Administering Forms

NOTE: if you are running your server on Linux, please see the documentation about Installing An OpenMRS Server On Linux.

Form Design Process

Define/Locate Concepts

The concept dictionary is a central part of OpenMRS. Concepts can commonly be thought of as questions and possible answers which are on forms (although they have other uses too).

The first step in designing forms is working out what data needs to be collected and how the form will look on paper. From there, building forms in OpenMRS is a gradual process that involves identifying/constructing concepts within your concept dictionary corresponding to the questions and possible answers on your form.

To identify concepts

1. Review your existing paper form.
2. Log into OpenMRS.
3. Go to the concept dictionary
4. For each field on the paper form, search the Concept Dictionary to locate an existing concept.
    a. If a concept already exists, it is helpful to mark the concept ID on the paper form or keep a list of form questions and corresponding concept IDs for your future reference.
    b. If a corresponding concept does not exist, add it to the concept dictionary.

Concepts have a number of possible data types. The most common are:

- **Coded:** Choose this if the concept you are setting up is a question with another concept as an answer. This is what you use if you want the user to choose the answer on the form using a drop down list / combo box / set of option buttons or set of tick boxes. After you choose the "coded" data type, you are given an option of choosing the possible answers to the question. (If you haven't yet set up the possible answers as concepts, you have to do this first before linking them to the question)
- **Numeric:** Choose this if the concept you are setting up is a question with a numeric answer, for example, weight. (Another example commonly used in OpenMRS demos is nasal hair length 🙂 If the value could be entered as a floating number (with decimals) as an answer to the question, check the "precise" box. If the user is limited to entering whole numbers, uncheck the "precise" box.
- **N/A:** This is used if the concept you are setting up is not actually a question at all, but a possible answer.
- **Text:** Choose this if the concept you are setting up is a question with a textual answer. Note that textual answers are not so easy to analyze later.
- **Boolean:** Choose this if the concept you are setting up is a question with a yes/no answer.

When setting up a concept, you will notice there is a tick box called "is set". This is useful for grouping observations together, by putting concepts together in sets. For example, one question may be "what are the reasons for poor adherence". This can be a concept with a coded data type, and several possible reasons can be set up as answers. However, the form can also have an "other, specify" space which is a concept with a text answer.

In order to gather information correctly, you should set up a third concept "combined reasons for poor adherence" which includes both the coded concept "reasons for poor adherence" and the "other specify" concept. When the user enters text for "other specify" the system will know that this particular "other specify" is referring to the "combined reasons for poor adherence" as it is in that set.

Create Form MetaData

After creating concepts, you must now set up your form in OpenMRS. This process involves using the OpenMRS application to populate the metadata related to your form. You can create a new form or duplicate an existing form (which allows you to reuse an existing form's metadata and, more importantly, its schema).

To view all forms in the system

1. Log into OpenMRS as an administrator.
2. Select the **Administration** link in the top navigation menu.
3. Select **Manage Forms** under the **Forms** link section.

To create a new form, follow these instructions:

1. Select the **Add Form** link at the top of the page

    NOTE: currently, the form submission engine requires encounter-based forms with a specific hierarchy (contained in the Basic Form definition); therefore, you should start new forms by copying the Basic Form and adding observations to that foundation.

    To duplicate an existing form, follow these instructions:
2. Select an existing form from the **Duplicate Form** field —

Example: The Basic Form contains all of the currently required fields in the proper hierarchy to be handled properly by the FormEntry engine.

3. Click the Duplicate button.
4. Update the form metadata, including the form name.

NOTE: While editing the form schema, ensure that the Published checkbox is **unchecked**. When we are ready to make the form available through the Form Entry module, we will need to check this checkbox.

Design Schema

This process involves using the OpenMRS application to create a schema for your form. This means telling OpenMRS which concepts are on your form.

To add new concepts to a form schema:

1. Choose the **Design Schema** use case.
2. Type the desired concept into the **Find Field Elements** search box (consult Define Concept section above). The search results should automatically display in the area below the search box.
3. Press the **Enter** key. The search box and search result index (i.e. the number to the left of the search result) should now be highlighted in gold.
4. Type the search result index in the search box.
5. Press the **Enter** key.

