ODK Integration to SMART Connect
Design Document
DRAFT

March 26 2015

Refractions Research

This report is made possible by the generous support of the American people through the United States Agency for International Development (USAID) under the terms of Cooperative Agreement No. RLA-A-00-07-00043-00. The contents are the responsibility of the Africa Biodiversity Collaborative Group (ABCG). Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of USAID or the United States Government. This publication was produced by Wildlife Conservation Society on behalf of ABCG.

AFRICA BIODIVERSITY COLLABORATIVE GROUP

Overview

ODK Tools Overview
The following is a simplified list of ODK tools that we expect users to employ when collecting data to store in SMART using SMART Connect:

- **ODK Aggregate** - The main ODK server-based tool that runs in a web server. It publishes blank forms to mobile devices, accepts completed forms from the devices and stores all of the completed data. The current version supports both ODK 1.0 and ODK 2.0 tool sets. **Current Version: 1.4.5**

- **ODK Collect** - The standard data collection tool that runs on Android devices. This is an ODK 1.0 data tool that works with Aggregate 1.4.5 and other ODK 1.0 paradigm tools. **Current Version: 1.4.5**

- **ODK Survey** - The new data collection tool in the ODK 2.0 paradigm. It will work with ODK Aggregate 1.4.5 and other ODK 2.0 compatible tools.
Current Version: Revision 126, Beta

- **ODK Briefcase** - This is a tool similar to ODK Aggregate that is designed to run on a desktop machine instead of a web server. It can compile data directly from ODK Collect, or from an ODK Aggregate server. It only work inside of the ODK 1.0 paradigm - there is no analogue for this tool in ODK 2.0. **Current Version: 1.4.5**

ODK Targeted Version
We will target ODK 1.4.5 tools with the development of SMART Connect because that is the latest stable version of all the tools we need to make the integration successful.

We will keep the ODK 2.0 paradigm in mind as we are building SMART Connect to minimize the amount of work needed to support 2.0 tools. The actual data collection tools of ODK Collect and ODK Survey are based on completely different form formats, so it will be necessary to build two export code bases to support both of these formats eventually; however writing one will be most of the work and we can re-use much of the code to write the second when we need support for it.

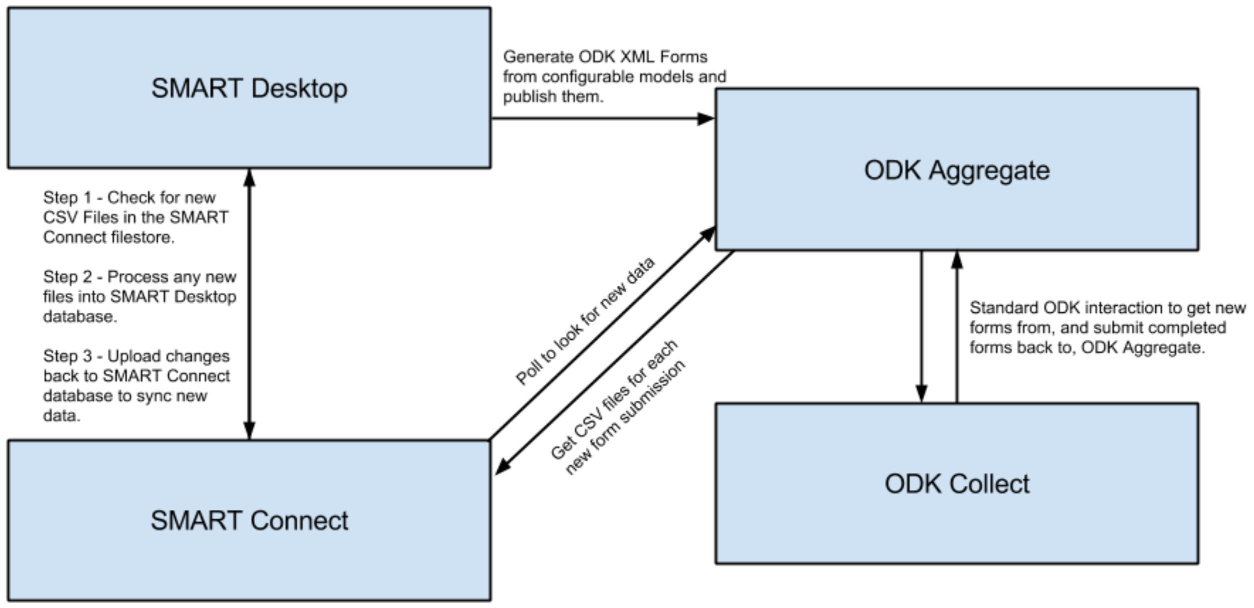
The main issue with supporting any ODK 2.0 tools right now, other than Aggregate 1.4.5, which we will use, is that the software is not at a stable release yet, and the documentation is not complete yet so it is very difficult to predict the changes that could occur before the tools stabilize.

