1 How to Code Modules

1.1 Exploring a Simple Module

Now that you understand the how the XML structure relates to FAIMS modules, let's explore the simple module we uploaded in the last section.

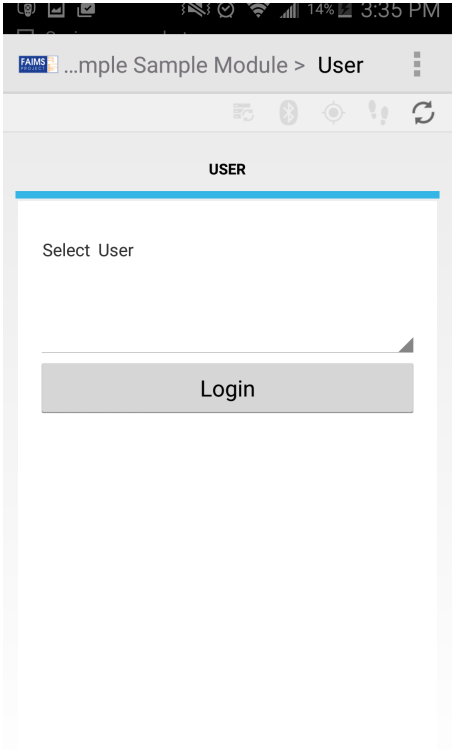

Figure 1.1 A user login screen.

To create this first login screen (Figure 1.1), we used the XML code in Codeblock 1.1.

```
<User f="nodata">
    <User>
        <Select_User t="dropdown" f="user"/>
        <Login t="button" l="Control"/>
    </User>
</User>
```

The outermost set of `<User>` tags comes right after the `<module>` tags, so they create a **tab group**. You can name these whatever you want. The only rule you should follow, for reasons we'll get into very shortly, is that it should start with an uppercase letter. The *tag name* you write is displayed in the breadcrumb navigation bar at the top of the screen.

Notice that the topmost `<User>` tag contains the *attribute* "f" with an *attribute value* of "nodata". Including the word "nodata" in the "f" attribute's value prevents the FAIMS-Tools from automatically generating unwanted code associated with the data schema. In practical terms: inputs that are provided in the "User" tab group, including the "Select User" dropdown menu, are not considered "data" that must be saved to the FAIMS database. We chose to put "nodata" here because we only want the "User" tab group to be for letting people log in.

Now that we've got our `<User>` tab group, it's time to make our actual `<User>` tab with its relevant GUI elements. We're naming this tab "User" as well, because it's where the User logs in and that seems like a good name, but we could have named it something else if we thought we'd get it confused with the tab group. Whatever tag name we choose will be displayed in the list of tabs. When you name a tab group something, its tag name must be capitalized.

The `<Select_User>` and `<Login>` elements represent GUI elements, or the actual things users will interact with when they're viewing a tab. Notice the "t" attribute, which is where we can define part of what this element looks like or does. You'll find a list of these in the FAIMS

cookbook. For `<Select_User>` we're using t="dropdown", which means this GUI element is a drop-down menu.

So how do we determine what's in that drop down menu? In this case, by creating an attribute, f="users", which in this context FAIMS understands to mean "get the list of usernames from the server and put them here."[1]

Setting t="button", as in the case of `<Login>`, creates a button you can tap. The purpose of the button is described by the text that comes after it; otherwise, it's just a button, which probably isn't going to be very useful for your module. Here we've included the code l="Control" within the tag, which causes FAIMS to link to the "Control" tab group when it is tapped. In fact, the "l" attribute works not only for buttons, but many other GUI elements as well. Note carefully that the "Control" tab group linked to by the "l" attribute's value is defined further down in the module.xml file; this code only functions because the destination is valid. If we had 1="Control" and then didn't actually have a "Control" to link to, it's safe to say this wouldn't work. Also note that references are case-sensitive, so writing "control" with a lower-case "c" would fail.

So let's say we're using our module, we've just clicked the dropdown menu and chosen our user, then we've hit the button that says "Login." If you're using the finished sample module, you'll find yourself looking at the screen in Figure 1.2.

