLSForm Rows

Table 17.54: survey

type	name	label	appearance	hint
select_one abcd_icon	grid_widget_compact	Grid select one widget	compact	select_one type with compact appearance, 4 image choices (a.jpg, b.jpg, c.jpg, d.jpg)

Table 17.55: choices

list_name	name	label	media::image
abcd_icon	a	A	a.jpg
abcd_icon	b	B	b.jpg
abcd_icon	c	C	c.jpg
abcd_icon	d	D	d.jpg

XForm XML

```
<bind nodeset="/all-widgets/select_one_widgets/grid_widget_compact" type="select1
↪"/>

<select1 appearance="compact" ref="/all-widgets/select_one_widgets/grid_widget_
↪compact">
  <label>Grid select one widget</label>
  <hint>select_one type with compact appearance, 4 image choices (a.jpg, b.jpg,␣
↪c.jpg, d.jpg)</hint>
  <item>
    <label ref="jr:itext('/all-widgets/select_one_widgets/grid_widget_compact/
↪a:label')"/>
    <value>a</value>
  </item>
  <item>
    <label ref="jr:itext('/all-widgets/select_one_widgets/grid_widget_compact/
↪b:label')"/>
    <value>b</value>
  </item>
  <item>
    <label ref="jr:itext('/all-widgets/select_one_widgets/grid_widget_compact/
↪c:label')"/>
    <value>c</value>
  </item>
  <item>
    <label ref="jr:itext('/all-widgets/select_one_widgets/grid_widget_compact/
↪d:label')"/>
    <value>d</value>
  </item>
</select1>
```

Compact Single Select with Images, width specified

With the compact-{x} style, you can specify the number of images to display on each row. To display two images on each row, specify an appearance of compact-2.

XLSForm Rows

Table 17.56: survey

type	name	label	appearance	hint
select_one abcd_icon	grid_widget_compact2	Grid select one widget	compact-2	select_one type with compact-2 appearance, 4 image choices (a.jpg, b.jpg, c.jpg, d.jpg)

Table 17.57: choices

list_name	name	label	media::image
abcd_icon	a	A	a.jpg
abcd_icon	b	B	b.jpg
abcd_icon	c	C	c.jpg
abcd_icon	d	D	d.jpg

XForm XML

```
<bind nodeset="/all-widgets/select_one_widgets/grid_widget_compact2" type=
↪"select1"/>

<select1 appearance="compact-2" ref="/all-widgets/select_one_widgets/grid_widget_
↪compact2">
  <label>Grid select one widget</label>
  <hint>select_one type with compact-2 appearance, 4 image choices (a.jpg, b.jpg,
↪ c.jpg, d.jpg)</hint>
  <item>
    <label ref="jr:itext('/all-widgets/select_one_widgets/grid_widget_compact2/
↪a:label')"/>
    <value>a</value>
  </item>
  <item>
    <label ref="jr:itext('/all-widgets/select_one_widgets/grid_widget_compact2/
↪b:label')"/>
    <value>b</value>
  </item>
  <item>
    <label ref="jr:itext('/all-widgets/select_one_widgets/grid_widget_compact2/
↪c:label')"/>
    <value>c</value>
  </item>
  <item>
    <label ref="jr:itext('/all-widgets/select_one_widgets/grid_widget_compact2/
↪d:label')"/>
    <value>d</value>
  </item>
</select1>
```

Compact Single Select with Images and Autoadvance

The quickcompact appearance attribute combines the design of the *Compact Single Select with Images* widget with the *Autoadvance Widget* functionality.

Video showing Compact single select widget and auto-advance after the question is answered.

XLSForm Rows

Table 17.58: survey

type	name	label	appearance	hint
select_one abcd_icon	grid_widget_quickcompact	Grid select one widget	quickcompact	select_one type with quickcompact appearance, 4 image choices (a.jpg, b.jpg, c.jpg, d.jpg)

Table 17.59: choices

list_name	name	label	media::image
abcd_icon	a	A	a.jpg
abcd_icon	b	B	b.jpg
abcd_icon	c	C	c.jpg
abcd_icon	d	D	d.jpg

XForm XML

```
<bind nodeset="/all-widgets/select_one_widgets/grid_widget_quickcompact2" type=
↪"select1"/>

<select1 appearance="quickcompact" ref="/all-widgets/select_one_widgets/grid_
↪widget_quickcompact">
  <label>Grid select one widget</label>
  <hint>select_one type with quickcompact appearance, 4 image choices (a.jpg, b.
↪jpg, c.jpg, d.jpg)</hint>
  <item>
    <label ref="jr:itext('/all-widgets/select_one_widgets/grid_widget_
↪quickcompact/a:label')"/>
    <value>a</value>
  </item>
  <item>
    <label ref="jr:itext('/all-widgets/select_one_widgets/grid_widget_
↪quickcompact/b:label')"/>
    <value>b</value>
  </item>
  <item>
    <label ref="jr:itext('/all-widgets/select_one_widgets/grid_widget_
↪quickcompact/c:label')"/>
    <value>c</value>
  </item>
```

```
  <item>
    <label ref="jr:itext('/all-widgets/select_one_widgets/grid_widget_
↪quickcompact/d:label')"/>
    <value>d</value>
  </item>
</select1>
```

Compact Single Select with Images and Autoadvance, width specified

As with *compact*, you can specify a width when using quickcompact. To display two images on each row, set the appearance attribute to quickcompact-2.

Video showing Compact-2 widget and auto-advance after the question is answered.

XLSForm Rows

Table 17.60: survey

type	name	label	appearance	hint
select_one abcd_icon	grid_widget_quickcompact2	Grid select one widget	quickcompact-2	select_one type with quickcompact-2 appearance, 4 image choices (a.jpg, b.jpg, c.jpg, d.jpg)

Table 17.61: choices

list_name	name	label	media::image
abcd_icon	a	A	a.jpg
abcd_icon	b	B	b.jpg
abcd_icon	c	C	c.jpg
abcd_icon	d	D	d.jpg

XForm XML

```
<bind nodeset="/all-widgets/select_one_widgets/grid_widget_quickcompact2" type=
↪"select1"/>

<select1 appearance="quickcompact-2" ref="/all-widgets/select_one_widgets/grid_
↪widget_quickcompact2">
  <label>Grid select one widget</label>
```

```
  <hint>select_one type with quickcompact-2 appearance, 4 image choices (a.jpg,␣
↪b.jpg, c.jpg, d.jpg)</hint>
  <item>
    <label ref="jr:itext('/all-widgets/select_one_widgets/grid_widget_
↪quickcompact2/a:label')"/>
    <value>a</value>
  </item>
  <item>
    <label ref="jr:itext('/all-widgets/select_one_widgets/grid_widget_
↪quickcompact2/b:label')"/>
    <value>b</value>
  </item>
  <item>
    <label ref="jr:itext('/all-widgets/select_one_widgets/grid_widget_
↪quickcompact2/c:label')"/>
    <value>c</value>
  </item>
  <item>
    <label ref="jr:itext('/all-widgets/select_one_widgets/grid_widget_
↪quickcompact2/d:label')"/>
    <value>d</value>
  </item>
</select1>
```

17.2.8 Multiselect Widgets

Default Multiselect Widget

XLSForm Rows

Table 17.62: survey

type	name	label	hint
select_multiple opt_abcd	select_multi_widget	Multi select widget	select_multiple type with no appearance, 4 text choices

Table 17.63: choices

list_name	name	label	media::image
abcd_icon	a	A	a.jpg
abcd_icon	b	B	b.jpg
abcd_icon	c	C	c.jpg
abcd_icon	d	D	d.jpg

XForm XML

```
<bind nodeset="/all-widgets/select_multi_widgets/select_multi_widget" type=
↪"select"/>

<select ref="/all-widgets/select_multi_widgets/select_multi_widget">
  <label>Multi select widget</label>
  <hint>select_multiple type with no appearance, 4 text choices</hint>
  <item>
    <label>A</label>
    <value>a</value>
  </item>
  <item>
    <label>B</label>
    <value>b</value>
  </item>
  <item>
    <label>C</label>
    <value>c</value>
  </item>
  <item>
    <label>D</label>
    <value>d</value>
  </item>
</select>
```

Compact Multiselect with Images

XLSForm Rows

Table 17.64: survey

type	name	label	appearance	hint
select_multiple abcd_icon	grid_multi_widget_compact	Grid select multiple widget	compact	select_multiple type with compact appearance, 4 image choices

Table 17.65: choices

list_name	name	label	media::image
abcd_icon	a	A	a.jpg
abcd_icon	b	B	b.jpg
abcd_icon	c	C	c.jpg
abcd_icon	d	D	d.jpg

XForm XML

```
<bind nodeset="/all-widgets/select_multi_widgets/grid_multi_widget_compact" type=
↪"select"/>

<select appearance="compact" ref="/all-widgets/select_multi_widgets/grid_multi_
↪widget_compact">
  <label>Grid select multiple widget</label>
  <hint>select_multiple type with compact appearance, 4 image choices</hint>
  <item>
    <label ref="jr:itext('/all-widgets/select_multi_widgets/grid_multi_widget_
↪compact/a:label')"/>
    <value>a</value>
  </item>
  <item>
    <label ref="jr:itext('/all-widgets/select_multi_widgets/grid_multi_widget_
↪compact/b:label')"/>
    <value>b</value>
  </item>
  <item>
    <label ref="jr:itext('/all-widgets/select_multi_widgets/grid_multi_widget_
↪compact/c:label')"/>
    <value>c</value>
  </item>
  <item>
```

```
    <label ref="jr:itext('/all-widgets/select_multi_widgets/grid_multi_widget_
↪compact/d:label')"/>
    <value>d</value>
  </item>
</select>
```

Compact Multiselect with Image Choices, style 2

XLSForm Rows

Table 17.66: survey

type	name	label	appearance	hint
select_multiple abcd_icon	grid_multi_widget	Grid select multiple widget	compact-2	select_multiple type with compact-2 appearance, 4 image choices

Table 17.67: choices

list_name	name	label	media::image
abcd_icon	a	A	a.jpg
abcd_icon	b	B	b.jpg
abcd_icon	c	C	c.jpg
abcd_icon	d	D	d.jpg

XForm XML

```
<bind nodeset="/all-widgets/select_multi_widgets/grid_multi_widget_compact2"␣
↪type="select"/>

<select appearance="compact-2" ref="/all-widgets/select_multi_widgets/grid_multi_
↪widget_compact2">
  <label>Grid select multiple widget</label>
  <hint>select_multiple type with compact-2 appearance, 4 image choices</hint>
  <item>
    <label ref="jr:itext('/all-widgets/select_multi_widgets/grid_multi_widget_
↪compact2/a:label')"/>
    <value>a</value>
  </item>
  <item>
    <label ref="jr:itext('/all-widgets/select_multi_widgets/grid_multi_widget_
↪compact2/b:label')"/>
    <value>b</value>
  </item>
  <item>
    <label ref="jr:itext('/all-widgets/select_multi_widgets/grid_multi_widget_
↪compact2/c:label')"/>
    <value>c</value>
  </item>
  <item>
    <label ref="jr:itext('/all-widgets/select_multi_widgets/grid_multi_widget_
↪compact2/d:label')"/>
    <value>d</value>
  </item>
</select>
```

Multiselect Spinner Widget

Adding the minimal appearance attribute places the choices into a drop-down style menu.

6:05
All widgets
This section contains "Select Multi Widgets"
Spinner widget: select multiple
select_multiple type with minimal appearance, 4 image choices
Select Answer
Selected:

6:07
All widgets
This section contains "Select Multi Widgets"
Spinner widget: select multiple
Spinner widget: select multiple
A
B
C
D
OK

XLSForm Rows

Table 17.68: survey

type	name	label	appearance	hint
select_multiple opt_abcd	spinner_multi_widget	Spinner widget: select multiple	minimal	select_multiple type with minimal appearance, 4 image choices

Table 17.69: choices

list_name	name	label
opt_abcd	a	A
opt_abcd	b	B
opt_abcd	c	C
opt_abcd	d	D

XForm XML

```
<bind nodeset="/all-widgets/select_multi_widgets/spinner_multi_widget" type=
↪"select"/>

<select appearance="minimal" ref="/all-widgets/select_multi_widgets/spinner_
↪multi_widget">
  <label>Spinner widget: select multiple</label>
  <hint>select_multiple type with minimal appearance, 4 image choices</hint>
  <item>
    <label>A</label>
    <value>a</value>
  </item>
  <item>
    <label>B</label>
    <value>b</value>
  </item>
  <item>
    <label>C</label>
    <value>c</value>
  </item>
  <item>
    <label>D</label>
    <value>d</value>
  </item>
```

```
</select>
```

17.2.9 Field List

The field-list appearance attribute, applied to a group of questions, displays them all on a single screen.