ODK Integration Overview

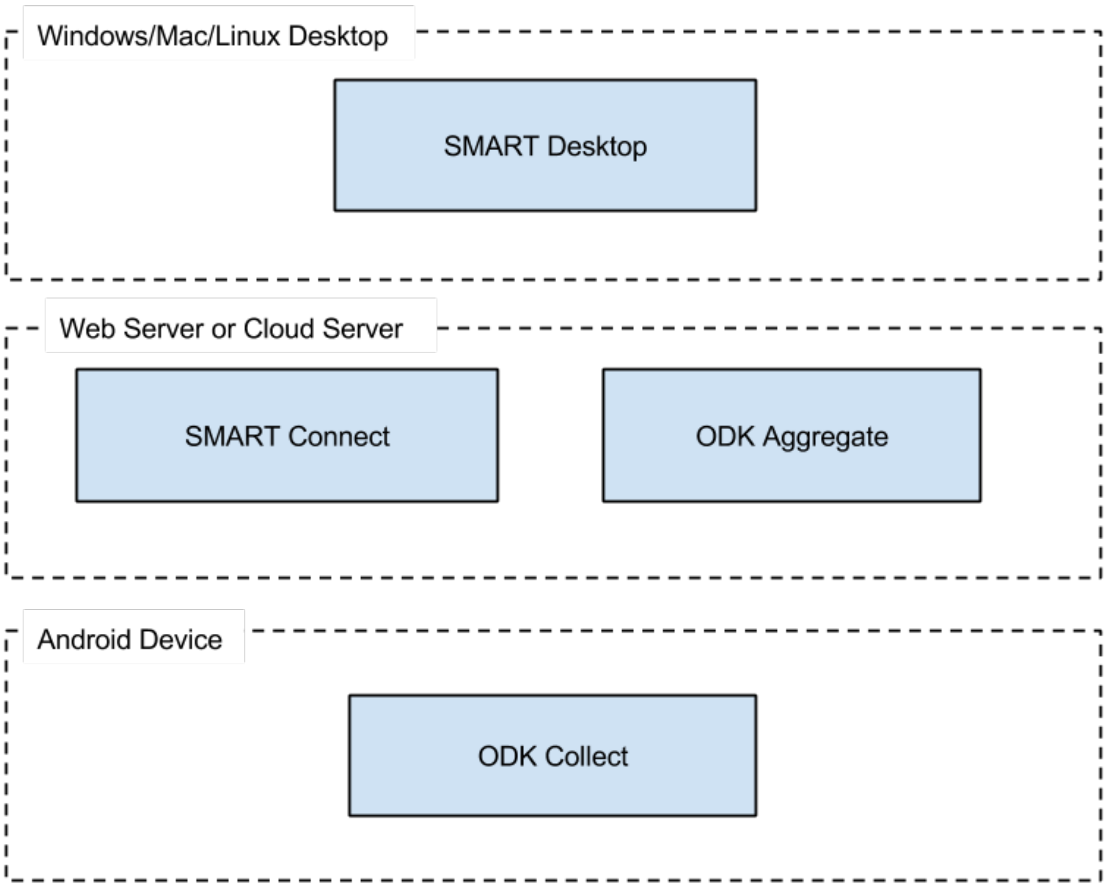
This document describes the design for allowing SMART to generate ODK forms which can be used to collect data and have that data automatically processed back into a SMART database and data model. This will allow ODK to be an additional data collection option to SMART, along with the current options of CyberTracker and manual data entry.