[1] **Note from FAIMS programmer Christian Nassif-Haynes:** If you use t=dropdown, then it is possible for the user to avoid logging in (in error or intentionally) by selecting the null option and clicking the login button. If you want to prevent logging with the null user option (de facto avoiding the login), then you need to manually modify the ui_logic.bsh file after it's been generated or use t=list, following the guidelines below. Lists, unlike dropdowns, do not allow null elements so using a t="list" for login, as in the code below, prevents the problem of logging in with null user.
<User f="nodata"> <User f="noscroll"> <Select_User t="list" f="user" l="Control"/> </User> </User>

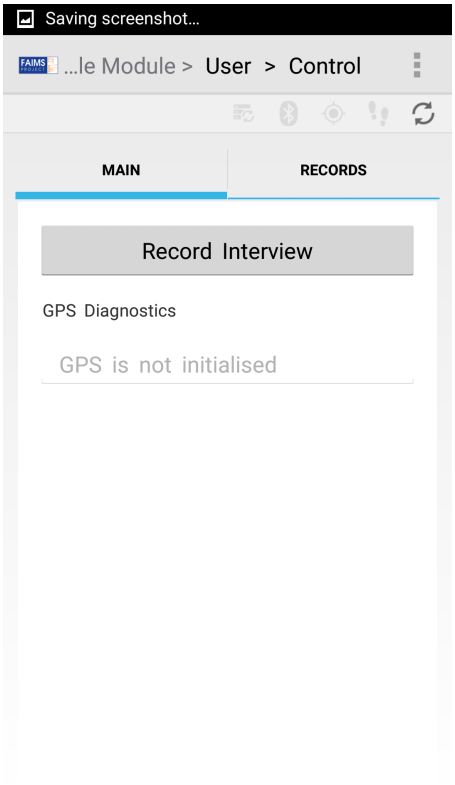

Figure 1.2 The "main" tab.

Let's take a look at the code for this tab group here. As you review Codeblock 1.2, see if you can figure out which elements are tab groups, tabs, and GUI elements.

```
<Control f="nodata">
    <Main>
        <Record_Interview t="button" l="Interview"/>
        <GPS_Diagnostics t="gpsdiag"/>
    </Main>
    <search>
        Records
    </search>
</Control>
```

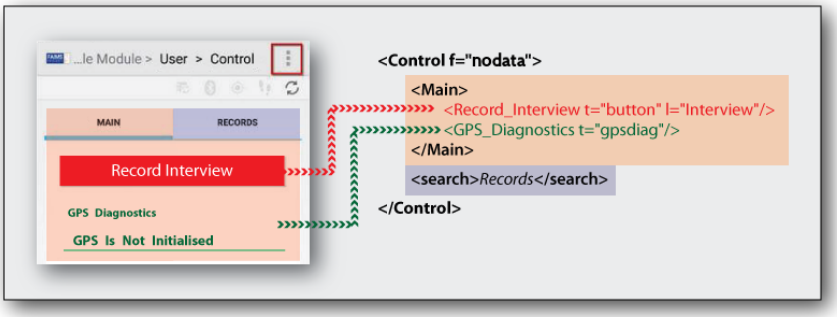

Figure 1.3 How XML in module.xml describes different FAIMS Elements.

The XML in module.xml describes the different FAIMS elements and how they should appear in the module.

This "Control" tab group encompasses two other elements, "Main" and "Search." Let's look at each individually.

The `<Main>` element creates the first tab. Inside that tab we have a GUI element, "Record_Interview." The t="button", which means this element is a button; 1="Interview", so the button links to another tab somewhere else in the module called "Interview." The other GUI element here is "GPS Diagnostics," which has the element type "gpsdiag." This element type specifically means that in the final module, it will create

text labels and display information about the Android device's GPS location. In the screenshot, we see the "gpsdiag" element at work: it's telling us that our phone's GPS is "not initialized," a charming way of saying "not turned on." Until we do turn it on, this is the best "gpsdiag" is going to do.

Let's turn on our Android device's internal GPS antenna by tapping *the three vertical dots* in the upper right of the screen to open the settings menu.

