SDMX-HD DSD Example and Technical Details

Example files

Attached is a zip file containing a DSD for two indicators. This example zip file will be used in the following examples to illustrate how to 'read' SDMX-HD packages, and prepare them to work with the OpenMRS integration.

SDMX-HD_example.zip

Package File Structure

If you look at the contents of the .zip file above, you'll notice that in the root of the archive is a file called DSD.xml. If you are building an SDMX message . zip file from scratch, you always need to make sure that the DSD.xml file is in the root of the archive, which might not be the case if you use a GUI archiver application. To ensure that the zip file is created correctly you can use the command line zip application, e.g.

```
cd <directory-containing-dsd-file>
zip -r <name-of-output-zip-file> . -x *.*~ *.DS_Store *.bak
```

The DSD.xml file is the main descriptor for the data set definition of the indicators that you are trying to export. It contains all of the concepts, codeLists, indicator disaggregation and indicator hierarchy descriptions, and key families for your export - all of which affect the structure of the indicator output.

The example package above was developed as an experiment to determine what the 'rules' are for exporting data in SDMX format. The ultimate goal of this work is to create a new SDMX-HD descriptor package for the 2012 Rwanda TracNET indicators. If you are building a new export, you could use the example file as a template for getting started. You just need to add your own indicators, disaggregations, and indicator hierarchy.

This example DSD is built on a WHO-published example. Many of the included files aren't used. The best way to determine what's important in a DSD is to look in the KeyFamily section and work backwards.

How Indicators Are Defined:

The way the OpenMRS SDMX-HD module determines what elements are 'indicators' as follows:

1) a Dimension defined in the KeyFamily section in the DSD is found that is mapped to CL_INDICATOR:

```
  <structure:Dimension conceptRef="INDICATOR" conceptSchemeRef="CS_COMMON" conceptVersion="1.0"
conceptSchemeAgency="SDMX-HD" codelist="CL_INDICATOR" codelistVersion="1.0" codelistAgency="SDMX-HD"/>
```

2) the CodeList CL_INDICATOR is loaded from the Dimension, and the Codes in the CodeList define each raw indicator.

In the example above, you can follow the pointers to find the indicators. The CodeList CL_INDICATOR referenced by the indicator dimension points to CL_INDICATOR, which is defined in the CodeList section of the DSD (just do a string search). You will see that CL_INDICATOR is defined by the following line in DSD.xml:

```
 <structure:CodeList id="CL_INDICATOR"  agencyID="SDMX-HD" version="1.0" isFinal="false" urn="urn:sdmx:org.sdmx.
infomodel.codelist=SDMX-HD:CL_INDICATOR" isExternalReference="true" uri="./additional/CL_INDICATOR.xml"
><structure:Name xml:lang="en">Indicator</structure:Name></structure:CodeList>
```

This then points to the external file ./additional/CL_INDICATOR.xml, which can be found at that location in the .zip file.

The contents of this file are fairly easy to read. This file defines a list of indicators:

```
<?xml version="1.0" encoding="UTF-8"?><?xml-stylesheet type="text/xsl" href="codelist.xsl"?>
<Structure xmlns="http://www.SDMX.org/resources/SDMXML/schemas/v2_0/message" xmlns:structure="http://www.SDMX.
org/resources/SDMXML/schemas/v2_0/structure" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:
schemaLocation="http://www.SDMX.org/resources/SDMXML/schemas/v2_0/message SDMXMessage.xsd">
  <Header>
    <ID>SDMX-HD-DSD</ID>
    <Test>false</Test>
    <Truncated>false</Truncated>
    <Name xml:lang="en">TRAC PLUS</Name>
    <Prepared>2009-03-20</Prepared>
    <Sender id="TRACPLUS">
      <Name>TRAC PLUS</Name>
      <Contact>
        <Name xml:lang="en">Admin</Name>
        <Department>Health Care Informatics</Department>
      </Contact>
    </Sender>
  </Header>
  <CodeLists>
    <structure:CodeList id="CL_INDICATOR" agencyID="SDMX-HD" version="1.0" isFinal="false" urn="urn:sdmx:org.
sdmx.infomodel.codelist=SDMX-HD:CL_INDICATOR" >
      <structure:Name xml:lang="en">Indicator</structure:Name>
      <structure:Description xml:lang="en">This is a placeholder codelist: populate it with indicator names.<
/structure:Description>
    <structure:Code value="1" urn="urn:sdmx:org.sdmx.infomodel.codelist.Code=SDMX-HD:CL_INDICATOR[1.0].1">
       <structure:Description xml:lang="en">TEST1: adult initial</structure:Description>
    </structure:Code>
    <structure:Code value="2" urn="urn:sdmx:org.sdmx.infomodel.codelist.Code=SDMX-HD:CL_INDICATOR[1.0].2">
       <structure:Description xml:lang="en">Test2: adult return</structure:Description>
    </structure:Code>
    </structure:CodeList>
  </CodeLists>
</Structure>
```