The basic concept will be for SMART Desktop to automatically generate an ODK XML form based on a selected configurable model (or complete data model) and upload the form to ODK Aggregate. From there, the form can then be used in ODK Collect. Once data is collected, the data upload process will also use ODK Aggregate to collect and disseminate the data.

SMART Connect will be designed to integrate with ODK Aggregate so that it can process data into a SMART database as new forms are completed and submitted to ODK Aggregate.


SMART Desktop
Generate ODK XML Forms from configurable models and publish them.
ODK Aggregate
Step 1 - Check for new CSV Files in the SMART Connect filestore.
Step 2 - Process any new files into SMART Desktop database.
Step 3 - Upload changes back to SMART Connect database to sync new data.
Poll to look for new data
Get CSV files for each new form submission
Standard ODK interaction to get new forms from, and submit completed forms back to, ODK Aggregate.
SMART Connect
ODK Collect

These components will be deployed on desktop, web-based and mobile platforms as shown below:



Limitations

- There will not be any support for using existing ODK forms or any forms not generated by SMART.
- Only forms generated in a particular Conservation Area, e.g. “CA XYZ”, can be used to load data back into “CA XYZ”. Configurable models and data models are unique to Conservation Areas and therefore a form generated by a certain CA will not process data correctly into any other CAs.
- Once the form is generated by SMART, any modification of that form could possibly invalidate its compatibility with SMART. Users can, in theory, edit the text labels and other surface attributes of a form, but any purposeful or accidental renaming of fields or values will invalidate the form or, at the very least, some data could be lost/dropped if the names no longer match to SMART data model names.
- ODK Survey is not compatible with ODK Collect forms. SMART will have to generate two separate formats to support both tools.

- As discussed in the section titled “ODK GPS Tracks” below, there is not an easy way to replicate a GPS track log with points every 5 or 10 seconds using ODK forms. Existing ODK users must be OK with this limitation now, or use a work-around. Any work around could most likely be adopted within the SMART Connect ODK integration.

Generating ODK Forms from SMART

There are two formats to create when generating forms from SMART into the two main ODK collection tools, “ODK Collect” and “ODK Survey”:

1. ODK XML format - used by “ODK Collect”
2. JSON Form Definition - used by “ODK Survey”

SMART could generate forms in one or both of these formats to support the applicable ODK tools. These exports will correspond to a selected Configurable Model stored in SMART, much like the existing CyberTracker Integration.

ODK Form XML Format (ODK Collect)