6:19
All widgets
List multi widget
Show only the inputs of these options and not the labels (type=selectmultiple yesno, appearance=list-nolabel)
List widget
This is a normal list widget with (type = select_one, appearance = list)
Yes
No
Don't Know
Not Applicable
List multi widget
This is a normal list widget with (type = select_multiple, appearance = list)
Yes
No
Don't Know
Not Applicable

XLSForm Rows

Table 17.70: survey

type	name	label	appearance	hint
begin_group	table_list_test	List group	field-list	
select_one yes_no	table_list_test_label	Label widget	label	Show only the labels of these options and not the inputs (type=select_one yes_no, appearance=label)
select_multiple yes_no	table_list_test_multi_label_2	Label multi widget	label	Show only the labels of these options and not the inputs (type=select_multiple yes_no, appearance=label)
select_one yes_no	table_list_1	List widget	list-nolabel	Show only the inputs of these options and not the labels (type=select_one yes_no, appearance=list-nolabel)
select_multiple yes_no	table_list_2	List multi widget	list-nolabel	Show only the inputs of these options and not the labels (type=select_multiple yes_no, appearance=list-nolabel)
select_one yes_no	list_widget	List widget	list	This is a normal list widget with (type = select_one, appearance = list)
select_multiple yes_no	list_multi_widget	List multi widget	list	This is a normal list widget with (type = select_multiple, appearance = list)
end_group				

Table 17.71: choices

list_name	name	label
yes_no	yes	Yes
yes_no	no	No
yes_no	dk	Don't Know
yes_no	na	Not Applicable

XForm XML

```
<instance>
  <all-widgets>
    <table_list_test>
      <table_list_test_label/>
```

```
      <table_list_test_label_2/>
      <table_list_1/>
      <table_list_2/>
      <list_widget/>
      <list_multi_widget/>
    </table_list_test>
  </all-widgets>
</instance>


<bind nodeset="/all-widgets/table_list_test/table_list_test_label" type="select1
↪"/>
<bind nodeset="/all-widgets/table_list_test/table_list_test_label_2" type="select
↪"/>
<bind nodeset="/all-widgets/table_list_test/table_list_1" type="select1"/>
<bind nodeset="/all-widgets/table_list_test/table_list_2" type="select"/>
<bind nodeset="/all-widgets/table_list_test/list_widget" type="select1"/>
<bind nodeset="/all-widgets/table_list_test/list_multi_widget" type="select"/>


<group appearance="field-list" ref="/all-widgets/table_list_test">
  <label>List group</label>
  <select1 appearance="label" ref="/all-widgets/table_list_test/table_list_test_
↪label">
    <label>Label widget</label>
    <hint>Show only the labels of these options and not the inputs (type=select_
↪one yes_no, appearance=label)</hint>
    <item>
      <label>Yes</label>
      <value>yes</value>
    </item>
    <item>
      <label>No</label>
      <value>no</value>
    </item>
    <item>
      <label>Don't Know</label>
      <value>dk</value>
    </item>
    <item>
      <label>Not Applicable</label>
      <value>na</value>
    </item>
  </select1>
  <select appearance="label" ref="/all-widgets/table_list_test/table_list_test_
↪label_2">
    <label>Label multi widget</label>
```

```
    <hint>Show only the labels of these options and not the inputs (type=select_
↪multiple yes_no, appearance=label)</hint>
    <item>
      <label>Yes</label>
      <value>yes</value>
    </item>
    <item>
      <label>No</label>
      <value>no</value>
    </item>
    <item>
      <label>Don't Know</label>
      <value>dk</value>
    </item>
    <item>
      <label>Not Applicable</label>
      <value>na</value>
    </item>
  </select>
  <select1 appearance="list-nolabel" ref="/all-widgets/table_list_test/table_
↪list_1">
    <label>List widget</label>
    <hint>Show only the inputs of these options and not the labels (type=select_
↪one yes_no, appearance=list-nolabel)</hint>
    <item>
      <label>Yes</label>
      <value>yes</value>
    </item>
    <item>
      <label>No</label>
      <value>no</value>
    </item>
    <item>
      <label>Don't Know</label>
      <value>dk</value>
    </item>
    <item>
      <label>Not Applicable</label>
      <value>na</value>
    </item>
  </select1>
  <select appearance="list-nolabel" ref="/all-widgets/table_list_test/table_list_
↪2">
    <label>List multi widget</label>
    <hint>Show only the inputs of these options and not the labels (type=select_
↪multiple yes_no, appearance=list-nolabel)</hint>
    <item>
```

```
      <label>Yes</label>
      <value>yes</value>
    </item>
    <item>
      <label>No</label>
      <value>no</value>
    </item>
    <item>
      <label>Don't Know</label>
      <value>dk</value>
    </item>
    <item>
      <label>Not Applicable</label>
      <value>na</value>
    </item>
  </select>
  <select1 appearance="list" ref="/all-widgets/table_list_test/list_widget">
    <label>List widget</label>
    <hint>This is a normal list widget with (type = select_one, appearance =␣
↪list)</hint>
    <item>
      <label>Yes</label>
      <value>yes</value>
    </item>
    <item>
      <label>No</label>
      <value>no</value>
    </item>
    <item>
      <label>Don't Know</label>
      <value>dk</value>
    </item>
    <item>
      <label>Not Applicable</label>
      <value>na</value>
    </item>
  </select1>
  <select appearance="list" ref="/all-widgets/table_list_test/list_multi_widget">
    <label>List multi widget</label>
    <hint>This is a normal list widget with (type = select_multiple, appearance␣
↪= list)</hint>
    <item>
      <label>Yes</label>
      <value>yes</value>
    </item>
    <item>
      <label>No</label>
```

```
      <value>no</value>
    </item>
    <item>
      <label>Don't Know</label>
      <value>dk</value>
    </item>
    <item>
      <label>Not Applicable</label>
      <value>na</value>
    </item>
  </select>
</group>
```

17.2.10 Trigger Widget

"Trigger" is another name for the *Acknowledge* widget. The example shown here includes the required attribute.

6:53
All widgets
Trigger widget
Prompts for confirmation. Useful to combine with required or relevant. (type=trigger)
OK. Please continue.
Sorry, this response is required!

6:55
All widgets
Trigger widget
Prompts for confirmation. Useful to combine with required or relevant. (type=trigger)
OK. Please continue.

XLSForm Rows

Table 17.72: survey

type	name	label	hint	required
trigger	my_trigger	Trigger widget	Prompts for confirmation. Useful to combine with required or relevant. (type=trigger)	true()

XForm XML

```
<bind nodeset="/all-widgets/my_trigger" required="true()"/>

<trigger ref="/all-widgets/my_trigger">
  <label>Trigger widget</label>
  <hint>Prompts for confirmation. Useful to combine with required or relevant.␣
↪(type=trigger)</hint>
</trigger>
```

17.3 Including Images as Choices

To include images as choices for select questions, specify the file name in the **choices** worksheet, in a column labeled media::image. The media files must be uploaded to the Android device in the `/sdcard/odk/forms/` directory, in a folder labeled *form-name*-media. When uploading a form to ODK Aggregate, a second upload prompt will allow you to upload your files directory.

CHAPTER

EIGHTEEN

INTERACTION OF CONSTRAINTS, RELEVANTS AND READ ONLY

18.1 Constraints

Adding constraints to the data fields in a form ensures data quality. For example, when asking for a person's age, you want to avoid impossible answers, like -22 or 200. Adding data constraints in your form is easy to do. You simply add a new column, called constraint, and type in the formula specifying the limits on the answer.

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

In the example below, the answer for the person's age must be less than or equal to 40. Note that . in the formula refers back to the question variable.

Table 18.1: **Survey**

type	name	label	constraint	constraint_message
integer	age	How old are you?	.<=40	Person must be 40 years old or younger to complete the survey.

For more details, refer constraints section in XLSForm documentation.

18.2 Relevants

Relevants are used to skip a question or make an additional question appear based on the response to a previous question. Relevants can be added to the question so that the question is asked only when the relevant is satisfied.

In the example below, school name will be asked only if age entered by the user is less than or equal to 18.

Table 18.2: **Survey**

type	name	label	relevant
integer	age	How old are you?	
text	School	Enter your school name	${age} <= 18

For more details, refer relevants section in XLSForm documentation.

18.3 Read only

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

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

To make questions as read only, add a **read_only** column to your survey worksheet. Under that column, mark questions as read only by writing yes.

In the example below, city is a read only value with default as California. User cannot modify this field.

Table 18.3: **Survey**

type	name	label	read_only	default
text	City	Enter a city where server will operate	yes	California

For more details, refer read only section in XLSForm documentation.

18.4 Required

Marking any question as required means the user will not be able to move on to the next question or submit the form without entering an answer for that question.

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

Table 18.4: **Survey**

type	name	label	required	
text	name	Enter your name	yes	

For more details, refer required section in XLSForm documentation.

18.5 Calculation

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

Table 18.5: **Survey**

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

For more details, refer calculation section in XLSForm documentation.

18.6 Tips on using above fields

- Constraints should only be applied to fields that can be changed. It is of no use to apply constraints to readonly fields or note fields.

Table 18.6: **Survey**

type	name	label	constraint	constraint_message	relevant	read_only	default	required	calculation	
text	name	Enter your name							yes	
integer	age	Enter your age	.<=18	You should be below 18 years to be eligible for the survey.					yes	
text	college	College name	${college}='IIT Roorkee'			yes	IIT Roorkee			
select_one yes_no	course	Have you chosen any courses?						yes		
integer	course_cnt	Enter number of courses you have chosen	1<=.<=6	You should choose at least 1 course and at most 6 courses.	${course}='yes'			yes		
integer	marks	Enter total marks obtained in all courses			${course}='yes'			yes		
calculate	total				${course_cnt}!="				${course_cnt}*100	
note	disp_max	Maximum marks possible are ${total}.	${total}<${marks}		${course_cnt}!="					

In this survey table, it is of no use to include a constraint in college field since it is already fixed and user cannot modify it. Further the constraint in the note field is of no use as it will just display an note that *Maximum allowed marks are total* but will allow the user to fill the form even with more than total marks.

To prevent the user from entering more than maximum marks, a constraint should be included in the marks field: **${marks}<=${total}** and the note should have a relevant: **${course_cnt}!=" and ${total}<${marks}**.

Table 18.7: **Survey**

type	name	label	constraint	constraint_message	relevant	read_only	default	required	calculation	
text	name	Enter your name							yes	
integer	age	Enter your age	.<=18	You should be below 18 years to be eligible for the survey.					yes	
text	college	College name				yes	IIT Roorkee			
select_one yes_no	course	Have you chosen any courses?						yes		
integer	course_cnt	Enter number of courses you have chosen	1<=.<=6	You should choose at least 1 course and at most 6 courses.	${course}='yes'			yes		
integer	marks	Enter total marks obtained in all courses			${course}='yes'			yes		
calculate	total				${course_cnt}!=''				${course_cnt}*100	
note	disp_max	Maximum marks possible are ${total}.	${total}<${marks}		${course_cnt}!=''					

These are the `.xml` files for the above forms:

1. `example_1`
2. `example_2`

These are the `.xlsx` files for the above forms:

1. `example_1`
2. `example_2`

- Whenever you perform a calculation make sure that it has proper relevants. Relevants should check that the variables required for calculation are not null and are supplied by the user. This will ensure that calculation fires only when no required variable is

null and correct value is generated. In the above forms calculate field has relevant set to **${course_cnt}!=”**, so that calculate fires only when some value is entered in number of courses.