In this example, there are two indicators, one called 'TEST1: adult initial' with value = 1, and 'TEST2: adult return'. In production, these would be real indicators with names like 'Number of Pediatric Primary Care Visits'. It is essential that each indicator has a unique value.

How Dimensions Are Defined:

Dimensions can be defined similarly to Indicators, but they don't have to have a specific name (Indicators, however, must be defined in CL_INDICATOR). In the example DSD, one of the dimensions is Gender. Following the pointers you'll see that:

1) in the KeyFamily declaration in the DSD the gender dimension is defined:

```
<structure:Dimension conceptRef="GENDER" conceptSchemeRef="CS_COMMON" conceptVersion="1.0"
  conceptSchemeAgency="SDMX-HD" codelist="CL_GENDER_TRAC" codelistVersion="1.0" codelistAgency="TRACPLUS"/>
```

This then points back to the CL_GENDER_TRAC CodeList, which is defined in the CodeList section of the DSD:

```
<structure:CodeList id="CL_GENDER_TRAC" agencyID="TRACPLUS" version="1.0" isFinal="false"
  urn="urn:sdmx:org.sdmx.infomodel.codelist=TRACPLUS:CL_GENDER_TRAC" isExternalReference="true"
  uri="./CUSTOM/TRACPLUS/v1.0/codelists/CL_GENDER+TRACPLUS+1.0.xml">
  <structure:Name xml:lang="en">Gender</structure:Name>
</structure:CodeList>
```

This then points to the file ./Custom/TRACPLUS/v1.0/codelists/CL_GENDER+TRACPLUS+1.0.xml

This file contains all the possible dimension values for Gender: Male and Female, where Male is 0 and Female is 1

```
<?xml version="1.0" encoding="UTF-8"?>
<Structure xmlns="http://www.SDMX.org/resources/SDMXML/schemas/v2_0/message" xmlns:structure="http://www.SDMX.
org/resources/SDMXML/schemas/v2_0/structure" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:
schemaLocation="http://www.SDMX.org/resources/SDMXML/schemas/v2_0/message SDMXMessage.xsd">
  <Header>
    <ID>SDMX-HD-DSD</ID>
    <Test>false</Test>
    <Truncated>false</Truncated>
    <Name xml:lang="en">TRAC PLUS</Name>
    <Prepared>2009-03-20</Prepared>
    <Sender id="TRACPLUS">
      <Name>TRAC PLUS</Name>
      <Contact>
        <Name xml:lang="en">Admin</Name>
        <Department>Health Care Informatics</Department>
      </Contact>
    </Sender>
  </Header>
  <CodeLists>
    <structure:CodeList id="CL_GENDER_TRAC" agencyID="TRACPLUS" version="1.0" urn="urn:sdmx:org.sdmx.infomodel.
codelist=TRACPLUS:CL_GENDER_TRAC">
      <structure:Name xml:lang="en">Gender</structure:Name>
      <structure:Code value="0" urn="urn:sdmx:org.sdmx.infomodel.codelist.Code=TRACPLUS:CL_GENDER_TRAC[1.0].0">
        <structure:Description xml:lang="en">Male</structure:Description>
      </structure:Code>
      <structure:Code value="1" urn="urn:sdmx:org.sdmx.infomodel.codelist.Code=TRACPLUS:CL_GENDER_TRAC[1.0].1">
        <structure:Description xml:lang="en">Female</structure:Description>
      </structure:Code>
    </structure:CodeList>
  </CodeLists>
</Structure>
```

So How Do I Use These Dimensions With OpenMRS?