An example of a very simple ODK form is shown below in its XML format:

```
<h:html xmlns="http://www.w3.org/2002/xforms" xmlns:h="http://www.w3.org/1999/xhtml" xmlns:ev="http://www.w3.org/2001/xml-
events" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:jr="http://openrosa.org/javarosa">
  <h:head>
    <h:title>Form with Groups and Ids</h:title>
    <model>
      <instance>
        <data id="build_Form-with-Groups-and-Ids_1426196807">
          <meta>
            <instanceID/>
          </meta>
          <Employees/>
          <observation jr:template="">
            <cat1/>
            <people>
              <actiontakenpeople/>
            </people>
            <animals>
              <numberOfAnimals/>
              <photos jr:template="">
                <Picture/>
              </photos>
            </animals>
          </observation>
        </data>
      </instance>
      <itext>
        <translation lang="eng">
          <text id="/data/Employees:label">
            <value>Enter Employee Name</value>
          </text>
          <text id="/data/Employees:hint">
            <value></value>
          </text>
          <text id="/data/observation:label">
            <value>New Observation</value>
          </text>
          <text id="/data/observation/cat1:label">
            <value>Select Category</value>
          </text>
          <text id="/data/observation/cat1:option0">
            <value>People - Direct Observation</value>
          </text>
          <text id="/data/observation/cat1:option1">
            <value>Animal Sighting</value>
          </text>
          <text id="/data/observation/people:label">
            <value>People - Direct Observation</value>
          </text>
          <text id="/data/observation/people/actiontakenpeople:label">
            <value>Action Taken People</value>
          </text>
          <text id="/data/observation/people/actiontakenpeople:option0">
            <value>Observed Only</value>
          </text>
          <text id="/data/observation/people/actiontakenpeople:option1">
            <value>Arrested</value>
          </text>
          <text id="/data/observation/animals:label">
            <value>Animal Sighted</value>
          </text>
          <text id="/data/observation/animals/photos:label">
            <value></value>
          </text>
          <text id="/data/observation/animals/photos/Picture:label">
            <value>Photo the Sighting</value>
          </text>
        </translation>
      </itext>
      <bind nodeset="/data/meta/instanceID" type="string" readonly="true()" calculate="concat('uuid:', uuid())"/>
      <bind nodeset="/data/Employees" type="string" required="true()"/>
      <bind nodeset="/data/observation/cat1" type="select"/>
      <bind nodeset="/data/observation/people" relevant="(selected(/data/observation/cat1, 'people'))"/>
      <bind nodeset="/data/observation/people/actiontakenpeople" type="select" relevant="(selected(/data/observation/cat1,
'people'))"/>
      <bind nodeset="/data/observation/animals" relevant="(selected(/data/observation/cat1, 'animals'))"/>
      <bind nodeset="/data/observation/animals/numberOfAnimals" type="int" relevant="(selected(/data/observation/cat1,
'animals'))"/>
      <bind nodeset="/data/observation/animals/photos/Picture" type="binary" relevant="(selected(/data/observation/cat1,
'animals'))"/>
    </model>
  </h:head>
  <h:body>
    <input ref="/data/Employees">
      <label ref="jr:itext('/data/Employees:label')"/>
      <hint ref="jr:itext('/data/Employees:hint')"/>
    </input>
    <group>
      <label ref="jr:itext('/data/observation:label')"/>
      <repeat nodeset="/data/observation">
        <select ref="/data/observation/cat1">
          <label ref="jr:itext('/data/observation/cat1:label')"/>
          <item>
            <label ref="jr:itext('/data/observation/cat1:option0')"/>
            <value>people</value>
          </item>
          <item>
            <label ref="jr:itext('/data/observation/cat1:option1')"/>
            <value>animals</value>
          </item>
        </select>
        <group>
          <label ref="jr:itext('/data/observation/people:label')"/>
          <select ref="/data/observation/people/actiontakenpeople">
            <label ref="jr:itext('/data/observation/people/actiontakenpeople:label')"/>
            <item>
              <label ref="jr:itext('/data/observation/people/actiontakenpeople:option0')"/>
              <value>observedonly</value>
            </item>
            <item>
              <label ref="jr:itext('/data/observation/people/actiontakenpeople:option1')"/>
              <value>arrested</value>
            </item>
          </select>
        </group>
        <group>
          <label ref="jr:itext('/data/observation/animals:label')"/>
          <input ref="/data/observation/animals/numberOfAnimals">
          </input>
          <group>
            <label ref="jr:itext('/data/observation/animals/photos:label')"/>
            <repeat nodeset="/data/observation/animals/photos">
              <upload ref="/data/observation/animals/photos/Picture" mediatype="image/*">
                <label ref="jr:itext('/data/observation/animals/photos/Picture:label')"/>
              </upload>
            </repeat>
          </group>
        </group>
      </repeat>
    </group>
  </h:body>
</h:html>
```

Groups & Looping

The <group> tag allows organization of individual form questions/widgets into groups. These groups can then be looped which will cause the Mobile device to ask the user if they wish to repeat the group until they select the ‘no’ option. This can be used for our “New Observation” option so that users will be prompted to enter a new observation each time they complete data entry for a single observation.