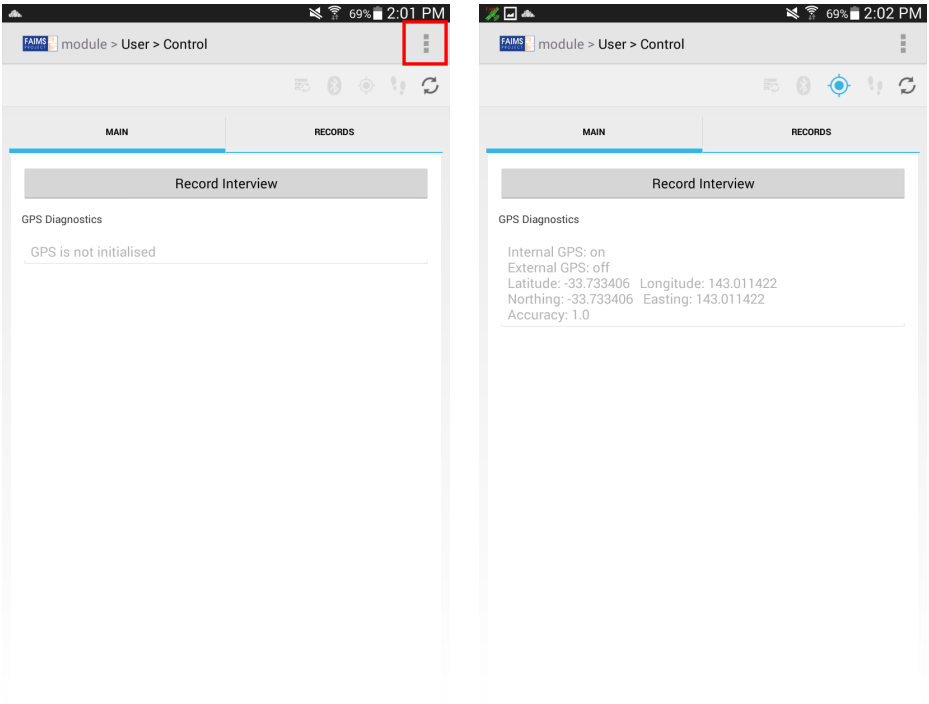

Figure 1.4 Turn on internal GPS by tapping the dots in the red box.

With the GPS antenna turned on, the "gpsdiag" element now displays quite a bit more information, including the status of the GPS antenna, location in both Latitude/Longitude and Easting/Northing, and an accuracy measurement.

So much for the `<Main>` tab. Let's look at the next tab, which you may have already noted is a little peculiar. For one thing, the tag name begins with a lowercase letter. Didn't we tell you never to do that? Furthermore, the text contained within, "Records," isn't in tags. What's going on here?

Don't worry; you haven't missed anything. It's just that there are certain kinds of tabs that FAIMS-Tools is already specially programmed to recognize. These kinds of tabs are complicated and a lot of work to make, so rather than ask you to create them, we've created a shorthand that FAIMS-Tools recognizes and runs with. These "shorthand" tabs have lowercase names, which is why you generally shouldn't come up with a lowercase tag name; you might accidentally write the name of a shorthand FAIMS-Tools recognizes, which will create a mess as it tries to follow its prewritten instructions at the same time as the instructions you've created.

In this case, the tab is "search," which, once some data has been collected, will contain GUI elements that allow you to search records according to various criteria, including term or entity type. We don't have to include code for all these GUI elements; FAIMS-Tools knows to include them in a tab labelled with the shorthand "search". The only thing we have included is the plaintext "Records", without a tag. When FAIMS-Tools sees text like this written inside an element, it labels the element that way instead of basing its name on the tab group. This is why you see "Records" as the tab label when you might have expected to see "search".