- It is not necessary that a field which has a constraint is a required field. It is possible to represent a case when a value may not be known but if it is known, it must meet certain characteristics. For example, if a question asks for an exam percentage, it can be left blank but if provided with an answer it should be less than or equal to 100.

Table 18.8: **Survey**

type	name	label	constraint	constraint_message	required
integer	percentage	Enter your percentage	.<=100	You cannot score more than 100%	

CHAPTER

NINETEEN

USING REGULAR EXPRESSIONS

19.1 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://www.regextester.com/
- https://regexr.com/
- http://www.regexe.com/.

19.2 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 19.1: **Survey**

type	name	label	constraint	constraint_message
text	ex	Enter example	regex(.,'^[a-zA-Z]{0,6}$')	Input can have a maximum of 6 alphabetic characters.

  In **XForm 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**:

  Table 19.2: **Survey**

type	name	label	constraint	constraint_message	appearance
text	tel_no	Enter your Telephone number	regex(.,'^(([0-9]{1})*[-.(]*([0-9]{3})[-.)]*[0-9]{3}[- .]*[0-9]{4})+$')	Telephone numbers should have 10 digits with optional separators.	numbers

  In **XForm XML**:

```
<bind constraint="Regex(.,'^(([0-9]{1})*[- .(]*([0-9]{3})[- .)]*[0-9]
↪{3}[- .]*[0-9]{4})+$')" 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.

CHAPTER

TWENTY

UPDATING FORMS

20.1 Making changes to existing form

Certain changes which don't involve adding or removing a question can be made without needing to replace the existing forms. The kind of changes are:

- Changing the text or translation of a label
- Changing validations, calculations, relevants
- Changing options for a select_one or select_multiple
- Changing the order of questions
- Adding translations
- Updating media including CSVs for your form

For such changes you can update the version and re-upload the form.

Note:

- If you add new media files or update existing media files for your form without any change in the form definition file or `.xml` file, you don't need to change the version.
- If you have used external select and add, update or delete the choices in external `.csv` file without any change in the form definition file or `.xml` file, you don't need to change the version. For more details on using external choices in your form, see this

Tip: Form version in XLSForm is a string of up to 10 numbers that describes this revision. Revised form definitions must have alphabetically greater versions than previous ones. A common convention is to use strings of the form **yyyymmddrr**. For example, 2017120701 is the 1st revision from Dec 7th, 2017.

For example:

Table 20.1: **Survey**

type	name	label	constraint	constraint_message	relevant	read_only	default	required	calculation
text	name	Enter your name						yes	
integer	sid	Enter your student id						yes	
integer	age	Enter your age	.<=18	You should be below 18 years to be eligible for the survey.				yes	
select_one dept	course	In which department have you chosen courses?						yes	
integer	course_cnt	Enter number of courses you have chosen	1<=.<=6	You should choose at least 1 course and at most 6 courses.	${course}!='none'			yes	
integer	marks	Enter total marks obtained in all courses			${course}!='none'			yes	
calculate	total				${course_cnt}!=''				${course_cnt}*100

Table 20.2: **Choices**

list name	name	label
dept	Physics	PHY
dept	Maths	MAT
dept	Chemistry	CHEM
dept	none	none

Table 20.3: **Settings**

form_title	form_id	default_language	version
Example_form	example_id	English	2017120700

If you want to make the following changes to the above form:

- Add a Spanish translation

- Change the relative order of first and second question
- Change age constraint from **18** to **20**
- Change the sid field to not required
- Change label for third question from **In which department have you chosen courses?** to **Name of Department**.
- Change the calculation from **${course_cnt}*100** to **${course_cnt}*50**
- Change the relevant for calculate to **${course}!='none' and ${course_cnt}!=''**
- Change the list name in choices from **dept** to **dept_name**
- Add a new choice in department list as **Computer**
- Change **Maths** to **Mathematics** and **MAT** to **MATHS** in choices

These changes can be made as:

Table 20.4: **Survey**

type	name	label::English (en)	label::Español (es)	constraint	constraint_message	relevant	read_only	default	required	calculation
integer	sid	Enter your student id	Ingrese su identificación de estudiante						no	
text	name	Enter your name	Introduzca su nombre						yes	
integer	age	Enter your age	Introduzca su edad	.<=20	You should be below 20 years to be eligible for the survey.				yes	
select_one dept_name	course	Name of Department	Nombre del departamento						yes	
integer	course_cnt	Enter number of courses you have chosen	Ingresa el número de cursos que has elegido	1<=.<=6	You should choose at least 1 course and at most 6 courses.	${course}!='none'			yes	
integer	marks	Enter total marks obtained in all courses	Ingrese las calificaciones totales obtenidas en todos los cursos			${course}!='none'			yes	
calculate	total					${course}!='none' and ${course_cnt}!=''				${course_cnt}*50

Table 20.5: **Choices**

list name	name	label
dept_name	Physics	PHY
dept_name	Mathematics	MATHS
dept_name	Chemistry	CHEM
dept_name	Computer	COMP
dept_name	none	none

Table 20.6: **Settings**

form_title	form_id	default_language	version
Example_form	example_id	English	2017120701

Note: You need to use the same filename when generating newer versions of your form since the XLSForm converter uses that filename in creating the form structure which is used by the server to help determine uniqueness.

If you use the same filename, you can upload the form and ODK Aggregate will automatically replace the old version and keep the existing form data. If you use different filenames, you would need to delete the old version of the form (and the existing form data) in order to upload the newer version.

In the examples below, we use different filenames for version upgrade since unique filenames are required for uploading files to the documentation website.

Due to different XLSForm filenames in these examples, the generated XForms will have different instance definition as follows:

For `example_form_v1.0.xlsx`:

```
<instance>
    <example_form_v1.0 id="example_id" version="2017120700">
```

For `example_form_v1.1.xlsx`:

```
<instance>
    <example_form_v1.1 id="example_id" version="2017120701">
```

These are the `.xml` files for the above forms:

1. `example_form_v1.0`
2. `example_form_v1.1`

These are the `.xlsx` files for the above forms:

1. `example_form_v1.0`
2. `example_form_v1.1`

Note: You cannot change the question type or name, form id and title. Example: In above form you cannot change the type from **select_one** to **select_multiple** or change name from **dept** to **department**.

Note: ODK Collect treats different versions of the same form completely independently. It won't explicitly notify the users of the existence of new versions. When a user tries to get new blank forms, a form with updated version will be selected by default in the list but there will be no explicit notification unless a user tries to get new blank forms.

Both versions of the form exist in the device of the user and the user will be allowed to fill an older version and submit the form to the Aggregate server. You will need to manually delete an older version from your device.

20.2 Replacing existing form

If you need to make deeper changes like changing question type, name, form id, form title etc then you will need to create a new form with the required changes.

Once you will modify your form, the data you will collect will be stored under that new form. However, you do not need to delete the previous form, instead, you may change the name of the new form. For example, if you had form name as **Example_form**, the revised form can be named as **Example_form_1.1**. Additionally, on your Aggregate restrict the previous form by unchecking *Downloadable* and *Accept Submissions* options.

> **Warning:** If you make changes like changing a question type or name with the same form id and title and update the version, you won't be able to re-upload the form. Aggregate will reject the form upload with an error message.

Error: Form Already Exists for this Namespace/Id attribute
org.opendatakit.aggregate.exception.ODKFormAlreadyExist

CHAPTER

TWENTYONE

PYXFORM

pyxform is a Python library that makes writing XForms for ODK Collect and enketo easy by converting XLS(X) spreadsheets into XForms. It is used as a library in a number of tools including the ODK online converter and Ona.

21.1 Running pyxform from command line

For those who want to convert forms at the command line, pyxform can be installed directly from the command line using pip:

```
$ pip install pyxform
```

The **xls2xform** command can then be used:

```
$ xls2xform path_to_XLSForm output_path
```

Tip: For further queries or for more information please refer to the Github repo.

CHAPTER

TWENTYTWO

ODK FORM VALIDATE

Building a form is an essential step in ODK data collection project. You can use any tool you prefer for that, but you need to make sure that your form is compatible with ODK. ODK Validate ensures that you have an OpenRosa compliant XForm, which prepares your form for deployment on ODK tools.

22.1 Installation

- You need to have java.
- Download ODK Validate tool from (`ODK Validate vN.N.N.jar` is a Java ARchive),it doesn't require installation.
- To run the tool, just click on ODK Validate vN.N.N.jar, you will see the following image:

`vN.N.N` refers to the current version of ODK Validate. The download page describes new features added to the current version.

Warning: If you couldn't open the jar file, try to use the command line:

```
java -jar path_to_jar
```

Tip: In Windows, you can select and drop the jar file to get the full path for the command above.

22.2 Usage

- Build your ODK form using either ODK Build or XLSForm.
- Convert the form into XML format, and save it to your local machine.
- Open ODk Validate tool and from *choose file*, select your XML form.
- The form will be validated automatically at the first time, and you can check if any error occurs.

Tip:

- If you see errors, try to correct the source file, and try to validate the form again.
- When you validate a form, the tool parses the XML script and checks it against ODK XForms specifications.

The following images illustrate two examples of:

1. a valid form
2. an invalid form

Note: ODK has a few additional restrictions that may not be tested by this tool. In particular, a form id must be defined for the form to work in Aggregate.

CHAPTER

TWENTYTHREE

LAUNCHING EXTERNAL APPS FROM COLLECT FORMS

See also:

Launching ODK Collect from External Apps

23.1 Launching external apps to populate single fields

ODK Collect can launch external applications to populate string, integer or decimal fields using the `ex:intentString` appearance. A `value` parameter that holds the current value for that field is passed to the application. Since v1.4.3, additional parameters can be specified. The names of these parameters are user defined and `value` is the only reserved name.

23.1.1 XLSForm

Table 23.1: survey

type	name	label	appearance
integer	counter	Click launch to start the counter app	ex:org.opendatakit.counter(form_id='counter-form', form_name='Counter Form', question_id='1', question_name='Counter')

23.1.2 XForm XML

```
<input appearance="ex:org.opendatakit.counter(form_id='counter-form', form_name=
↪'Counter Form', question_id='1', question_name='Counter')" ref="/counter/
↪counter">
    <label>Click launch to start the counter app</label>
</input>
```

In the examples above, the parameters specified are `form_id`, `form_name`, `question_id` and `question_name`. Any number of extra parameters can be specified. The parameter values can be:

- XPath expressions referring to other fields and including function calls
- String literals defined in single quotes
- Raw integers or decimals

23.2 Launching external apps to populate multiple fields

Since v1.4.3, a `field-list` group can have an `intent` attribute that allows an external application to populate it.

23.2.1 XLSForm

Table 23.2: survey

type	name	label	appearance	body::intent
begin_group	mygroup	Fields to populate	field-list	org.mycompany.myapp(my_text='Some text', uuid=/myform/meta/instanceID)
text	sometext	Some text		
integer	someinteger	Some integer		
end_group				

```
<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/sometext">
    <label>Some text</label>
  </input>
  <input ref="/myform/mygroup/someinteger">
    <label>Some integer</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 bundle of values

returned by the external application contains values whose keys match the type and the name of the sub-fields, then these values 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, or integer. Any other sub-field is invisible to the external app.

CHAPTER

TWENTYFOUR

CONTRIBUTING TO ODK DOCS

24.1 Docs Technical Guide

This document explains the tools and workflows used when contributing to ODK Documentation.

Note: Developer and authoring tools have lots of options and alternatives. Local tools and workflows presented in this guide are what the authors feel would be easiest for a non-coding contributor to set up and use.

You should feel free to use your preferred tools.

24.1.1 Docs as Code

ODK Documentation follows (as much as possible) the Docs as Code philosophy.

This means:

- Documentation source files are written in a plain text format. (We use reStructuredText.)
- Documentation source files are kept under version control. (We use git and Github.)
- Documentation is built from source to a published output using a static site generator. (We use Sphinx.)
- Documentation builds are run, tested, and deployed automatically using continuous integration tools. (We use CircleCI.)