NOTE: You can also drag n' drop the desired field element from the search result box to the form schema. However, when starting with a blank form schema it is difficult to locate the place to "drop" the field element. You can also double-click the desired field element, but this requires you to "update" the its metadata.

To move form elements within the form schema:

1. Click the concept in the form schema
2. Move the mouse to the desired location.
3. Drop the form element into the desired location in the form element tree.

To update form element metadata:

1. Right-click the desired form element in the schema.
2. Click **'Edit Field'**.
3. Select **'Edit for this form only'** to edit the form element for the current form.
4. Enter the appropriate information (in most cases you will only need to change Multi?, Field #, Field Part, Page #, Min, Max, and Required).
    - **Multi:** Indicates whether the schema should allow multiple values for a single form element (i.e. multiple answers to a single question like "What medications are you currently taking?").
    - **Field #:** The field number corresponding to the actual form element in your form. This value may be left blank.
    - **Field Part:** The sub field number/letter corresponding to the actual form element in your form (i.e. 'a' in 1a). This value may be left blank.
    - Page #: The page number where this form element appears. This value may be left blank.
    - **Min:** The minimum number of times that this form element should appear on the form. This value may be left blank. Default = 0.
    - **Max:** The maximum number of times that this form element should appear on the form. This value may be left blank. Default = 1. Use -1 to allow for an arbitrary number of values, for example, if you intend to use a repeating table in the form which gathers many observations for this field.
    - **Required:** Indicates whether the form element should be required.

To delete the form element from the schema:

1. Select the X icon next to the form element name.

Default value

- The default value field is a velocity expression that is rendered every time the form is loaded
- If using the FormEntry Module, these variables are available to you:
    - **enterer:** User object for the person filling out the form (e.g. ${enterer.userId})
    - dateEntered: Date object for the current date+time
    - **patient:** Patient object (e.g. ${patient.personName})
    - timestamp: Date format: yyyyMMdd'T'HH:mm:ss.SSSZ
    - **date:** Date format: yyyyMMdd
    - **time:** Time format: HH:mm:ss
    - **sessionId:** Current user's java session id
    - **uid:** A unique identifier for the form being filled out that will carry through to the hl7 message
    - **patientEncounters:** The current patient's past encounters. Most recent is first in the list
    - **relationships:** All relationships the person currently has (as of version 4.2 of formentry)

Form Schema Requirements

At this point, the form schema is fairly constrained by the XSLT that translates the submitted form data (within FormEntry) into HL7. So, if you are designing forms for FormEntry (for use with InfoPath), you must follow these guidelines. Initially, we thought we'd be making up a separate XSLT for every form; however, a single XSLT has gotten us much further than anticipated.

First of all, you want to start with the basic form schema. Here are some guides...

- Form schemas should have top level nodes: PATIENT, ENCOUNTER, OBS, PROBLEM LIST, and ORDERS
- Most of the children in the PATIENT section (in the starter schema) are necessary, but I think only patient.patient_id is required

- While the XSLT can handle alternate identifiers (there's an example of our MTCT-PLUS identifier in the default XSLT), InfoPath is not the desired way to add identifiers to the system. Rather, we'd encourage you to edit patient demographics through the patient administration functions of the web application.
- In the ENCOUNTER section, encounter_datetime, location_id, and provider_id are required for things to work.
- The OBS section should linked to the concept MEDICAL RECORD OBSERVATIONS (in the schema designer, you should see the concept id for MEDICAL RECORD OBSERVATIONS in parentheses after "OBS" – e.g., on the demo site, you'll see "OBS (1238)" for the OBS section.
- Within the OBS section, you may place either simple observations (with coded, date, boolean, numeric, or text datatype) or sets. Sets should contain 1 or more children that are all simple observations. The XSLT does not support sets-within-sets, so your OBS section may contain elements and elements w/ children, but should not go any deeper.
- PROBLEM LIST (if you use it), should be just like the starter schema (problem list with two children: problem added and problem resolved). If you are not collecting diagnoses on your form, you could try deleting this section in the schema, but I'd probably just leave it there and not use it on the form.
- The ORDERS section should contain only sets (no simple observations directly under ORDERS) following the design in the starter schema. Again, only one level is supported – no sets within sets. At this stage, we are converting all of these orders to observations and have not completed a path to actually generate entries in the order tables from an HL7 message (i.e., an InfoPath form submission).

Other Resources

- See Administering FormEntry