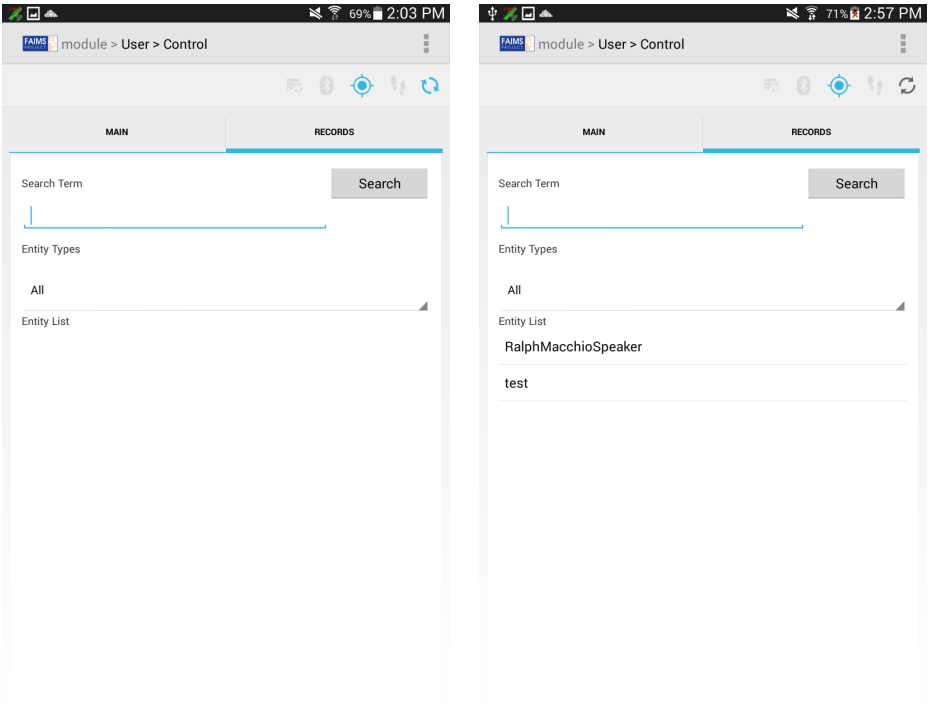

Figure 1.5 The Search element before and after adding a few records.

Going back to the finished module, let's tap the "Main" tab and then click the "Record Interview" button, which, you may recall, will take us to "Interview." "Interview" looks like Figure 1.6:

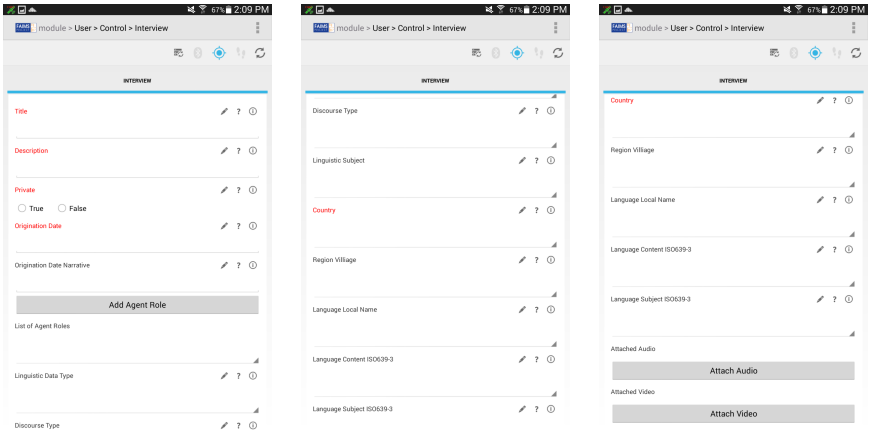

Figure 1.6 Scroll to see all fields in the "Interview" tab.