The 'Docs as Code' approach has many advantages, but we are aware that this approach can feel difficult for writers who aren't used to dealing with the command line. It can also be difficult for coders who are used to this approach, but who typically use simpler authoring tools like Jekyll and Markdown.

This section of the Contributor Guide walks through our authoring and publishing workflow and toolchain, to make it as easy as possible for you to contribute.

24.1.2 Overview of Workflow

When you first get started you'll need to:

1. Fork the ODK Docs repo your own Github Account
2. Clone it down to your local machine
3. Install dependencies

And then each time you work you will:

1. Make a branch for a specific task
2. Make commits as you go
3. Build and view the docs locally, running the style guide tests.
- Correct any errors and commit those changes
4. Push your branch to your Github fork
5. Issue a pull request against the current working branch of the main repo (usually `master`)
6. Pull the latest changes to `master` back to your local machine from the main repo.
7. Repeat

24.1.3 Setting up your Environment

Terminal (Command Line)

Warning: This contributor guide is written from a Linux/Unix (Bash terminal) perspective, which is relevant to all flavors of Linux and macOS. We consider the Bash terminal commands to be the canonical way to build and work with the docs.

We have also provided explanations for how to adapt these commands to the Windows Command Prompt. This is different than Windows Powershell, and not all the commands will work in Powershell. For more details on the Windows Command Prompt, see this article.

If you are on a Windows machine, you may prefer to use the adapted Windows instructions here. Alternatively, you can follow the Bash commands:

- use the Linux subsystem (Windows 10)
- use a bash terminal emulator, such as
  - *Cygwin*
  - gitbash

The lead maintainers of this docs repo are not Windows users, so we rely on our contributor community to keep Windows-specific information complete and accurate. Contributions to this guide with explanations and help for Windows users are greatly appreciated.

Contributing to the docs requires interacting with git, GitHub, Python, and Sphinx, which requires the use of the Terminal. This is common among Linux users. Mac users unfamiliar with the Terminal can learn more from this tutorial.

Python

Python 3

If you don't know, check to see if you have Python 3 installed:

```
$ python3
```

On windows:

```
> python
```

If you get an error, you probably don't have it and will need to *install Python 3*.

On Windows make sure to select the option "Add python to the Path" while installing, otherwise you'll need to add it manually.

See this video for more details.

If the Python command-line interpreter starts up, type `quit()` to exit.

Virtual Environment

A virtual environment is a Python tool for sandboxing dependencies. It lets you install whatever you need for a particular project, without messing up any other Python environments you might need.

Check to see if you have virtualenv installed:

```
$ virtualenv
```

If you get a help message with information about commands, you have it. If you don't have it, you'll get a `command not found` message.

If you don't have it, install it using `pip` by running:

```
$ pip install virtualenv
```

Then, create a directory called `odk`. This will contain your virtualenv and the docs repo as subdirectories.

```
$ mkdir odk
$ cd odk
```

Now, inside that directory, create a python3 virtualenv.

```
$ virtualenv -p python3 odkenv
```

On Windows use:

```
> path\to\python\python -m venv odkenv
```

The last part, `odkenv`, is the name of virtual environment. It can be whatever name you'd like to call it, but we'll use `odkenv` throughout these docs.

Activate your virtual environment with:

```
$ source odkenv/bin/activate
```

On Windows use:

```
> odkenv\Scripts\activate
```

When you are done working, deactivate it with:

```
$ deactivate
```

Github and git

Git is a distributed version control system. It makes it possible to track changes in files over time, merge changes made by different contributors, compare different versions of the same file, and revert a file to an earlier point.

Git is complicated, but you do not need to understand its advanced features or inner workings to use it.

GitHub is an online service for hosting git repositories. It also provides additional collaboration tools like issue trackers and project boards. Open Data Kit uses GitHub for its public code and documentation projects.

You will need to:

1. Install git
2. Make sure that git is installed properly by typing (git) in the terminal or command prompt
   - On Windows: If you get any errors, check if your environment variables are set up correctly. (See this StackOverflow answer for details.)
3. Get a GitHub account.

GLFS

We use Git Large File Storage (GLFS) to handle large binary files like images and videos. Once installed, you normally won't need to do anything else. GLFS is largely transparent when using git.

1. Install GLFS.

> **Warning: On Windows**
>
> Make sure `git-lfs.exe` and `git.exe` are under the same main directory on Windows. (See this page for reference.

GLFS tracks binary files as defined in the `.gitattributes` file in the repo. Most common binary file formats are already listed, but there might be others we haven't thought of yet.

Adding new file formats to GLFS tracking

If you are adding binary files to the repo, and they are in formats not already tracked, **it is your responsibility to make sure they are tracked by GLFS.**

To make sure they are properly tracked, add them to the `.gitattributes` file.

```
# file type section heading
*.{extension-to-track} filter=lfs diff=lfs merge=lfs -text
```

You can also use the command line.

```
$ glfs track *.{file-extension}
```

This will add a line to `.gitattributes`.

Note: Please keep `.gitattributes` organized by placing the new file format declaration in the appropriate section, or creating a new section as needed.

Warning: Updates to `.gitattributes` must be done in a commit before the commit that adds the new binary files.

We will not accept Pull Requests that include binary files untracked by GLFS.

Android Tools

Some testing and documentation tasks (including *making screenshots from ODK Collect*) require the *Android Debug Bridge* command line tool. You can either install Android Studio or install ADB as standalone SDK tool.

Android Studio

ADB (Android Debug Bridge) is part of Android Studio, and is typically installed by default when you install Android Studio.

This is the best way to get **adb** if you plan to do any other Android development. To use it from the command line, add the SDK Platform tools to your path.

On Mac, add the following to your `.bash_profile`

```
export PATH=$PATH:~/Library/Android/sdk/tools/
```

Note: On Windows, you have to run Android Studio once to complete the installation of ADB. The tool can be found in `C:/Users/user-name/AppData/Local/Android/sdk/platform-tools`. To add it to the environment variable path, use the following command:

```
set PATH=%PATH%;C:\Users\your user name\AppData\Local\Android\sdk\platform-tools
```

Warning: The path specified above assumes a default installation of Android Studio. You may have put Android Studio in a different location.

24.1.4 Getting ready to work

Fork the docs

Go to the ODK Doc repo on GitHub and use the *Fork* button (top right) to create your own copy. After the process completes, you'll be looking at your own fork on GitHub.

Clone to local

From your own fork of the repo on GitHub, select the *Clone or download* button. Copy the URI from the text box that opens up. It will be something like: `https://github.com/your-gh-username/docs.git`

Open your terminal, and *cd* to your preferred directory. Then *git clone* the repo:

```
$ git clone https://github.com/your-github-username/docs.git
.
.
.
$ cd docs
```

The rest of the documentation assumes you are in the directory for the repo (the directory containing `conf.py` and `index.rst`).

Tip:

- The `clone` command creates a new directory inside the current one. So you do not need to create a new *odk-docs* directory first.
- As noted above, we recommend a master `odk` directory that holds your virtualenv directory and your git repo in two separate subdirectories. So you would be in that master `odk` directory when you clone down the repo.
- Double check that the right folders are in the right places

```
- odk/
  - odkenv/
  - docs/
```

Set the upstream remote

When you clone down a repo, the local copy calls your GitHub copy `origin`. You should also set `upstream` as the name of the main ODK Docs GitHub repo.

```
$ git remote add upstream https://github.com/opendatakit/docs.git
```

Or in Windows:

```
> git remote add upstream https://github.com/opendatakit/docs.git
```

Run `git remote -v` to check the status, you should see something like this:

```
$ origin https://github.com/your-github-username/docs.git (fetch)
$ origin https://github.com/your-github-username/docs.git (push)
$ upstream https://github.com/opendatakit/docs.git (fetch)
$ upstream https://github.com/opendatakit/docs.git (push)
```

Install Dependencies

The first time you clone down the repo, you need to install the dependencies. Make sure you have your Python 3 virtual environment set up and activated, then:

```
$ pip install -r requirements.txt
```

Note: If you are working on the design, testing, or deployment of the docs, you might find the need to install an additional PyPi package. If you do, please update the `requirements.txt` file with **`pip freeze > requirements.txt`**. Pull Requests which change `requirements.txt` should include a note about why the new packages are needed.

Note: If you have problems when running the Sphinx commands (see below), you may have a dependency issue. Try running **`pip install -r requirements.txt`** again.

24.1.5 Workflow details

Pull in changes from upstream

As other people make changes to the docs, you need to keep your local copy up to date.

You probably won't need to do this the first time, but you should always pull in any changes from the main repository before working.

```
$ git pull upstream
```

Note: If you get this message:

```
You asked to pull from the remote 'upstream', but did not specify a branch.
Because this is not the default configured remote for your current branch,
you must specify a branch on the command line.
```

Try running `git pull upstream master` instead.

Make a New Branch

Choose a specific, deliverable task to work on. This should be an active issue from our issue tracker on GitHub.

Create a new branch in which you will work on this specific issue. The branch name should briefly describe what you are doing. For example, the original author of this contributor guide worked in a branch called `contributing`.

Also, make sure that all the branches are derived from `master`, to avoid mixing up work from different issues commits.

```
$ git checkout -b branch-name
```

Tip: Branch names should be short, lowercase, and use hyphens for separators.

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`

Work on the Docs

Write and edit files in your favorite editor.

Note: To work on ODK Docs, you need to work in a code editor.

If you've never used a code editor before, you should know that they are a little different than other writing environments like MS Word or your email editor.

People have strong opinions about code editors, and nearly everyone who uses them regularly has a favorite.

If you're new to using an editor, you might want to try Atom or Sublime, which are both popular and easy to use, and they both have decent support for reStructuredText syntax.

Spell check your work

After making changes to the docs, you need to run the spell checker. To run the spell checker:

```
$ sphinx-build -b spelling src build/spelling
```

If there are any warnings, make sure that you fix them to avoid build failure.

The error messages will be displayed on terminal as well as stored in a file `/build/spelling/output.txt`.

If you find a word which is not misspelled and will have repeated use in docs, add it to the spelling list in the file `/src/spelling_wordlist.txt`.

If you find a word which is not misspelled and is only required in a particular file, use the `spelling` directive to list the words before the file content.

```
.. spelling::

  vN.N

Upgrading Aggregate
=======================
```

Style-test your work

After making changes to the docs, you need to run the style-guide checks. To run the style guide checks on all the docs:

```
$ python style-test.py
```

To run the style guide checks on specified files:

```
$ python style-test.py filename1.rst filename2.rst ...
```

If you want to run the style guide checks on the files you modified, use the option `-d` or `--diff`.

```
$ python style-test.py -d
```

Note: Using this option, only checks the files that have been edited since the last **`git commit`**. So, if you modify the files, make sure to check them before making a commit.

To run the tests on modified files and some other specified files:

```
$ python style-test.py -d filename1.rst filename2.rst ...
```

The output will consist of a list of warnings and errors. Make sure to go through the warnings and eliminate the ones which violate the style guide rules.

Note: It is not necessary to fix all the warnings but you should go through each warning and decide if a change makes sense.

If there are any errors, you need to fix them to avoid build failure. You can manually fix the errors or use the option `-f` or `--fix`.

```
$ python style-test.py -f
```

If you want to fix the errors in some specified files:

```
$ python style-test.py -f filename1.rst filename2.rst ...
```

Tip: After automatic fixing of errors, make sure to go through the changes made.

To ignore any part of a file from being checked, you can enclose it in comments:

```
.. startignore

Some text which is to be ignored while testing.

.. endignore
```

You can also generate `.csv` format output using the `-o` or `--out_path` with the output filename:

```
$ python style-test.py -o output.csv
```

To generate output for a some specified files:

```
$ python style-test.py -o output.csv filename1.rst filename2.rst ...
```

Note: Make sure to specify the output file before the input files when you use the option `-o`.