Dimension Hierarchies can be applied on an indicator-by-indicator basis. To see how this works, look at the CodeListHeirarchy section of the DSD, which says:

```
<structure:Hierarchy id="HY_INDICATOR_DISAGGREGATION">
  <structure:Name>Indicator to Disaggregation Hierarchy</structure:Name>
  <!-- ADD THE CODEREFs HERE -->
  <!-- Explicitly list all indicators, even if there are no associated dimensions.-->
  <!-- If an indicator is not mentioned here it will default to all possible dimensions being calculated.-->
  <structure:CodeRef>
    <structure:CodelistAliasRef>AL_INDICATOR</structure:CodelistAliasRef>
    <structure:CodeID>1</structure:CodeID>
    <structure:CodeRef>
      <structure:CodelistAliasRef>AL_GENDER_TRAC</structure:CodelistAliasRef>
      <structure:CodeID>0</structure:CodeID>
    </structure:CodeRef>
    <structure:CodeRef>
      <structure:CodelistAliasRef>AL_GENDER_TRAC</structure:CodelistAliasRef>
      <structure:CodeID>1</structure:CodeID>
    </structure:CodeRef>
  </structure:CodeRef>
  <structure:CodeRef>
    <structure:CodelistAliasRef>AL_INDICATOR</structure:CodelistAliasRef>
    <structure:CodeID>2</structure:CodeID>
    <structure:CodeRef>
      <structure:CodelistAliasRef>AL_GENDER_TRAC</structure:CodelistAliasRef>
      <structure:CodeID>0</structure:CodeID>
      <structure:CodeRef>
        <structure:CodelistAliasRef>AL_AGE_GROUP_TRAC</structure:CodelistAliasRef>
        <structure:CodeID>0</structure:CodeID>
      </structure:CodeRef>
    </structure:CodeRef>
    <structure:CodeRef>
      <structure:CodelistAliasRef>AL_GENDER_TRAC</structure:CodelistAliasRef>
      <structure:CodeID>1</structure:CodeID>
    </structure:CodeRef>
  </structure:CodeRef>
</structure:Hierarchy>
```

In this example, you can see how Gender is applied to the two indicators. Indicator 1 is disaggregated into Male And Female. Indicator 2 is disaggregated into Male And Female, with Males being broken down by another Age Group dimension.

You will also notice that this section uses aliases to refer to CodeLists. This seems to be standard practice for hierarchies in the DSD, and you'll notice that the code lists are matched with their aliases at the beginning of the heirarchy section in the DSD:

```
<structure:CodelistRef>
  <structure:AgencyID>SDMX-HD</structure:AgencyID>
  <structure:CodelistID>CL_INDICATOR</structure:CodelistID>
  <structure:Version>1.0</structure:Version>
  <structure:Alias>AL_INDICATOR</structure:Alias>
</structure:CodelistRef>
<structure:CodelistRef>
  <structure:AgencyID>TRACPLUS</structure:AgencyID>
  <structure:CodelistID>CL_AGE_GROUP_TRAC</structure:CodelistID>
  <structure:Version>1.0</structure:Version>
  <structure:Alias>AL_AGE_GROUP_TRAC</structure:Alias>
</structure:CodelistRef>
<structure:CodelistRef>
  <structure:AgencyID>TRACPLUS</structure:AgencyID>
  <structure:CodelistID>CL_GENDER_TRAC</structure:CodelistID>
  <structure:Version>1.0</structure:Version>
  <structure:Alias>AL_GENDER_TRAC</structure:Alias>
</structure:CodelistRef>
<structure:CodelistRef>
  <structure:AgencyID>TRACPLUS</structure:AgencyID>
  <structure:CodelistID>CL_ISET</structure:CodelistID>
  <structure:Version>1.0</structure:Version>
  <structure:Alias>AL_ISET</structure:Alias>
</structure:CodelistRef>
```