The code to create this long screen is below. This is a lot more code than you've seen before in one go, but it serves as a very useful example of a lot of different techniques, so resist the urge to skim it. Take your time, and when you don't recognize what a tag is or what an element is for, see if you can guess what it does just from context.

```
<Interview>
    <Interview>
        <Title f="id notnull">
            <desc>This title should be a sensible title, unique to each item, briefly
summarising the contents of the item, for example "Ilocano songs recorded in Burgos, Ilocos
Sur, Philippines, 17 April 1993"</desc>
        </Title>
        <Description f="notnull">
            <desc>Description may include but is not limited to: an abstract, table of
contents, reference to a graphical representation of content, or a free-text summary account of
the content. {[}DCMT{]} Description may also offer an annotation, or a qualitative or
evaluative comment about the resource, such as a statement about suitability for a particular
application or context.</desc>
        </Description>
        <Private t="radio" f="notnull">
            <desc>Choose either "false", meaning that the metadata for the item should be
publicly available, or "true", meaning that the metadata for the item should be hidden (perhaps
because you plan to check it and edit it later).</desc>
            <opts>
                <opt>True</opt>
                <opt>False</opt>
            </opts>
```

```
        </Private>
        <Origination_Date f="notnull">
            <desc>Date the item was captured or created, using the format yyyy-mm-dd. If you
are unsure of the day, month or decade enter the first day of the relevant period: e.g. "1970s"
1970-01-01, "2001" 2001-01-01, "February 1993" 1993-02-01. If entering a date of this type,
clarify in the originationDateNarrative field. If you really did record on 1 January 2001, say
so in the originationDate field.</desc>
        </Origination_Date>
        <Origination_Date_Narrative>
            <desc>Use this field to provide any necessary comments on the scope of the value
you entered in the origination date field, e.g. "unknown date in February 1993"</desc>
        </Origination_Date_Narrative>
        <Add_Agent_Role t="button" lc="Agent_Role"/>
        <List_of_Agent_Roles t="dropdown" ec="Agent_Role"/>
        <Linguistic_Data_Type>
            <desc>If data are relevant to linguistics, choose one of the three basic
linguistics data types. Primary text: Linguistic material which is itself the object of study;
Lexicon: a systematic listing of lexical items; Language description: describes a language or
some aspect(s) of a language via a systematic documentation of linguistic structures. If your
data are not relevant to linguistics, leave this field blank.</desc>
            <opts>
                <opt>Lexicon</opt>.
                <opt>Language Description</opt>
                <opt>Primary Text</opt>
            </opts>
        </Linguistic_Data_Type>
        <Discourse_Type>
            <desc>Used to describe the content of a resource as representing discourse of a
particular structural type. Dialogue: interactive discourse with two or more participants;
drama: planned, creative, rendition of discourse involving two or more participants; formulaic:
ritually or conventionally structured discourse; ludic: language whose primary function is to
be part of play, or a style of speech that involves a creative manipulation of the structures
of the language; oratory: public speaking, or of speaking eloquently according to rules or
conventions; narrative: monologic discourse which represents temporally organized events;
procedural: explanation or description of a method, process, or situation having ordered steps;
report: a factual account of some event or circumstance; singing: words or sounds
{[}articulated{]} in succession with musical inflections or modulations of the voice;
unintelligible: utterances that are not intended to be interpretable as ordinary
language.</desc>
            <opts>
                <opt>Dialogue</opt>
                <opt>Drama</opt>
                <opt>Narrative</opt>
                <opt>Procedural</opt>
                <opt>Ludic</opt>
                <opt>Singing</opt>
                <opt>Oratory</opt>
                <opt>Report</opt>
                <opt>Unintelligible speech</opt>
```

```
                <opt>Formulaic</opt>
            </opts>
        </Discourse_Type>
        <Linguistic_Subject>
            <desc>Use to describe the content of a resource if it is about a particular
subfield of linguistic science.</desc>
            <opts>
                <opt>Phonology</opt>
                <opt>Text And Corpus Linguistics</opt>
                <opt>Historical Linguistics</opt>
                <opt>Language Documentation</opt>
                <opt>Lexicography</opt>
                <opt>Typology</opt>
            </opts>
        </Linguistic_Subject>
        <Country f="notnull">
            <desc>This should be the standard name of the country in which the file was
recorded (see http://www.ethnologue.com/country_index.asp). Prefix the country name with the
two-letter ISO3166-1 code (http://www.iso.org/iso/country_codes.htm).</desc>
            <opts>
                <opt>PH - Philippines</opt>
                <opt>AU - Australia</opt>
            </opts>
        </Country>
        <Region_Villiage>
            <desc>Indicate the geographical scope of the item. Enter data in the order
locality, state or province, country.</desc>
            <opts>
                <opt>{[}locality{]}, {[}state or province{]}, {[}country{]}</opt>
                <opt>Burgos, Ilocos Sur, Philippines</opt>
            </opts>
        </Region_Villiage>
        <Language_Local_Name>
            <desc>The purpose of this field is to reflect language names in local use, with
local spellings, if different from official name.</desc>
            <opts>
                <opt>Language - local spelling {[}free text{]}</opt>
                <opt>Ilocano</opt>
            </opts>
        </Language_Local_Name>
        <Language_Content_ISO639-3>
            <desc>Content language is the language included in your data (spoken and/or
written). Insert the 3-letter ISO 639-3 code for your language, and the standard name of the
language as spelt in the ethnologue entry {[}search on www.ethnologue.com/site_search.asp{]}.
Separate the code and the language with a hyphen, e.g. "ilo - Ilocano"</desc>
            <opts>
                <opt>mis - Uncoded languages</opt>
                <opt>und - Undetermined languages</opt>
                <opt>mul - Multiple languages</opt>
```

```
                <opt>zxx - No linguistic content</opt>
                <opt>{[}3-letter ISO639-3 code{]} - {[}Ethnologue name of language{]}</opt>
                <opt>ilo - Ilocano</opt>
                <opt>eng - English</opt>
            </opts>
        </Language_Content_ISO639-3>
        <Language_Subject_ISO639-3>
            <desc>Subject language is the language that is the subject of your research. Insert
the 3-letter ISO 639-3 code for your language, and the standard name of the language as spelt
in the ethnologue entry {[}search on www.ethnologue.com/site_search.asp{]}. Separate the code
and the language with a hyphen, e.g. "ilo - Ilocano"</desc>
            <opts>
                <opt>zxx - No linguistic content</opt>
                <opt>{[}3-letter ISO639-3 code{]} - {[}Language subject of your
research{]}</opt>
                <opt>mis - Uncoded languages</opt>
                <opt>und - Undetermined languages</opt>
                <opt>mul - Multiple languages</opt>
                <opt>ilo - Ilocano</opt>
            </opts>
        </Language_Subject_ISO639-3>
        <Attached_Audio t="audio"/>
        <Attached_Video t="video"/>
    </Interview>
</Interview>
```

