Institute of Mathematics and Computer Science
European Bioinformatics Institute

SIMS
Configuration Guide

Version 2.14

Table of Contents

1 Database

SIMS DB is based on three-level architecture: Bottom level (*a3_samples*) → Middle level (*a2_visits*) → Top level (*a1_persons*). In the following text these tables are referenced as "level tables". There is a meta-data table associated with each level table: Top meta-data table (*a1_person_meta_data*), Middle meta-data table (*a2_visit_meta_data*), Bottom meta-data table (*a3_sample_meta_data*). In the following text these tables are referenced as "meta-data tables".

See Figure 1 Default SIMS DB configuration.

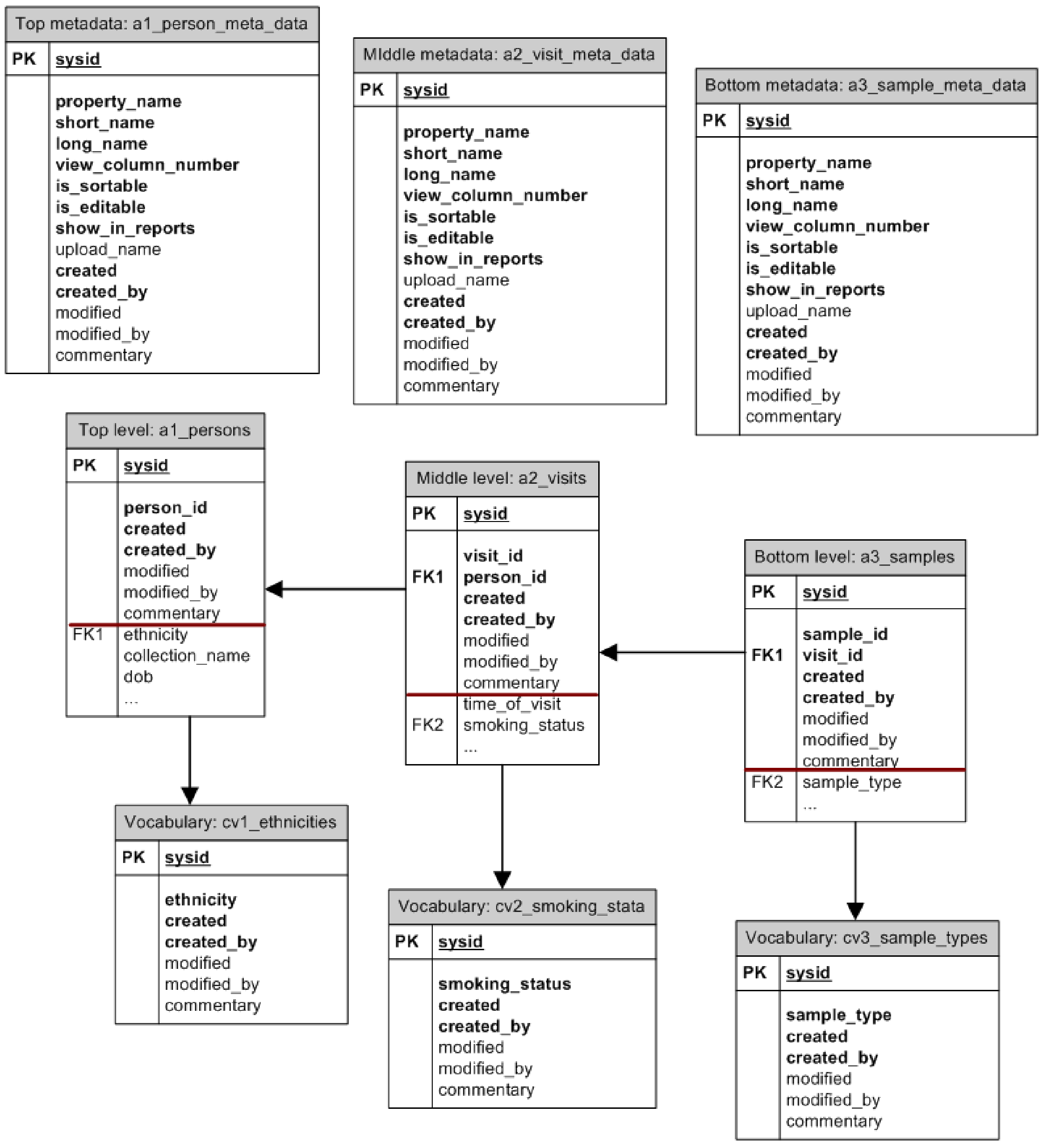

Figure 1 ***Default SIMS DB configuration***

All changes in SIMS configuration start at database. It is possible to change level table name and attributes (to delete, to add new, to rename attributes).