This file results in the following lines in the output file, as long as you have your Gender (and Age Group) dimensions mapped correctly in OpenMRS (see Ryan's tutorial on how to do this).

```
<ns:OBS_VALUE GENDER="0" value="37" INDICATOR="1"/>
<ns:OBS_VALUE GENDER="1" value="45" INDICATOR="1"/>
<ns:OBS_VALUE AGROUP="0" GENDER="99" value="0" INDICATOR="2"/>
<ns:OBS_VALUE GENDER="1" value="55" INDICATOR="2"/>
```

You should be able to see how this output reflects the hierarchy above. Indicator 1 is disaggregated by Gender, Indicator 2 is disaggregated by Gender, with Males having the extra AGROUP (Age Group) applied.

IMPORTANT NOTE ABOUT DIMENSION HIERARCHIES: By default, if you have dimensions defined in your KeyFamily in your DSD, all indicators that you don't include in the HY_INDICATOR_DISAGGREGATION hierarchy in your DSD will automatically have all possible dimensions applied to them. So, for example, if you built a DSD with the gender dimension in the KeyFamily, but you didn't explicitly relate Gender to your indicators in HY_INDICATOR_DISAGGREGATION, the result would be that both indicator 1 and 2 would be disaggregated in the result file by Gender. If you only wanted indicator 1 to be disaggregated, but not indicator 2, you need to explicitly say so in the HY_INDICATOR_DISAGGREGATION section, like so:

```
<structure:Hierarchy id="HY_INDICATOR_DISAGGREGATION">
  <structure:Name>Indicator to Disaggregation Hierarchy</structure:Name>
  <structure:CodeRef>
    <structure:CodelistAliasRef>AL_INDICATOR</structure:CodelistAliasRef>
    <structure:CodeID>1</structure:CodeID>
  <structure:CodeRef>
    <structure:CodelistAliasRef>AL_GENDER_TRAC</structure:CodelistAliasRef>
    <structure:CodeID>0</structure:CodeID>
  </structure:CodeRef>
  <structure:CodeRef>
    <structure:CodelistAliasRef>AL_GENDER_TRAC</structure:CodelistAliasRef>
    <structure:CodeID>1</structure:CodeID>
  </structure:CodeRef>
  </structure:CodeRef>
  <structure:CodeRef>
    <structure:CodelistAliasRef>AL_INDICATOR</structure:CodelistAliasRef>
    <structure:CodeID>2</structure:CodeID>
  </structure:CodeRef>
</structure:Hierarchy>
```

You'll notice here that indicator 2 is referenced, but has no Gender applied. The output file for this descriptor will be something like:

```
<ns:OBS_VALUE GENDER="0" value="37" INDICATOR="1"/>
<ns:OBS_VALUE GENDER="1" value="45" INDICATOR="1"/>
<ns:OBS_VALUE value="154" INDICATOR="2"/>
```

How Do I Setup Dimensions In OpenMRS To Match This Example?

There is a simple list of steps that were undertaken to test the above SDMX-HD package. For more information, see the documentation for the Reporting Module.

1) In the reporting module, I first designed a couple of Cohort Queries. These are

a) a SQLCohortQuery that I called 'Gender'. This I defined as:

```
select person_id from person where gender = :gender
```

with a String parameter called 'gender'.

b) Two age queries, one called 'Over age X in Years' and one called 'Under Age X In Years'. Each of these i just used the UI to define. Nothing tricky here.

c) I created two EncounterQueries to represent the logic behind my base indicators. In these, i just return 1) a count of all 'adult initial' encounters, and 2) a count of all 'adult return' encounters.

2) I defined my indicators. To do this, create two new IndicatorDefinitions, each mapped to one of the EncounterQueries described above.

3) I defined my dimensions. To do this, create two new DimensionDefinitions, one for Gender and one for AgeGroup. For Gender, I defined two keys, each mapped to our Gender CohortDefinition. The first one is MALE, and should parameterize the 'gender' parameter with 'M', the second is Female which parameterizes the same CohortDefinition with 'F'.