When you run the style guide checks, style testing scripts are generated before the test starts and are removed automatically after the testing is complete. If you want to keep these scripts for debugging, use the option `-s` or `--store`:

```
$ python style-test.py -s
```

Use the option `-h` or `--help` to get any help about style guide testing:

```
$ python style-test.py -h
```

Build, View, and Debug

To build the documentation into a viewable website:

```
$ sphinx-build -b dirhtml src build
```

This calls the sphinx-build utility. The `-b` switch specifies the builder, which in this case is `html`. `src` refers to the src directory which contains all `.rst` files (the build source) and `build` refers to the target of the build (the built files will be put into a directory labeled `build`).

When you run the build, you may see error or warning messages. These indicate potential problems with the documentation, like:

- syntax errors
- broken links
- terms not included in the glossary

Error and warning messages include a file name and line number for tracking them down. Try to resolve all your errors and warnings before issuing a pull request. If this is not possible, please add a note in your pull request so that we can help you debug the problem.

We will not merge Pull Requests that have warnings or errors in them.

Note: Because of a bug in Sphinx the line numbers in error and warning messages will be off by the length of *rst_prolog* in `conf.py`.

To view the documentation in your web browser, you can use Python's built-in web server.

```
$ cd build
$ python -m http.server 8000
```

Then open your browser and go to http://localhost:8000

Read through your doc edits in the browser and correct any issues in your source files. You'll need to shut down the web server (`CTRL C`) before rebuilding, then return to the main directory of the repo (**`cd ..`**).

It's a good idea to delete the `build` directory before each rebuild.

```
$ rm -rf build
$ sphinx-build -b dirhtml . build
```

Tip: The script `b.sh` automatically runs all the build commands. It saves typing. In the future, it will also become the canonical build script for ODK Docs, including additional tests and other build tasks.

Push Your Branch

Once your work on the issue is completed, add the files you've changed or created, and write a relevant commit message describing the changes.

```
$ git add my_changed_files
$ git commit -m "A small but relevant commit message"
```

Then, push the changes. The first time you do this on any branch, you'll need to specify the branch name:

```
$ git push origin branch-name
```

After that, you only need to use the **`push`** command:

```
$ git push
```

Note: `origin` is the local label for your GitHub fork.

Issue a Pull Request

A *pull request* (or PR) is a request from you to the ODK Docs maintainers, for us to pull in your changes to the main repo.

Go the main docs repo on GitHub. You'll see a message there referencing your recently pushed branches. Select *Compare & pull request* to start a pull request.

Follow GitHub's instructions. The *Base fork* should be the main repo, and *base* should be `master`. Your repo and working fork should be listed beside them. (This should all populate by default, but be sure to double check.) If there is a green **Able to be merged** message, you can proceed.

You must include a PR comment. Things to include:

- A summary of what you did.
- A note about anything that probably should have been done, but you didn't do.
- A note about any new work this PR will create.
- The issue number you are working on. If the PR completes the issue, include the text `Closes #` and the issue number.
- A note about any errors or warnings, and why you did not or could not resolve them.
- A note justifying any changes to `requirements.txt`.
- A note about any difficulties, questions, or concerns that came up while working on this issue.

Complete the pull request. The maintainers will review it as quickly as possible. If there are any problems the maintainers can't deal with, they will reach out to you.

Note: If you happen to rename any document file (`*.rst`), then be sure that you add the redirect in your PR.

To add the redirect go to `s3_website.yml`, and add a mapping from the old file name to the new file name below the **redirects:** line, one mapping per line.

If you have renamed `old-name.rst` to `new-name.rst`:

```
redirects:
 old-name/index.html: new-name
```

Notice the inclusion of `/index.html` on the left side.

Keep Going

Once the PR is merged, you'll need to pull in the changes from the main repo (`upstream`) into your local copy.

```
$ git checkout master
$ git pull upstream master
```

Then you should push those change to your copy on GitHub (`origin`).

```
$ git push
```

If you want to delete your branch from before, you can do that:

```
$ git branch -d branch-name
```

Now you can find a new issue to work on, create a new branch, and get to work again.

24.2 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.

24.2.1 Indentation

Indentation is meaningful in Sphinx and reStructuredText text. Usually, indenting a section means that is "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**.
- 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.
```

24.2.2 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.

```
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.)

```
******************
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.

24.2.3 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.

```
.. 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.

```
.. 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.opendatakit.org/page-name`.

If we used subdirectories, then our URIs would look like `docs.opendatakit.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.

24.2.4 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:

```
Title of the Page - <h1> - Equal Signs
======================================


Major Section - <h2> - Hyphens
-------------------------------


Subsection - <h3> - Tildes
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
Sub-subsection - <h4> - Double Quotes
"""""""""""""""""""""""""""""""""""""""""""


Sub-sub-subsection - <h5> - Single Quotes
'''''''''''''''''''''''''''''''''''''''''''''''''''
```

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.

```
************************************************
Page Title - <h1> - Asterisks above and below
************************************************


Major Section - <h2> - Equal Signs
=======================================


Subsection - <h3> - Hyphens
----------------------------------


Sub-subsection - <h4> - Tildes
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~


Sub-sub-subsection - <h5> - Double Quotes
"""""""""""""""""""""""""""""""""""""""""""""""


Sub-sub-sub-subsection - <h6> - Single Quotes
'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
```

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

```
.. _section-label:

Section Title
----------------

Lorem ipsum content of section blah blah.
```

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...

.. _aggregate-getting-started:

Getting Started
-----------------
```

24.2.5 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
- 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

#. 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.

Note: See *Ordered and unordered lists* in the *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.

Tables

Grid style

```
+------------+------------+-----------+
| Header 1   | Header 2   | Header 3  |
+============+============+===========+
| body row 1 | column 2   | column 3  |
+------------+------------+-----------+
| body row 2 | Cells may span columns.|
+------------+------------+-----------+
| body row 3 | Cells may  | - Cells   |
+------------+ span rows. | - contain |
| body row 4 |            | - blocks. |
+------------+------------+-----------+
```

Header 1	Header 2	Header 3
body row 1	column 2	column 3
body row 2	Cells may span columns.	
body row 3	Cells may span rows.	• Cells • contain • blocks.
body row 4		

Simple style

```
=====  =====  ======
   Inputs     Output
------------  ------
  A      B    A or B
=====  =====  ======
False  False  False
True   False  True
```

```
False  True   True
True   True   True
=====  =====  ======
```

Inputs		Output
A	B	A or B
False	False	False
True	False	True
False	True	True
True	True	True

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).

```
.. 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 24.1: Example Table

Treat	Quantity	Description
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!

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.

```
.. csv-table:: Table using # as delimiter
  :header: "Name", "Grade"
  :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 24.2: Table aligned to right

Name	Grade
Peter	A
Paul	B

: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"
   "Paul", "B"
```

Table 24.3: Table with different number of columns in each row

Name	Grade
Peter	
Paul	B

- 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.

24.2.6 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:

```
: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>`
```

- Link to this document:
  - *Contributing to ODK Docs*
  - *anchor text*
- Link to this section:
  - *Cross referencing*
  - *anchor text*
- Link to a term:
  - *participant*
  - *anchor text*

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.

:kbd:
Marks up a sequence of literal keyboard strokes.

```
To stop the local server, type :kbd:`CTRL C`.
```

:command:
Marks up a terminal command.

```
To build the documentation, use :command:`sphinx-build`.
```

:option:
Marks up a terminal command option.

```
The :option:`-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 :th:`appearance` attribute␣
↪begins with :tc:`ex:`.
```

:formstate:

Specifies the state of the form in *ODK Collect*, which could be one of the following:

- Blank
- Finalized
- Saved
- Sent
- Deleted

```
:formstate:`Sent`
```

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.

```
:abbr:`ODK (Open Data Kit)`
```

:dfn:

Marks the defining instance of a term outside the glossary.

```
:dfn:`Open Data Kit` (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.

```
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.

```
launch the :program:`ODK Aggregate Installer`
```

Images and Figures

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

You must perform lossless compression on the source images. Use one of the following tools to optimize the images:

- **ImageOptim** is a tool that allows us to optimize the images. It is not format specific which means it can optimize both jpeg as well as png images. You can download it from here . After launching ImageOptim.app, dragging and dropping images into its window gives you an in-place optimized file.
- **Pngout** is another option for optimizing png images. Installation and usage instructions can be found here .
- **Mozjpeg** can be used to optimize jpeg images. Installation and related information can be found on this link .

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`
- `3987948p2983768oh184692p094.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`

Screenshots from ODK Collect

If you have set up local *Android Tools*, 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:

```
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.

`.. 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 Collect

Android Debug Bridge (ADB) can be used to capture a screen recording from *ODK Collect*.

```
$ 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.

```
.. code-block:: rst

  Use the ``code-block`` directive to markup code samples.

.. code-block:: python

  print("Hello ODK!")

.. code-block:: console

  $ python --version

.. code-block:: 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-block:: rst
  :class: unwrap
```

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.

```
.. code-block:: python
  :class: wrap
```

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://opendatakit.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.

- To create a hyperlink reference for ODK Forum, use `|forum|_`.

```
You can ask support questions in |forum|_.
```

You can ask support questions in ODK Forum.

- To create a hyperlink reference for contributors guide, use `|contrib-guide|_`.

```
Be sure to read the |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 |arrow| icon opens the jump menu.

.. |arrow| image:: /img/{document-subdirectory}/{file}.*
           :alt: Alt text.
```

24.3 Style Guide

24.3.1 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 :guilabel:`Save`.
```

Wrong

```
You may see an error message that says, "Something went wrong."
```

Right

```
You may get an error: ``Something went wrong.``
```

```
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.

```
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)
        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.
```

```
@memoize
def check_comma(text):
    """Use serial comma after penultimate item."""
    err = "style-guide.serial-comma"
    msg = "Use serial comma after penultimate item."
    regex = "\,\s[a-zA-Z0-9]+\sand\s"

    return existence_check(text, [regex], err, msg, require_padding=False)
```

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
- GitHub
- Python
```

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
```

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\.e\.", "ie", "\*ie\.\*", "\*ie\*",
        "e\.g\.", "eg", "\*eg\.\*", "\*eg\*",
```

```
        "viz\.", "viz", "\*viz\.\*", "\*viz\*",
        "c\.f\.", "cf", "\*cf\.\*", "\*cf\*",
        "n\.b\.", "nb", "\*nb\.\*", "\*nb\*",
        "q\.v\.", "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.
```

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.

```
@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.
```

```
@memoize
def check_semicolon(text):
    """Avoid using semicolon."""
    err = "style-guide.check-semicolon"
    msg = "Avoid using semicolon."
    regex = ";"

    return existence_check(text, [regex], err, msg, require_padding=False)
```

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.

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...
```

Right

```
ODK Collect is an Android app. It can be used for...
```

Right

```
ODK Collect is an Android app that is used to...
```

```
@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

```
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.
```

```
@memoize
def check_howto(text):
    """Avoid using how to construct."""
    err = "style-guide.check-howto"
    msg = "Avoid using \"How to\" construction."
    regex = "(How to.*)(\n)([=~\-\"\*]+)"

    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 *Form Widgets* document.

In these cases, you may want to use the `rubric` directive.

```
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,})"

    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.

24.3.2 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="dcd98b7102dd2f0e8b11d0f600bfb0c093",
                         opaque="5ccc069c403ebaf9f0171e9517f40e41"
Content-Type: text/html
Content-Length: 311
```

Meaningful names

When writing sample code, avoid meaningless names.

Wrong

```
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**:

```
<element>
  Some content.
</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.

```
<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.

```
<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*.

```
<element>
  Content.
  <child-element>
    More content.
  </child-element>
</element>
```

When an element is empty, it can be called a *null element*.

```
<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.

```
<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.

24.3.3 ODK jargon

ODK and ODK Docs

Wrong

- Odk
- odk
- Open data kit
- OpenDataKit
- the Open Data Kit
- ODK docs
- ODK documentation

Right

- ODK
- Open Data Kit
- ODK Docs
- ODK Documentation

Probably want to avoid...

- Open Data Kit Documentation

```
@memoize
def check_odkspell(text):
    """ODK spelling usage."""
    err = "style-guide.spelling-odk"
    msg = "ODK spell check. '{}' is the preferred usage."

    preferences = [

        ["Open Data Kit",         ["Open data kit"]],
        ["Open Data Kit",         ["OpenDataKit"]],
        ["ODK",                   ["Odk"]],
        ["ODK",                   ["{0} odk"]],
        ["ODK Docs",              ["ODK docs"]],
```

```
        ["ODK Documentation",     ["ODK documentation"]]
    ]

    return preferred_forms_check(text, preferences, err, msg, ignore_case=False)
```

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

```
@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.