Relevance

The “relevant” attribute within a <bind> tag allows the form creator to define whether or not to show a particular question or group to the user. Using these tags we can build our category trees and attribute trees and only show users questions that are relevant to the category they select, etc. The example above shows a simple category selection “<select ref="/data/observation/cat1">” with two options (animals or people) and then only the attribute questions for either animals or people are shown next depending on the user selection of “cat1”.

Images

The ODK XML format also supports image collection. The following snippet from the example above shows an image collection that asks the user to keep adding photos until they are finished and do not want to add any more photos, using the same looping as before:

```
<repeat nodeset="/data/observation/animals/photos">
  <upload ref="/data/observation/animals/photos/Picture" mediatype="image/*">
    <label ref="jr:itext('/data/observation/animals/photos/Picture:label')"/>
  </upload>
</repeat>
```

Other media such as audio and video recordings are also supported in ODK and could be added at a later time if SMART configurable models add support for those media types.

Required Fields and Data Restrictions

ODK forms support the idea of a “required” field, where the user is not allowed to proceed to the next question until the field is filled out:

```
<bind nodeset="/data/Employees" type="string" required="true()"/>
```

ODK forms also support data restrictions. A few examples are:

- minimum and maximum on numeric fields
- minimum length and maximum length on text fields
- minimum and maximum on date fields
- custom constraints on all field types

This will allow SMART to generate forms that implement the data restrictions we have in our SMART data model directly in the user's data-collection device.

ODK Form Limitations

Special characters are not allowed in the Data Name ODK field.

For each individual question in an ODK form there are 4 main fields:

- Data Name - the name of the form variable
- Label - the text displayed to the user
- Hint - a text hint for the user
- Value - filled out by the user unless a "select" widget is used, in which case the form creator defines what value each selection represents.

The Data Name text can only contain characters and numbers. One simply needs to be aware that no special characters can be used in automated names for this field. All of SMART's category and tree keys have periods in them so these must always go in the value fields or otherwise be removed from the data name field.

Internationalization

ODK Collect also supports internationalization such that we can provide language alternatives in a single form export. Text labels can reference an id and ids can have different values per language, e.g.:

```
<label>section c</label>
<input ref="language">
        <label ref="jr:itext('hello')"/>
        <hint ref="jr:itext('hint')"/>
</input>
```

where the label is defined in the data model section as:

```
<translation lang="english">
      <text id="hello">
        <value>i speak english</value>
      </text>
      <text id="hint">
        <value>menu button to change languages</value>
      </text>
    </translation>
    <translation lang="italian">
      <text id="hello">
        <value>ci&#xE0;o</value>
      </text>
      <text id="hint">
```

```
        <value>"hello" in italian</value>
      </text>
</translation>
```

Users can then use the built-in ODK menu to switch between all supported languages anytime they wish. We may need to export only a single language from SMART to the ODK form if exporting many languages into the same file makes the file size too large and starts to affect performance, but this will be tested once we have a full data model export to test.

ODK GPS Tracks

There does not appear to be an easy way to automatically gather GPS points on a scheduled basis in ODK. There is a GPS location widget that captures a GPS point, but a user is required to interact with the form each time this is done. It is likely that when we load data into SMART we will only have all the observation points and can use those to generate a track, but it may not be very accurate if the data collectors went long distances in-between observations.

Data collectors also have the existing options in SMART to upload a track if they have a standard GPS device, or another Android application collecting their track data separately from ODK.

Organizations could also train their data collection teams to mark a blank observation in the ODK form every few minutes or so, to increase the number of track points recorded.

At a minimum we should have an option that allows users to select something like “Take Position Point” instead of recording an observation. This would simply show the GPS collection page and allow users to manually record points as often as they like.