For the AgeGroup dimensions, I did something similar.

The main idea here is that you should create a dimension KEY for each item that you have in the descriptor file in the .zip SDMX-HD file for that dimension. For example, in my CL_AGE_GROUP_TRAC CodeList, I have the following items:

- 15 and over
- Under 15
- Under 5
- Under 18 Months.

So, in my Age Group DimensionDefinition in OpenMRS, i need to define a key for each of these (it may be helpful to use the same name for the key in OpenMRS as you use in the CodeList descriptions, so there is no ambiguity about what maps to what).

Finally, it is important to point out that you can have a different CohortDefinition (or other OpenMRS Definition Type) for each Key in a dimension. You can make your indicator and dimension definitions as 'smart' as they need to be.

4) Finally, follow the instructions in Ryan's tutorial for how to upload and map an SDMX-HD package. ***It is important to do things in order*** -- configuration first, then dimensions, then indicators, then attributes. Then you should be able to just run the thing.

How To Define Sections

In the 0.6.6 version of the SDMX-integration module, you can have your indicators grouped into sections in the output. I'm going to use this in the final TracNET output to explicitly group ART and PRE-ART indicators (these are the two values in the CL_ISET CodeList).

Each group is defined in a CodeList file called CL_ISET declared in the CodeList section of the DSD:

```
<structure:CodeList id="CL_ISET" agencyID="SDMX-HD" version="1.0" isFinal="false" urn="urn:sdmx:org.sdmx.
infomodel.codelist=SDMX-HD:CL_ISET" isExternalReference="true" uri="./additional/CL_ISET.xml"><structure:Name
xml:lang="en">Indicator Set</structure:Name></structure:CodeList>
```

Then, you can explicitly put each indicator into a Section in the output in the INDICATOR_SET_INDICATOR_HIERARCHY:

```
<structure:Hierarchy id="INDICATOR_SET_INDICATOR_HIERARCHY"><structure:Name>Indicator Set to Indicator
Hierarchy</structure:Name>
  <structure:CodeRef>
    <structure:CodelistAliasRef>AL_ISET</structure:CodelistAliasRef>
    <structure:CodeID>0</structure:CodeID>
    <structure:CodeRef>
      <structure:CodelistAliasRef>AL_INDICATOR</structure:CodelistAliasRef>
      <structure:CodeID>1</structure:CodeID>
    </structure:CodeRef>
  </structure:CodeRef>
  <structure:CodeRef>
    <structure:CodelistAliasRef>AL_ISET</structure:CodelistAliasRef>
    <structure:CodeID>1</structure:CodeID>
    <structure:CodeRef>
      <structure:CodelistAliasRef>AL_INDICATOR</structure:CodelistAliasRef>
      <structure:CodeID>2</structure:CodeID>
    </structure:CodeRef>
  </structure:CodeRef>
</structure:Hierarchy>
```