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

```
@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)
```

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.
- An online tool and an offline 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.

24.4 Docs Developer Guide

This document is for contributors working on the design, templating, deployment, or development of the ODK Docs website.

24.4.1 Tech Overview

ODK Docs uses:

- Sphinx, a static-site generator written in Python

  Sphinx uses:

  - Docutils for parsing reStructuredText
  - Jinja for templating

- 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

24.4.2 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.

24.4.3 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.

24.4.4 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:

```
/* Example CSS PR #xyx */

div[class^='example'] {
  color: black;
}
```

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:

```
/* New section starts */

/* Example CSS PR #xyx */

div[class^='example'] {
  color: black;
}

/* New section ends */
```

24.4.5 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.

`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

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

```
@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**.

```
.. code-block:: python
  :class: style-checks
```

The generated file after parsing code for style checks is `style-checks.py`.

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

```
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 an built-in proselint check, specify the check name in the check list in the function `exclude_checks()` in the file `style-test.py`.

24.5 Tips for Making Good Contributions

24.5.1 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

24.5.2 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

24.5.3 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.

24.5.4 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.

24.5.5 Claim issues

If you decide to work on an issue, let the community know you are working on it by *claiming* the issue.

> @opendatakit-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.

24.5.6 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.

24.5.7 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.

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.

> @opendatakit-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.)

24.5.8 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.

24.5.9 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.

24.5.10 Actually install and use Open Data Kit 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, follow the *Getting Started With ODK* guide so that you are familiar with the core ODK tools.

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.

24.5.11 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.

24.5.12 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.**

24.6 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).

24.6.1 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

24.6.2 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

- 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 < https://pypi.python.org/pypi/pip>_* 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:

```
> 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).

```
> pip install virtualenv
```

Create a new directory for your odkdocs work:

```
> 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.

```
> virtualenv -p <python path/python.exe> odkenv
```

After creating the virtualenv, multiple files are copied into the folder odkenv in your working directory.

```
> ls odkenv
```

The folder Scripts contains all virtualenv controls as ".bat" files.

To activate the odkenv:

```
> 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 user␣
↪name\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/opendatakit/docs.git
```

24.6.3 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.

CHAPTER

TWENTYFIVE

OPENROSA

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.

25.1 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.

25.1.1 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.

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="opendatakit.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 implementor's registered domain name should be used as part of the scheme (for example:`opendatakit.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

- 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>
```

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.

```
<instanceID>opendatakit.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>`).

```
<model>
  <instance>
    <data xmlns:jr="http://openrosa.org/xforms"
          id="example.org:myFormId"
          version="1" >
      <jr:meta>
        <jr:timeStart/>
        <jr:timeEnd/>
        <jr:instanceID/>
        </jr:meta>
```

Examples

Blank form metadata

```
<h:head>
  <h:title>Metablock example</h:title>
  <model>
    <instance>
      <data xmlns:jr="http://openrosa.org/xforms"
            xmlns="http://example.org/meta"
            version="1" >
        <jr:meta>
          <jr:deviceID/>
          <jr:timeStart/>
          <jr:timeEnd/>
          <jr:instanceID/>
        </jr:meta>
        .
        .
        .
      </data>
    </instance>
    .
    .
    .
  </model>
</h:head>
```

Form submission metadata

```
<?xml version='1.0'?>
  <data version="1"
        xmlns:jr="http://openrosa.org/xforms"
        xmlns="http://example.org/meta">
    <jr:meta>
      <jr:deviceID>uuid:38DN0236SAKWJOJNQB3XJI9RW</jr:deviceID>
      <jr:timeStart>2010-08-12T04:08:29.765-5:00</jr:timeStart>
      <jr:timeEnd>2010-08-12T04:10:23.062-5:00</jr:timeEnd>
      <jr:instanceID>uuid.dimagi.org:GEPSJLOGH13TY8L77066GEJJW</jr:instanceID>
      <jr:userID>chwid.dimagi.org:Akende</jr:userID>
    </jr:meta>
    .
    .
    .
```

```
</data>
```

25.1.2 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:

Header	Values	Required
Accept-Language	The key for what language a response is expected in.	No. Response acceptable in any locale
X-OpenRosa-Version	1.0	Yes
Date	The date on the device in format: *Mon, 14 Feb 2011 16:48:15 GMT*	Yes

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.

Arg	Values	Required
items	true|false	No. Defaults to false

HTTP Responses

Servers should include the following headers in all responses.

Header	Values	Required
Content-Language	Two-letter language code (ISO 639-1)	No. Recommended for internationalization.
X-OpenRosa-Version	1.0	Yes
Date	The date on the server in the format: *Mon, 14 Feb 2011 16:48:15 GMT*	Yes

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␣
↪to group messages by type -->
        <!-- PAYLOADS HERE-->                                                ␣
↪           <!-- 0 or many: additional payloads to be parsed per platform-->
    </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"/> <!-- optional --
↪>
    <!--... Some platform specific response ...-->
</OpenRosaResponse>
```

APIs using this response can then include their namespaced 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>
```

```
...
  <OpenRosaResponse xmlns="http://openrosa.org/http/response">
      <message nature="submit_success">Thanks, you've submitted 5 forms today </
↪message>
...
  </OpenRosaResponse>
...
  <OpenRosaResponse xmlns="http://openrosa.org/http/response">
      <message nature="submit_user_registered">User 'paul' created succesfully!</
↪message>
...
  </OpenRosaResponse>
...
  <OpenRosaResponse xmlns="http://openrosa.org/http/response">
      <message nature="submit_success">Thanks, you've submitted 6 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!

25.1.3 Authentication API

This standard specifies the Request and Response format through which OpenRosa compliant servers authorize HTTP transactions.

Overview

This API provides a standardized 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.

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

- 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

RFC2069

```
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:

  response = MD5(HA1:nonce:nonceCount:clientNonce:qop:HA2)

if qop directive is unspecified
```

```
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
```

25.1.4 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.

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 AppEngine will be rejected prior to being sent to any server, and will therefore not contain this (or any) OpenRosa header.

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:

```
<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.

Code	HTTP Meaning	ODK Meaning
200	UNUSED	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.
201	Form Received	Everything went great. Thanks for submitting.
202	Accepted	We got and saved your data, but may not have fully processed it. You should not try to resubmit.
204	No Content	Status returned in response to a HEAD request.
401	Unauthorized	Client tried to post something it didn't have permission to post.
403	Forbidden	You're not allowed to post to this server.
404	Not Found	Unknown URI endpoint, domain, or other
413	Request too large	The request body is too large for the server to process
500	Internal Server Error	Something went awry on the server and we're not sure what it was

25.1.5 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 implementor- 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 version='1.0' encoding='UTF-8' ?>
<xforms xmlns="http://openrosa.org/xforms/xformsList">
  <xform>
    <formID>mydomain.org:formId</id>
    <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</
↪namifestUrl>
  </xform>
  <xform>
    <formID>http://mydomain.org/uniqueFormXmlns</id>
    <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

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 version='1.0' encoding='UTF-8' ?>
<manifest xmlns="http://openrosa.org/xforms/xformsManifest">
 <mediaFile>
  <filename>badger.png</filename>
  <hash>md5:c28fc778a9291672baddd04ac880a05d</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/>`

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.

CHAPTER

TWENTYSIX

ODK JAVAROSA

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

26.1 Architecture

26.2 Integration Points

This section describes how the different parts of ODK use JavaRosa.

26.2.1 Aggregate

26.2.2 Briefcase

26.2.3 Collect

26.2.4 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
```

CHAPTER

TWENTYSEVEN

LAUNCHING ODK COLLECT FROM EXTERNAL APPS

See also:

Launching External Apps from Collect Forms

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.

27.1 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.

27.2 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.

27.2.1 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);
```

This displays a list of forms and allows the user to select one and fill it.

Similarly for an instance of the form:

```
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.

27.2.2 Getting the URI of a form or instance chosen by the user

```
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 `onActivityResultMethod` in the following way:

```
@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:

```
intent.setType("vnd.android.cursor.dir/vnd.odk.instance");
```

27.2.3 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.

```
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

TWENTYEIGHT

BRIEFCASE AGGREGATE API

28.1 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.

28.2 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.

28.3 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)

28.4 Interface Details

28.4.1 OpenRosa Form Discovery API

Vanilla conformance.

28.4.2 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:

```
<OpenRosaResponse xmlns="http://openrosa.org/http/response">
<message>...</message>
<submissionMetadata xmlns="http://www.opendatakit.org/xforms"
        id="formid" version="..." encrypted="yes" instanceID="..."
        submissionDate="..."
        isComplete="true" markedAsCompleteDate="..."/>
```

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:

```
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.

28.4.3 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.

28.4.4 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:

```
<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>
```

```
   <td>
    <input id="form_def_file" class="gwt-Button" type="file"
           name="form_def_file" size="80">
   </td>
  </tr>
  <tr>
   <td>
    <label for="mediaFiles">Optional Media file(s):</label>
   </td>
   <td>
    <input id="mediaFiles" class="gwt-Button" type="file"
           multiple="" name="datafile" size="80,20">
    <input id="clear_media_files" class="gwt-Button"
           type="button"
           onclick="clearMediaInputField('mediaFiles')"
           value="Clear">
   </td>
  </tr>
  <tr>
   <td>
    <input class="gwt-Button" type="submit"
           value="Upload Form" name="button">
   </td>
  </tr>
 </tbody>
</table>
</form>
```

28.4.5 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:

```
<idChunk xmlns="http://opendatakit.org/submissions">
 <idList>
  <id>uuid:e5aa4247-cfb8-4cc5-87cd-52fbfe491b13</id>
  <id>uuid:8ca4433a-fed2-4d1c-b333-c0378203f68f</id>
 </idList>
 <resumptionCursor>opaquedata</resumptionCursor>
</idChunk>
```

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.

28.4.6 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:

```
<submission xmlns="http://opendatakit.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.

CHAPTER

TWENTYNINE

ENCRYPTED FORMS

29.1 Overview

Encrypted forms provide a mechanism to keep your data private even when using **http:** for communications (e.g., when you do not have an **SSL certificate** or **https:** is not available). Encrypted forms may also enable Google App Engine deployments (and deployments using other web database services, e.g., AWS) 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 up to ODK Aggregate and downloaded to ODK Briefcase. ODK Briefcase, when supplied with the asymmetric private key (which ODK Briefcase never stores), can then decrypt and export the form data as a CSV file for your use.

This process ensures that the finalized form's data (and media attachments) are encrypted before being submitted to ODK Aggregate, remain encrypted while stored on ODK Aggregate, and remain encrypted as the data and attachments are pulled into ODK Briefcase, where they are again stored in encrypted form.

Note:

- ODK Aggregate cannot meaningfully publish encrypted forms to Google Spreadsheets or Fusion Tables since the encryption obscures the entire contents of the form and ODK Aggregate never possesses the asymmetric key required to decrypt the form.
- When using encrypted forms, ODK Aggregate serves only as a data aggregation point — you must download, decrypt, and export the data using ODK Briefcase to access the unencrypted data

The non-encrypted data is available on the ODK 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), ODK 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 ODK Aggregate, with the encrypted form contents and its encrypted attachments appearing as attachments to this submission manifest "form."

29.2 Requirements

Encrypted forms require the following *minimum versions* of the ODK tools

- *ODK Collect 1.2 Release Candidate 1 (RC1) or higher*
- *ODK Aggregate 1.0.4 Production or higher*
- *ODK Briefcase 1.0 Production or higher*

Warning: Encrypted form definitions must include a unique OpenRosa instanceID and have an explicit `<submission/>` element

These requirements are covered in more detail below.

29.3 Security Concerns

While ODK Collect attempts to remove all unencrypted copies of a finalized form and its attachments from the device, because ODK 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 ODK 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.