Level tables

There are following mandatory attributes for each of the three mentioned level tables:

Attribute	Type	Description
sysid	bigint	Internal record identificator. Not null.
a<level><name>_id	character varying(400)	External record identificator. Not null.
<previous_level_name>	character varying(400)	If not Top level. Reference to the previous level. Not null.
created_by	Bigint	SIMS user who created the record. Not null.
modified_by	Bigint	SIMS user who last modified the record
created	timestamp without time zone	Record creation date and time. Not null.
modified	timestamp without time zone	Last record modification date and time.
commentary	Text	Comment in free text format
owner_group	bigint	SIMS user group that this record belongs to
technology	bigint	The collection that this record belongs to
is_pending	bigint	Record status in case of transfer
batch_id	character varying(400)	
protocol_files	character varying(400)	
data_files	character varying(400)	

Table 1 ***Level table***

Other level table attributes are optional and can be changed if needed. The following types of optional attributes are supported in SIMS:

SIMS notation	Title	Description
IT	Integer	Integer is a *bigint* datatype in PostgreSQL
ST	String	String is *character varying (400)* datatype in PostgreSQL
DT	Date	Date is "*timestamp without time zone*" datatype in PostgreSQL
FT	Float	Float is *numeric(20,5)* datatype in PostgreSQL

VT	Vocabulary	Vocabulary is not the real type of column, but "label" for the case when table's column references to vocabulary (small table in DB with particular structure). This type will be presented as list in SIMS forms with values from appropriate vocabulary table.
DVT	Dependant vocabulary	
IVT	ID influencing vocabulary	

Table 2 ***Types of attributes***

Vocabularies

1. In the default configuration there are a lot of vocabulary tables all of them are named **cv**<level>**_**<name> (example: "*cv3_sample_types*"). Level indicates that this table is referenced from main table of appropriate level (example: "*cv3_sample_transport_conditions*" is referenced from table "*a3_samples*").
2. In vocabularies that are named **cv0_**<name> are listed values for common promptings in SIMS forms.
3. Vocabularies that participate in ID generation have an extra column "*abbreviation*" (type *character varying(400)*). Such tables are named the same way.
4. Vocabularies with names **cv**<level>**m_**<table name1>_<table name2> indicate that this table is used to link together main vocabulary and dependant vocabulary (example: "*cv3m_sample_types_sample_subtypes*" links together entries from main vocabulary "*cv3_sample_types*" and dependant vocabulary "*cv3_sample_subtypes*").

N.B. Described here naming principles of tables in SIMS database are not required condition for system functioning. User can rename tables following other rules.

Described types of optional attributes are associated with columns of level tables using hibernate mappings and meta-data tables.

Meta-data tables

Structure of meta-data tables can't be changed for normal SIMS functioning. However, context of meta-data tables has to be changed together with changes in level tables.

Example of new configuration

Let assumed that in new SIMS configuration Middle level table is renamed – *a2_characteristics*, attribute *time_of_visit* is renamed into *measurement_time* and number of attributes are added:

string_characteristic is **ST** attribute (see Table 2 Types of attributes), *integer_characteristic* is **IT** attribute, *float_characteristic* is **FT** attribute and *list_characteristic* is **VT** attribute. Since new VT attribute has been added new vocabulary table is created (*cv2_list_values*) and new foreign key is created for Middle level table *a2_characteristics*.

See Figure 2 Example of new SIMS DB configuration.