JSON Form Definition (ODK Survey)

Below is an example of the format:
https://drive.google.com/file/d/0ByqUEY8Cd1BIQ0djS3VaWWF6YkE/view?usp=sharing

This format is typically produced through the use of an ODK tool, “XLSXConverter”, https://opendatakit.org/assets/beta-4/xlsxconverter/index.html

To automate this process SMART could produce the JSON format directly so that users do not need to convert and save their files. This format is not documented very well, but basically supports all of the same features as the ODK XML format, along with additional features listed here:
https://opendatakit.org/use/2_0_tools/odk-application-designer-2-0-rev126/

This ODK 2.0 tool is in Alpha at the time of writing, and it is entirely possible that this format will evolve going forward. We suggest that support for this format not be implemented now and we can consider adding support in the future if there is user-demand for features in ODK Survey that are not in ODK Collect. The ODK 2.0 tools will likely have an official release by the time that occurs as well.

Publishing an ODK Form

SMART will support two ways of publishing ODK forms so that users can use them in their data collection tools:

1. Push the form to ODK Aggregate
2. Manually download the XML and copy it to a mobile device

Push Form to ODK Aggregate Server

A typical SMART Connect instance with ODK Support will include a configured ODK Aggregate Server as part of the installed software on the SMART Connect Server (a user could also use an existing ODK Aggregate running on a different computer). Each CA can have its own ODK Aggregate Server, or they can all use the same one - either option will work.

In this case, when a user generates a new ODK form from a configurable model (or data model), they will be asked if they wish to publish the ODK form to the Aggregate Server for users to start data collection. This publishing method will be completed using the “formUpload” function of the ODK Aggregate API, documented here:
https://code.google.com/p/opendatakit/wiki/BriefcaseAggregateAPI

Using this API, the new, blank form will be published to the ODK Aggregate Server. Data collectors who have ODK tools like ODK Collect on mobile devices can have them configured with the ODK Aggregate Server URL. This will allow data collectors to find and download any new blank forms as soon as they are published to the ODK Aggregate Server. There do not appear to be any options for the mobile devices to be actively notified of new forms; they must use the “Get Blank Form” option in ODK Collect.

At this stage, all of the user interaction will follow existing ODK practices for data collection and submission back to the Aggregate Server.

Manually Installing a SMART-generated ODK Form

When someone uses SMART desktop to generate an ODK form, he/she will also have the option of saving that form to a local file instead of publishing it to ODK Aggregate. This will mainly be used when users are creating new forms and wish to test them out before officially publishing them out to production.

The SMART software will simply ask the user for a location to save the XML file to, and export the selected configurable model to an ODK form file in that location. The user can then copy the form directly to their mobile device as directed by the ODK documentation.

For ODK Collect, users simply copy the XML form into the following directory on their android device:
\<android device>\Internal storage\odk\forms\

For ODK Survey, the directory is slightly different:
\<android device>\Internal storage\opendatakit\survey\

Importing Completed Forms into SMART

Importing ODK form data into a SMART database is a two step process:

1. Getting raw data from ODK into CSV format and copying it onto the SMART Connect server
2. Processing the CSV data and loading it into the SMART Database

Accessing CSV Data from ODK

There are three potential ways of retrieving data collected by the ODK tool "ODK Collect", all of which will produce a CSV format that can then be processed by SMART:

1. GUI Briefcase tool - connects to ODK Collect directly
2. ODK Briefcase CLI (Command Line Interface) - connects to Aggregate using Briefcase and gets data from the Aggregate database
3. Aggregate API

These methods are described below.

1 - GUI Briefcase tool

With this method, the user would use the Briefcase UI to connect directly to their Android ODK Collect instance and download data. ODK Aggregate is not used.
The process to get data into Briefcase is somewhat laborious and not conducive to automation. Therefore, this will not be used as the standard approach, but it may be useful for some users to use this process manually for testing purposes. From ODK Briefcase, users can manually export CSV files as they wish.