29.4 Configuration

For ODK Briefcase, you must download and install the *Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 6* from the Java download site; for previous versions of Java, it is available here. This is required for decryption to be successful.

To install the JCE:

1. Select the downloaded UCE zip archive file in the file viewer.
2. Right-click, choose *Extract files…* and extract the files from the downloaded zip archive.
3. Navigate into the extracted directory tree and copy the `local_policy.jar` and `US_export_policy.jar` files to the `libsecurity` directory under the installation directory of the Java Runtime Environment (JRE) of your computer, replacing earlier versions of these files.

29.4.1 Windows

The JRE is usually installed here:

`C:\Program Files\Java\jre1.x.x_xxx\lib\security`

You might also have a JDK. If you do, you must also install the JCE files there:

`C:\Program Files\Java\jdk1.x.x_xxx\jre\lib\security`

29.4.2 macOS

The JRE is usually found here:

`/Library/Internet Plug-Ins/JavaAppletPlugin.plugin/Contents/Home/lib/security`

You might also have a JDK. If you do, you must also install the JCE files there:

`/Library/Java/JavaVirtualMachines/jdk1.x.x_xxx.jdk/Contents/Home/jre/lib/security`

29.5 Uploading Finalized Forms

```
<h:html xmlns="http://www.w3.org/2002/xforms"
     xmlns:h="http://www.w3.org/1999/xhtml"
     xmlns:orx="http://openrosa.org/xforms/">
<h:head>
```

```
<h:title>Sample Form</h:title>
<model>
 <itext>
   <translation lang="English" default="">
     <text id="ask_name">
       <value form="long">Please enter your name:</value>
       <value form="short">Respondent's name</value>
     </text>
   </translation>
 </itext>
 <instance>
   <sample id="sample-v1.0">
     <orx:meta>
        <orx:instanceID/>
     </orx:meta>
     <name/>
   </sample>
 </instance>
 <submission method="form-data-post"
             action="https://my-opendatakit.appspot.com/submission"
             base64RsaPublicKey="MIIBIjANB...JCwIDAQAB" />
 <bind nodeset="/sample/meta/instanceID" type="string" readonly="true()"
       calculate="concat('uuid:', uuid())"/>
 <bind nodeset="/sample/name" type="string" />
 </model>
 </h:head>
 <h:body>
 <input ref="name">
    <label ref="jr:itext('ask_name')"/>
 </input>
 </h:body>
 </h:html>
```

If you are using *XLSForm*, then form encryption is governed by the *settings* on the Settings Worksheet. Encrypted forms must specify a *submission_url* and a *public_key* on this worksheet. If both are specified, XLSForm will generate an encrypted-form definition. Skip to the following sections to see how to create a public-private key pair and specify the public key.

The required element to make this form an encrypted form is the `<submission/>` tag. Within this tag, the method attribute should always be **form-data-post**. The action attribute should be the url to which the submission should be posted; this is the ODK Aggregate website url with Aggregate.html replaced by submission. Finally, what identifies the form as an encrypted form is the presence of a *base64RsaPublicKey* attribute. This should be the base64 encoding of the RSA public key that ODK Collect uses to encrypt the symmetric encryption key it creates to encrypt a finalized instance of this form (a different symmetric

encryption key is created for every finalized form)

Note:

- The presence of the OpenRosa metadata block, as defined here: OpenRosa 1.0 Metadata Schema; is required.
- You can define any value for the instanceID field, but it must be unique across all collected surveys.
- As shown, the `<bind/>` calculates an instanceID value comparable to the instanceID value ODK Aggregate will generate when a form does not have an instanceID field.
- If you use punctuation other than colon and dash, or any special characters, please test thoroughly to ensure that ODK Aggregate correctly handles those characters during submissions and when the form is pulled down to ODK Briefcase.

29.6 Creating RSA Key pair

RSA public-private key pairs are generated using the OpenSSL software package. This is pre-installed on OSX and Linux but needs to be downloaded and installed on Windows.

29.6.1 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)

29.6.2 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. e.g.,

```
> cd C:\OpenSSL-Win32\bin
```

If you are on a Mac, open the terminal. Change directories to your Desktop.

```
$ cd ~/Desktop
```

29.6.3 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
```

29.6.4 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.

29.6.5 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 ODK Briefcase when decrypting the data.

29.6.6 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).

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 checkout the tutorial encrypted-XLSForm.It is for viewing purpose only but you can make your own copy to edit it.

29.7 Operations

Operationally, you would add the form definition to the ODK Aggregate server identified in the `<submission>` tag's action attribute, and deploy everything using ODK Collect 1.2 RC1 or later, 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 ODK Aggregate, use ODK Briefcase to download the encrypted submissions to your PC, and then specify the private key PEM file when decrypting and generating the CSV files.

Note:

- ODK Aggregate will only hold the encrypted submission with no access to the private key
- ODK 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 at-

tachments are missing or if there was tampering (other than the wholesale replacement of a submission, which can't be detected).

CHAPTER

THIRTY

SECURITY AND PRIVACY

Below is an attempt to enumerate known security and privacy considerations of the ODK software.

30.1 1. License Terms

The ODK software is released under the Apache 2 License.

All our installers, programs, and source code are provided AS-IS with no warranty or conditions, and without any liability obligations. See the license text for details.

30.2 2. Communication Channels

None of the downloadable ODK software transmits or communicates any information back to us and 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.

Our user website (opendatakit.org) does not knowingly use cookies, does not knowingly contain 3rd party ads, and does not collect personally identifiable information from the general public.

30.3 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 people may want to review the libraries and source code on our source code site.

30.4 4. ODK Aggregate Communications

When setting up your own webserver 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.

Only transmissions over an https:// connection are obscured from observers.

The definition of an encrypted form (*check here*) is transmitted in plaintext (unencrypted) to the device. When a filled-in submission for an encrypted form is finalized, it is encrypted on the device and transmitted in encrypted form. While this may meet requirements for obscured transmission over unsecured http:// connections, unsecured connections still allow observers to alter the form definition to potentially remove the encryption, capture any filled-in forms, or potentially intercept and thereby prevent their transmission to your server.

30.5 5. ODK Aggregate Deployments to Google App Engine or 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.

The form definition and associated media files of an encrypted form (*ODK see here*) are stored on the server in plaintext (unencrypted). When a filled-in submission for an encrypted form is finalized, it is encrypted on the device and transmitted to the server in encrypted form, where it is stored. The secret key required for decryption is not stored on the server, thereby preventing anyone at the hosting service from seeing your filled-in form data and attachments unless they break the encryption.

See *Planning Your Aggregate Deployment* for other considerations.

30.6 6. Encrypted Form Security

The form definition and associated media files of an *ODK encrypted form* are stored on the server in plaintext (unencrypted). And are transmitted and stored on the devices in plaintext.

Prior to finalizing a filled-in form, all form data and attachments are stored in plaintext (unencrypted) on the device.

At the time a filled-in form is finalized, a random 256-bit encryption/decryption key is generated for that filled-in form using the SecureRandom number generator (found here). This ensures that every filled-in form has its own unique 256-bit encryption/decryption key.

The filled-in form data and all media attachments are then encrypted with that key using 256-bit AES Cipher Feedback (CFB) streaming-block encryption. Once encrypted, all plaintext files and attachments for that filled-in form 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) filled-in form data and attachments may remain in the free-list of the SDCard until they are overwritten with new content.

On the server, if an observer were able to access your encrypted data, since each filled-in submission uses a different key, each submission would need to be cracked separately.

At this point in time, cracking AES encryption is viewed as impossible for all but the most advanced governmental agencies (e.g., the NSA).

30.7 7. ODK Aggregate 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.

30.8 8. Google Gmail Authentication

For authentication of Google accounts (Gmail or Google Apps), *ODK Aggregate* accepts OAuth2 tokens with rights to view a user's e-mail address (just the address - not the e-mail or user profile) as proof-of-identity.

This is a very weak proof-of-identity; every time you authorize Google to share your e-mail address with other sites or applications, those sites or applications have the permissions necessary to act on your behalf on *ODK Aggregate* (should they want to).

For this reason, it may be inappropriate to declare and grant Google e-mail addresses access to your site; this access is required for ODK 2.0 Sync functionality at rev 128 and earlier.

30.9 9. Identifying Information Transmission and Storage

During data submission, some identifying information is transmitted and stored on the server:

1. *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.

2. If *ODK Aggregate* is configured to require authentication (username / password or Google account) for submission (i.e., the Data Collector permission is NOT granted to the anonymousUser), then the username (or Google account) 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 or Google account 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, etc., will cause the authenticated username or Google account 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 or Google account will be identified in the server log together with summary information on what was deleted.

30.10 10. ODK Collect

We gather anonymous aggregate user behavior through Google Analytics. We use secure HTTPS communication to transfer this data off the device and the data are available to ODK's maintainers. Users may disable analytics in the settings of *ODK Collect*.

30.11 11. ODK Build

We require secure HTTPS connections to ODK Build. We gather anonymous aggregate user behavior through Google Analytics. We use secure HTTPS communication to transfer this data and the data are available to ODK's maintainers.

30.12 12. Online XLSForm tool

The online *XLSForm* tool does not use a secure connection. This means that your form definition files (both XLS and XML) are visible to a determined observer when submitted and downloaded from that site, as are any reported errors in the form.

Furthermore, the online XLSForm tool 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).

The offline XLSForm tool and the ODK Validate tool, because they operate locally without any network communications, provide a secure alternative to the convenience of this online tool.

30.13 13. Websites

Our websites, downloads site, and the online XLSForm tool can or do use cookies and can or do log all interactions. We also utilize security software, spam-blocking, and web-analytics tools (e.g., Google Web Analytics) that may track visitors and their access patterns on our web properties.

30.14 14. Google Play Store

Downloads from the Google Play store are compiled into aggregated usage statistics on our management portal.

Crash reports you elect to send are provided to us as anonymous crash reports. By design, these do not contain survey field values or other device- or user- specific data.

30.15 15. Other Resources

Towards a Secure Framework for mHealth. A Case Study in Mobile Data Collection Systems. Samson Hussien Gejibo. Ph.D. Dissertation at the University of Bergen, 2015.

CHAPTER

THIRTYONE

VERIFYING DOWNLOADS

The downloads page now has a link to a file listing the SHA256 hashes for all download files.

The long hash strings in that file can be used to confirm that the file you download is not truncated, incomplete or altered.

```
sha256 hashes for all download files

if a signature for a download is not in this file please let us know!