If you want to use Sections, you must explicitly include all indicators in a Section explicitly. If you define no hierarchy for your indicators, all indicators will end up in a default Section in the output. Here is the complete output of report, as defined by the DSD attached to this page. Note how the sections match the sections descibed here:

```
<CrossSectionalData xmlns="http://www.SDMX.org/resources/SDMXML/schemas/v2_0/message" xmlns:ns="urn:sdmx:org.
sdmx.infomodel.keyfamily.KeyFamily=WHO:SDMX-HD:1.0:cross" xmlns:cross="http://www.SDMX.org/resources/SDMXML
/schemas/v2_0/cross" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.SDMX.
org/resources/SDMXML/schemas/v2_0/structure SDMXMessage.xsd urn:sdmx:org.sdmx.infomodel.keyfamily.KeyFamily=WHO:
SDMX-HD:1.0:cross CrossSectional.xsd http://www.SDMX.org/resources/SDMXML/schemas/v2_0/cross
SDMXCrossSectionalData.xsd">
  <Header>
    <ID>SDMX-HD-CSDS</ID>
    <Test>false</Test>
    <Truncated>false</Truncated>
    <Name xml:lang="en">OpenMRS SDMX-HD Export</Name>
    <Prepared>2012-01-04</Prepared>
    <Sender id="OMRS">
      <Name>OpenMRS</Name>
    </Sender>
    <ReportingBegin>2012-01-01</ReportingBegin>
    <ReportingEnd>2012-01-31</ReportingEnd>
  </Header>
  <ns:DataSet reportingBeginDate="2012-01-01" reportingEndDate="2012-01-31" dataProviderID="417">
    <ns:Group TIME_PERIOD="2012-01" FREQ="M">
      <ns:Section description="PRE-ART">
        <ns:OBS_VALUE GENDER="0" value="0" INDICATOR="1"/>
        <ns:OBS_VALUE GENDER="1" value="0" INDICATOR="1"/>
      </ns:Section>
      <ns:Section description="ART">
        <ns:OBS_VALUE AGROUP="0" GENDER="0" value="0" INDICATOR="2"/>
        <ns:OBS_VALUE GENDER="1" value="0" INDICATOR="2"/>
      </ns:Section>
    </ns:Group>
  </ns:DataSet>
</CrossSectionalData>
```

Notice how each indicator ended up in its own Section.

Notes About Final Report Output

The above output is the complete SDMX-HD indicator result for the DSD attached at the top of this page. When you run an SDMX-HD report in OpenMRS, you must choose the SDMX-HD renderer. When you do this, you get a .zip file, and this .zip file contains the entire SDMX-HD .zip file you started with, with the addition of DATA_CROSS.xml which is added to the root directory of the zip file, next to DSD.xml. DATA_CROSS.xml contains the indicator result shown above.

1) First, you'll notice that there is a DataSet attribute called 'dataProviderId'. This is an optional attribute that you can set in the SDMX OpenMRS interface during the SDMX to OpenMRS indicator mapping process. When you're in the configuration screen, you can set this to a fixed value. For the TracNET report, this is where each center will enter the FOSA ID for that center (FOSA IDs are numeric identifiers for each health facility in Rwanda, issued by the Rwandan Ministry of Health).

2) Second, you'll notice that the Group item contains the attributes describing the time period for the report. This is a special, and optional, feature of the SDMX integration module. To get this to work, you need a CodeList explicitly called CL_FREQ, and this CodeList needs to include the Codes 'M' for monthly and 'A' for annual. Basically, you can follow from the KeyFamily:

<structure:Attribute conceptRef="FREQ" conceptSchemeRef="CS_COMMON" conceptVersion="1.0" conceptSchemeAgency="SDMX-HD" codelist="CL_FREQ" codelistVersion="1.0" codelistAgency="SDMX-HD" attachmentLevel="Group" assignmentStatus="Mandatory"/>

```
<structure:Attribute conceptRef="FREQ" conceptSchemeRef="CS_COMMON" conceptVersion="1.0"
  conceptSchemeAgency="SDMX-HD" codelist="CL_FREQ" codelistVersion="1.0" codelistAgency="SDMX-HD"
  attachmentLevel="Group" assignmentStatus="Mandatory"/>
```

points to the CodeList:

```
<structure:CodeList id="CL_FREQ" agencyID="TRACPLUS" version="1.0" isFinal="false"
  urn="urn:sdmx:org.sdmx.infomodel.codelist=TRACPLUS:CL_FREQ_TRAC" isExternalReference="true"
  uri="./CUSTOM/TRACPLUS/v1.0/codelists/CL_FREQ+TRACPLUS+1.0.xml">
  <structure:Name xml:lang="en">Report Frequency</structure:Name>
</structure:CodeList>
```

which then points to the file CL_FREQ+TRACPLUS+1.0.xml, which contains the CodeList values.

If you define all of these things in your DSD, in the OpenMRS SDMX interface, in the configuration page, you'll see a drop-down that allows you to choose Monthly or Annual. For the TracNET report, we'll be using Monthly. Using this feature also throws an error if the report startDate and endDate are not valid dates for the first and last day of a given month (if monthly) or year (if annual).