First, the easy stuff. You should already be able to guess what the first element, `<Interview>`, is: a tab group. Since there's another opening tag also called `<Interview>`, you've probably also guessed that the tab group `<Interview>` has a tab labelled `<Interview>`. So let's skip straight to the GUI elements located within this tab.

The first element is `<Title>` (Codeblock 1.3).

```
<Title f="id notnull">
    <desc>This title should be a sensible title, unique to each
item, briefly summarising the contents of the item, for example
"Ilocano songs recorded in Burgos, Ilocos Sur, Philippines, 17
April 1993"</desc>
</Title>
```

For the `<Title>` element, there is no UI "type" (represented by t=) specified. So it becomes the default "type;" a text input field. (See *Type Guessing for GUI Elements in FAIMS-Tools* for an explanation of how this was determined.) The flag "f=notnull" designates that this is a required field; the record cannot be saved if it is empty. If you can imagine your stress levels skyrocketing because of teammembers not remembering to enter in their (X), set the relevant element to "f=notnull" and they will have no choice but to remember.

`<desc>` allows you to set a description that your users can access by tapping and holding for a few seconds on the info button as in Figure 1.7.

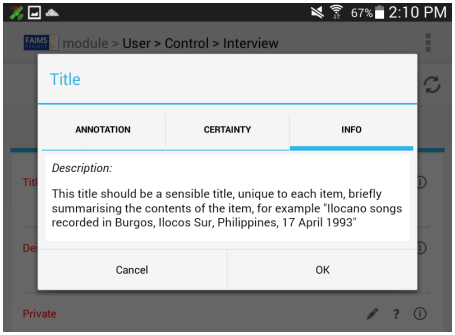

Figure 1.7 A description pop-up appears.

That wraps up the `<Title>` element. Now we have an opening tag for another GUI element, `<Private>`. Private is a radio button UI object, required, has a description, and includes two options ("True" and "False"). Review the code in Codeblock 1.4 and see if you can understand how all of this is accomplished.

This next element (Codeblock 1.5), `<Add_Agent_Role>`, designates a button element ('t="button"') that links to the tab group "Agent_Role." This will allow users to register new Agent Roles.

Note that this time, instead of using an "l" to redirect elsewhere, "lc" was used. The difference is that using "lc" instead of "l" establishes a parent-child relationship, with the entity linking becoming a parent and

```
<Private t="radio" f="notnull">
    <desc>Choose either "false", meaning that the metadata for the
item should be publicly available, or "true", meaning that the
metadata for the item should be hidden (perhaps because you plan
to check it and edit it later).</desc>
    <opts>
        <opt>True</opt>
        <opt>False</opt>
    </opts>
</Private>
```

```
<Add_Agent_Role t="button" lc="Agent_Role"/>
```

the entity being linked to becoming a child. This is useful for organizing your module's data in a neat, hierarchical way.

Codeblock 1.6 designates a dropdown menu, `<List_of_Agent_Roles>`, which is populated with a list of Agent_Role records. Specifically, they will be the Agent_Role records which were saved using the button element above. The FAIMS-Tools knows these are the right records to display because the button and dropdown menu appear in the same tab group.

```
<List_of_Agent_Roles t="dropdown" ec="Agent_Role"/>
```

The final two element types used in this tab are the 't="audio"'' and 't="video"' types (Codeblock 1.7). These two element types allow you to record audio and video files and attach them to your records.

```
<Attached_Audio t="audio"/>

<Attached_Video t="video"/>
```