Details on the data transfer are shown under the headings, "Pulling data directly from an Android 2.x device" and "Pulling data directly from an Android 4.x device" on this web page: https://opendatakit.org/use/briefcase/

2 - ODK Briefcase CLI (Command Line Interface)

With this method, users would use the "Upload" (to ODK Aggregate) option provided by the Android ODK Collect software. Once uploaded to ODK Aggregate, the command line interface inside ODK Briefcase would allow for SMART Connect to automate the pulling of data from

Aggregate and producing CSV files from this data (for loading into SMART). This option requires ODK Briefcase. ODK Briefcase is not scheduled for any support in ODK 2.0, therefore it would only be good for short term use. We do not plan to implement this option.

3 - Aggregate API

With this method, users would use the “Upload” (to ODK Aggregate) option provided by the Android ODK Collect software, and then SMART Connect would use an API provided by ODK Aggregate to request form data (for processing and loading into a SMART database). It is documented here:
https://code.google.com/p/opendatakit/wiki/BriefcaseAggregateAPI
This API appears to be supported in both 1.0 and 2.0 compatible versions (1.4.5) of ODK Aggregate, and therefore is most likely to continue to be useful as ODK continues to evolve. We recommend using this API as the method for pulling data out of ODK and into a CSV format.

SMART Connect will be configured to check for data from its associated ODK Aggregate Server on a regular basis. This interval is user-defined and discussed in later sections of this document. This code will be run using a Cron job or similar process to automatically run without any user interaction. Each time that the data check is initiated, SMART Connect Java code running on the server will perform the following processes:

- Use OpenRosa Form Discovery API to get a list of all form definitions; each form definition will match to a single configurable model in SMART.
- For each form definition, get a list of all submitted, completed forms using the “view/submissionList” API call.
- Compare the list from the previous step with a stored list of completed forms that have already been processed. Download each of the new ones using the “view/downloadSubmission” API call.
- Add the freshly downloaded forms to our stored list of completed form data that we have already retrieved.

This is the planned method that SMART Connect will use for automated data processing of ODK data, because it will be the most consistent through newer versions of the software and it is the most stable, programmatic method.

Frequency of Data Access from ODK

There are a few methods by which ODK Aggregate can enable live publishing of data; they are listed here: https://opendatakit.org/use/aggregate/data-transfer/#Publishing

Most of these methods are targeted to direct human interactions, not programmatic access, therefore are not as suitable for our purposes of integrating with SMART as the above APIs are. The ones that are programmatic are generally not stable releases yet, and are as standards-based as the Aggregate API is.

Given that live publishing of data is not suitable, SMART Connect will have to poll for new data from ODK at regular intervals. Since the ODK Aggregate server should normally be residing on the same server as SMART Connect, it is unlikely that polling, even at fairly high frequency (every minute) will cause any issues. The frequency of polling should be user-configurable.

SMART Connect will have to track form IDs to know which completed form data have already been downloaded and processed, because the Aggregate API we plan to use does not contain any features that would help us track which data we have already downloaded.

Users should also have the option of triggering a data download manually from the SMART Connect Administration web pages.

Processing the CSV Data into the SMART Database

SMART will convert CSV data from ODK back into the SMART database of the associated CA for which the data was collected.