24cde69f4474e4c21469f8a85cd4fd7e39d06445df67dcb4440f0ec8574aeb24 "BreathCounter v1.1.apk"
d8f63b629a66297d2cf66394491838b771de24f2ed74fab0026f9ab4a919e593 "ODK Aggregate v0.9.8.1.zip"
4ab544a17f8d4b33cb11f8ea025f9461332b0c85181dfc5ebbdd8e46acf1957d "ODK Aggregate v1.0.4 linux-installer.run"
27f4d4678a1e63adb5bb5651264497bf2becd802bd159721259dfa75e41fd883 "ODK Aggregate v1.0.4 linux-x64-installer.run"
804cc899bab70c1bf014561aaf0de0838039645324f60816e741cd672110cafc "ODK Aggregate v1.0.4 osx-installer.app.zip"
72db368caa8f7e654e9cb2667dad5923c6bf20192aa4ea1a7adb5f887f4fc76 "ODK Aggregate v1.0.4 windows-installer.exe"
685c60cf8de45bfa3a2e4d0d95fab9952e7d08cbc5c1325f815e4946f9c5686c "ODK Aggregate v1.1.0 linux-installer.run"
93938e0b063bec0e2e41f50e75bf45c945a417f19cd99a5264448e8bad524f96 "ODK Aggregate v1.1.0 linux-x64-installer.run"
1ab7aa8a6ec74c986e981c133b2dbe43229f711b65e2a11f5fc7fe2bf6ab28d4 "ODK Aggregate v1.1.0 osx-installer.app.zip"
1c439c16b28663487c454e04be6df2fa7bf79c95417bba6894167fb6e4cda713 "ODK Aggregate v1.1.0 windows-installer.exe"
1c7ccb42a61ec61f179ba949321fff9cb5f0ed0b7814d53dbf3458fb7737e2af "ODK Aggregate v1.2.0 linux-installer.run"
408a87e98c86629b600713f355afd4344e53e406c91645f7e61d1ae42a07dda9 "ODK Aggregate v1.2.0 linux-x64-installer.run"
4cd598894ea2e47d8039418a3115405fbb5abe2bbcbf06b68b23e620a5104788 "ODK Aggregate v1.2.0 osx-installer.app.zip"
638839049ed1ebcdb34f8faab2a332f0a99b771a110ce23b52028135489f4bae "ODK Aggregate v1.2.0 windows-installer.exe"
e9fba0fe072671b184c5139c5829a2f8494f9e61e365756c7575ac0320ecdfdf "ODK Aggregate v1.3.0 linux-installer.run"
705aefab8b22874d341501695e04d57c925d7a7acd86db0e90901504da7ecbf7 "ODK Aggregate v1.3.0 linux-x64-installer.run"
2f690b9df4f8219aa1edb9467617717f03b5eee58a9bffec43c25242e024eaed "ODK Aggregate v1.3.0 osx-installer.app.zip"
dcb38efd6ebda0d2af35afad1b6816e34c225b5692b5453523c4da8df6fcfa28 "ODK Aggregate v1.3.0 windows-installer.exe"
811642d4d1f32bb7708f3d32ce4fd5642d3603f11fa008cbea3ce5540400e1ec "ODK Aggregate v1.3.4 linux-installer.run"
cd90a57579657f33adbf41206d12da270185625cf3d44b56eac4973ff3d1ea58 "ODK Aggregate v1.3.4 linux-x64-installer.run"
254cc676da589ea18630c7af992d38641bad6bb200f2caea5a0e10fd0274db29 "ODK Aggregate v1.3.4 osx-installer.app.zip"
2ad9d80f606d9ab473cf5659a4e83dc73527dc38b3e9c48ddd4e6fab5cba4049 "ODK Aggregate v1.3.4 windows-installer.exe"
a8e2683786c4eefafeae4b152c4e5e3773be61411dac604c5ba3a4d4aee3112e "ODK Aggregate v1.4.10 linux-installer.run"
4548c398cea143444ed4a1343d4ff4c943741e6d5b368b4857b11f46b655a90c "ODK Aggregate v1.4.10 linux-x64-installer.run"
475e2e3faa0095359384d3659277b21fdbc769625670c32a8ba1bf291da74db7 "ODK Aggregate v1.4.10 osx-installer.app.zip"
9488d52992d22bd8f0caa7e46390b6b6bc16f79b3ad85fe74372217337e8a239 "ODK Aggregate v1.4.10 windows-installer.exe"
21436216819a3b2f30fc861e28bf98089a4e9a239675e64532e697044d88b55d "ODK Aggregate v1.4.11 linux-installer.run"
474b488fcaeceea9fa5963bbfeee21b08f304a3c48e143ca1c103d807e5f5b35 "ODK Aggregate v1.4.11 linux-x64-installer.run"
be6c0daa7f244dc40ee16e4af5fd03f70aee212629d1fd92ed19f34893cca146 "ODK Aggregate v1.4.11 osx-installer.app.zip"
56b88590d3fde7dcd0e6744cc2f77036327a8f8791da01f68e17ba69215e3c87 "ODK Aggregate v1.4.11 windows-installer.exe"
4f2c3a786c13bb43b720446ce76d197cf67fe77ee55890d4d9f492c9956bce13 "ODK Aggregate v1.4.12 linux-installer.run"
4f2c3a786c13bb43b720446ce76d197cf67fe77ee55890d4d9f492c9956bce13 "ODK Aggregate v1.4.12 linux-x64-installer.run"
979c2ee8a4d14036bccc8042a8177e52bd83d652936ab7556c3fd5781c38edfa "ODK Aggregate v1.4.12 osx-installer.app.zip"
8857f11ea4423094f9b843b4c9a33a6fc74abfff00e4845b3f02e75427dd191d "ODK Aggregate v1.4.12 windows-installer.exe"
20450b7546ac1cbe3c5a8887488a5d802aedfee941e5323769b01d5fe381e439 "ODK Aggregate v1.4.13 linux-installer.run"
cbff6e5eb0b9dad0c50b1faa73cfdbe1b52cf3285568a421b5e640aa1859d6dd "ODK Aggregate v1.4.13 linux-x64-installer.run"
75b195ab997df632a9e5786d4a1b3afa1753fd665a709b2fcd729ca43a2a3f65 "ODK Aggregate v1.4.13 osx-installer.app.zip"
eb1d99683eed6e03608db688d9d97b3d7b0c60c94ca90fa8bd9b5e65c1bd04e4 "ODK Aggregate v1.4.13 windows-installer.exe"
693c3fa52a9af67c17ebe0220818692c99b3da9a01f78651e5262fe5a01b3414 "ODK Aggregate v1.4.14 linux-installer.run"
```

31.1 Verifying on Linux and Mac

- Open a terminal window.
- Navigate to the folder containing the file you downloaded. (cd ”path/to/file”)

- Then type **`sha256sum "filename"`** on the terminal.

```
$ cd path/to/file
$ sha256sum "filename"
```

31.2 Verifying on Windows

- Open a cmd window.
- Navigate to the folder containing the file you downloaded (cd ”path/to/file”).
- Then type **`certUtil -hashfile "filename" SHA256`** on your command prompt.

```
> cd path/to/file
> certUtil -hashfile "filename" SHA256
```

If the downloaded file is not corrupted, the long hash string printed by these tools will match the one listed for that file in the link at the top of our downloads page.

To check this :-

- Copy the SHA256 hash displayed on your terminal or cmd.

- Then open the link which contains the list of all SHA256 hashes for the downloaded files.
- Press Ctrl+F in the window and paste the copied signature in the box which opens on the top right corner. You will get a single result highlighted if the signature is found.

Fig. 31.1: Match for hash found

Tip:

Fig. 31.2: No match found for the hash

- In Linux terminal, the name of the downloaded file is also printed along with the hash. Be careful not to copy it.
- When you get a match, also check that the name of the file in the list aside the hash is the same as that of your downloaded file.

CHAPTER

THIRTYTWO

FAQ

32.1 What is ODK?

Open Data Kit (ODK) is a free and open source set of tools which help organizations create mobile data collection solutions.

Our blog and deployments page have good examples of what others have used ODK for. Our research page also has videos and slides that explain ODK.

32.2 How do I use ODK?

Please read the *Getting Started Guide* section of our documentation to understand the initial instructions.

Please read through the documentation on our implementer instructions and on the developer wiki.

If that doesn't help, search the ODK forum to see if the issue has been answered.

If you don't find an answer after searching, then post your question to the appropriate category.

32.3 What kinds of questions do you answer on the mailing list?

Our team only answers questions about the supported tools. We generally do not answer form design questions, and instead, refer you to the form design guide. You are more than welcome to post any message – someone else in the community is likely to help.

32.4 An ODK tool isn't working the way I expected. What do I do?

Make sure you are running the latest versions of ODK software. You can find them on our downloads page.

Search the issue tracker to see if the problem you are having has been reported. If you can't find it, report it there.

Please be precise about the tool you are using, the version of the software, and include all the steps you did so we can reproduce the same problem. If you can get a stack trace, please attach it. If the problem is with a form, please attach it as well. If you are new to filing bug reports, read How to Report Bugs Effectively.

If your issue is urgent, we recommend you hire help from one of the ODK implementations companies.

32.5 How do I get a stack trace or log after an Android "Force Close"?

See Collect Troubleshooting. Capturing an error log is more difficult now that the newer Android operating systems restrict access to the system log stream. Once you begin capturing a log and have reproduced the crash, please open an issue here and attach the captured log file.

32.6 How do I request a new feature or enhancement?

Search the issue tracker to see if the feature you want has been suggested. If you find it, vote for it by adding yourself to receive notifications. If you don't find it, open a new issue (feature request) describing the scenario which you need the feature for.

32.7 How do I customize or implement an ODK solution?

The core team does not provide code or implementation support beyond what is described on our implementer site and on the developer site. If you need more support, hire help from one of the ODK implementations companies.

32.8 How should I cite Open Data Kit in a publication?

People often ask "How do I cite ODK?". Since Open Data Kit is an academic research project, please cite the academic research papers corresponding to the appropriate tool version.

If you are using ODK tools with a version code 1.x cite: Open Data Kit: Tools to Build Information Services for Developing Regions Carl Hartung, Yaw Anokwa, Waylon Brunette, Adam Lerer, Clint Tseng, Gaetano Borriello In ICTD, 2010. http://dl.acm.org/citation.cfm?id=2369236

If you are using ODK tools with a version code 2.x cite: *Open Data Kit 2.0: Expanding and Refining Information Services for Developing Regions <http://www.hotmobile.org/2013/papers/full/2.pdf>_* Waylon Brunette, Mitchell Sundt, Nicola Dell, Rohit Chaudhri, Nathan Breit, Gaetano Borriello In HotMobile, 2013. http://dl.acm.org/citation.cfm?id=2444790

Other publications to cite for individual ODK tools are available on the Research page.

32.9 Are there other data collection systems I should consider?

Of course! There are many other data collection systems that might work better for you. Some of them use ODK in some way:

- Ona;
- Survey CTO;
- KoBo Toolbox;
- Commcare HQ;
- DoForms;
- DataWinners;
- ViewWorld;
- PhiCollect;

While other are ODK compatible:

- JavaRosa;
- OpenXData;
- RapidSMS;

If you want to find out more, MobileActive is a great place to learn more about data collection. Mobile Data Collection Tools - Comparison Matrix and Mobile-Phone-Based Data

Collection Systems Comparison Table, Mobile and Web Technologies for Social and Economic Development report, Comparing Mobile Solutions for GIS Data Collection and Display, and Nomad Mobile Collection Systems Decision Tool are also good resources.

We also have peer-reviewed research and user stories that describe the situations where ODK is likely to be easier to use, less error-prone, more cost-effective and more timely when compared to other data collection systems.

32.10 Should I email members of the ODK team directly?

Probably not. If you have a private question you cannot post to the list, please send it to contact@opendatakit.org.

32.11 What Android phone/tablet/device should I use?

The Android ecosystem changes too rapidly to recommend one device. In general, we recommend you get devices that you can source in-country and run the latest Android OS (even though we support Android OS 1.6+ and higher). ODK Collect will run on most Android form factors (including tablets and netbooks).

We recommend you spend a little more to get a higher quality device, instead of buying the cheapest phone. If you need a supplier, try:

- Amazon;
- Newegg;
- Ebay;
- N1 Wireless;

For a list of all Android devices, make a search in:

- Wikipedia;
- GSMArena;
- Phone Scoop;

32.12 Why are my messages on the list being delayed?

If you are a first-time poster (or are not subscribed) to the lists, your emails are moderated. This process can take a few hours, so no need to send multiple messages.

32.13 Do you support XForms?

Not exactly. We support the OpenROSA 1.0 subset of XForms described at https://bitbucket.org/javarosa/javarosa/wiki/xform-jr-compat.

32.14 Do you have an API?

Yes. We support the OpenROSA 1.0 API as described at https://bitbucket.org/javarosa/javarosa/wiki/OpenRosaAPI.

32.15 How do I change the language that Collect is using?

The tools understand the text internationalization features of javarosa XForms and can restructure your input so that the question text is grouped into an <itext/> translation block for internationalization. Therefore, to support multiple languages you need to specify the question text in the appropriate language using the tag. To change the language the XForm's questions are being rendered in, simply click the 'Menu' button on the Android while filling out a form and click the 'Change Language' button.

To change the language ODK Collect (version 1.2 and higher) is using to render the user interface (e.g., button text, instructions) you need to change the phone's language settings. Collect determines its user interface language (not question language) based on the phone's overall settings. For example, in the phone settings, if you set the phone's locale to Espanol, Collect will render its navigation text in Spanish.

CHAPTER

THIRTYTHREE

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.

Form Uploader Part of ODK.

A tool for uploading blank forms and their media files to ODK Aggregate.

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.