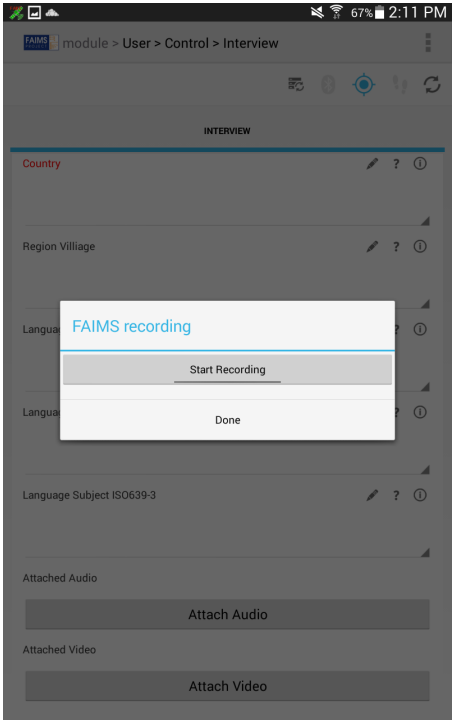

Figure 1.8 With the audio UI element, you'll get a popup window that allows you to start and stop your audio recording.

The final elements are on the Agent Role tab group, and include a few element types we've already seen: two text input fields: `<First_Name>` and `<Last_Name>` both with flags that designate them as "ids" and a dropdown menu, `<Role>` which contains a few options and is also flagged as an "id" (Codeblock 1.8).

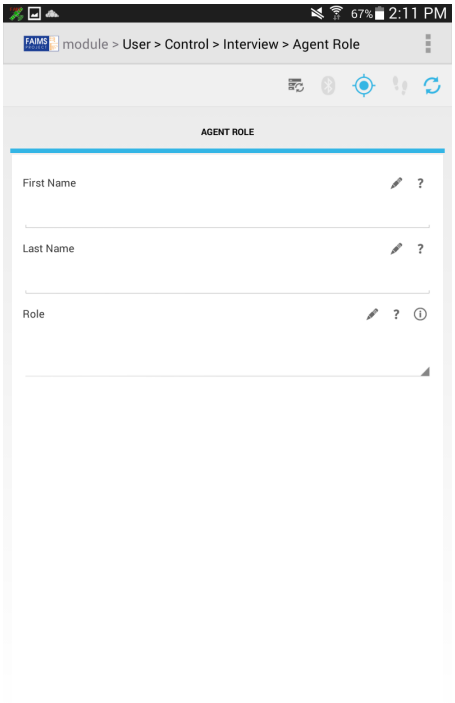

Figure 1.9 "Agent Role" tab group appears when adding agent roles.

Now that we've explained each part, go back and review the whole one more time. You can go a long way creating modules for your team only by using the techniques we've explicitly covered above. It's very possible that you've already learned everything you need to know to make an effective module for your team.

As an exercise, follow these instructions and produce your own copy of the module's code in module.xml. Then save that to the server and run the generate.sh script to produce necessary files. Go to "create module" and upload the necessary files to the server as the "Simple Sample Module."

Congratulations; you've just a simple module.

```
<Agent_Role>
    <desc>Enter participant name in the format Lastname,
Firstname. Choose the participant role from the closed vocabulary
provided. Use the description field to provide additional
information on role or agents. Enter participant name in the
format Lastname, Firstname. Choose the participant role from the
closed vocabulary provided. Add more participants by clicking the
"+" button to the right. If you need to provide extra information
on the agent or the role, use the item's "Description" field to
provide additional information on role or agents.</desc>
    <Agent_Role>
        <First_Name f="id"/>
        <Last_Name f="id"/>
        <Role f="id">
            <opts>
                <opt>Data Inputter</opt>
                <opt>Performer</opt>
                <opt>Speaker</opt>
                <opt>Developer</opt>
                <opt>Transcriber</opt>
                <opt>Photographer</opt>
                <opt>Interpreter</opt>
                <opt>Singer</opt>
                <opt>Signer</opt>
                <opt>Compiler</opt>
                <opt>Recorder</opt>
                <opt>Depositor</opt>
                <opt>Interviewer</opt>
                <opt>Editor</opt>
                <opt>Author</opt>
                <opt>Translator</opt>
                <opt>Researcher</opt>
                <opt>Annotator</opt>
                <opt>Participant</opt>
            </opts>
        </Role>
    </Agent_Role>
</Agent_Role>
```