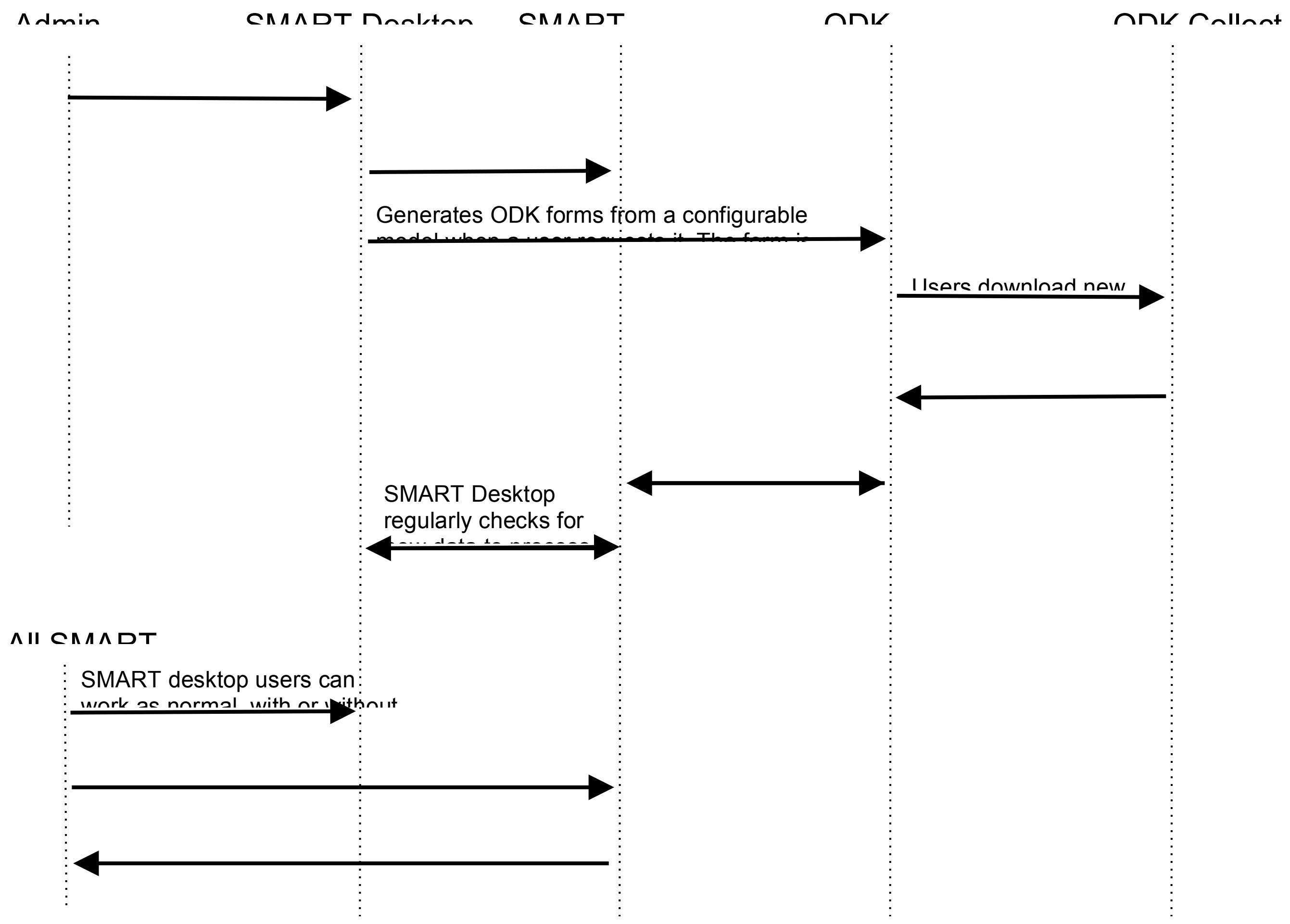
This process will consist of matching all category-related form questions and the attribute-related form questions with their selections and SMART data model keys, which will be embedded into the forms when they are generated by SMART.

Frequency of Data Access from SMART Connect to SMART

SMART will process data from SMART Connect in much the same way as SMART Connect processed the data from ODK. SMART will poll the Connect instance on a user-defined schedule (every X minutes) as well as at start-up of the desktop application depending on the configuration setup for each CA in SMART that is configured as a SMART Connect Master Database.

The details of this process will be the same for a number of data collection types and is fully described in the SMART Connect Design Document.

Data Flow Summary


Admin
SMART Desktop
SMART
ODK
ODK Collect
Generates ODK forms from a configurable
Users download new
SMART Desktop
regularly checks for
All SMART
SMART desktop users can