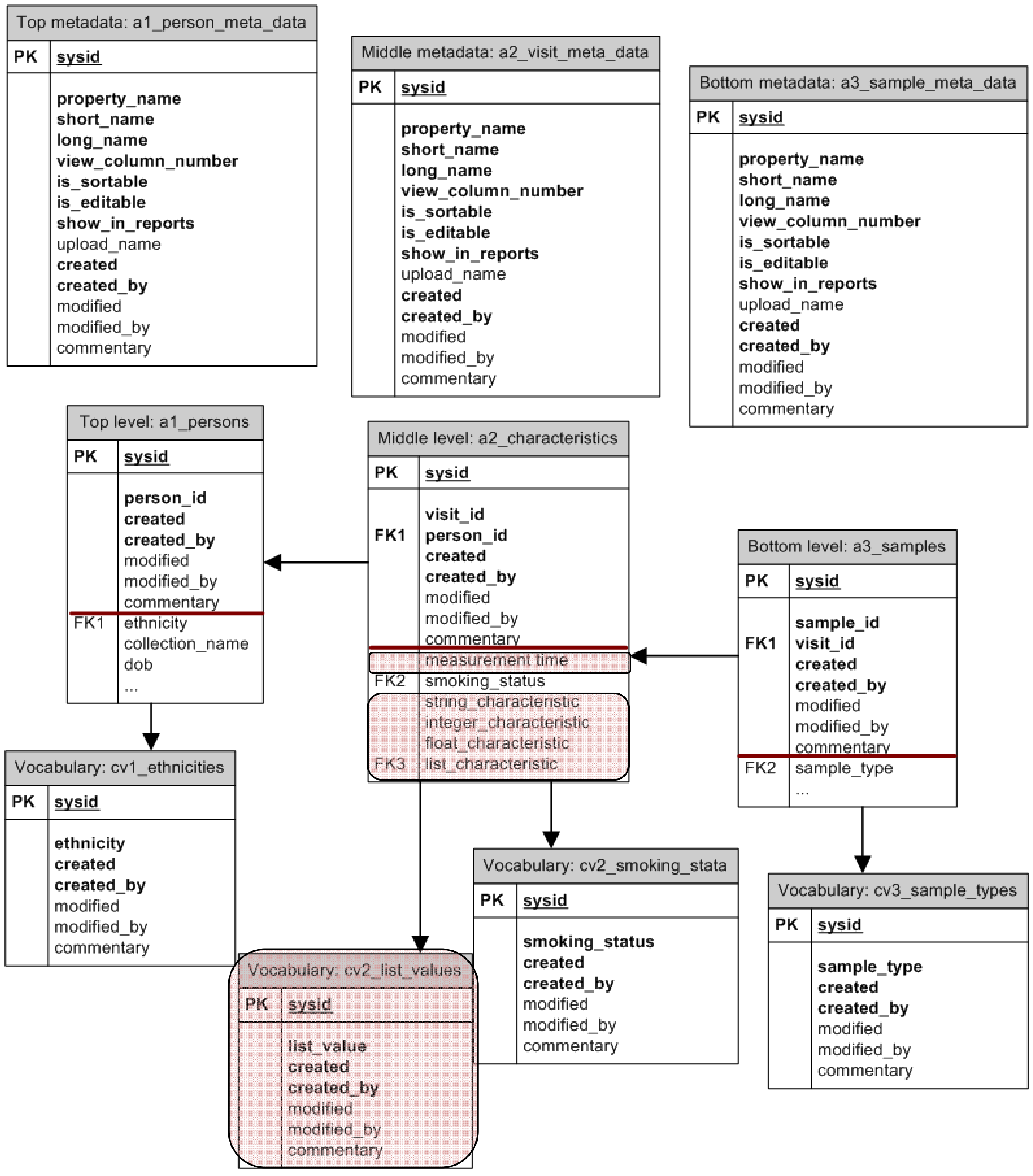

Figure 2 ***Example of new SIMS DB configuration***

After changes in DB context of meta-data tables and hibernate mappings have to be changed in appropriate way. See details in the following sections.

2 Hibernate Mappings

Hibernate mapping has to be changed in appropriate way together with changes in structure of level tables. Besides, new Hibernate mapping files have to be created together with new vocabulary table.

1. Top level appropriate mapping file is: */.../sims/hibernate/Top.hbm.xml*
2. Middle level appropriate mapping file is: */.../sims/hibernate/Middle.hbm.xml*
3. Bottom level appropriate mapping file is: */.../sims/hibernate/Bottom.hbm.xml*
4. Vocabulary's tables: */.../sims/hibernate/VT<number>.hbm.xml*
5. ID changing vocabularies mapping file: */.../sims/hibernate/IVT<number>.hbm.xml*
6. Dependant vocabularies mapping file: */.../sims/hibernate/DVT<number>.hbm.xml*

The changes can be divided into following parts:

1. Table name. If level or vocabulary table name has been changed then in the mapping file string *<class name="Top/Middle/Bottom" table="a1/2/3_old name">* has to be changed: *<class name="Top/Middle/Bottom" table="a1/2/3_new name">*

In example of new DB configuration (Figure 2) Middle level table name *a2_visits* has been changed to *a2_characteristics*. In Hibernate mapping file */.../sims/hibernate/Middle.hbm.xml* the same change has to be done:

```
...
 <class name="Middle" table="a2_characteristics">
...
```

2. Attributes.

Attributes of level tables are mapped to appropriate types (ST, IT, FT, DT or VT) in Hibernate mapping files.

In example (Figure 2) attribute *time_of_visit* of table *a2_characteristic* has been changed to *measurement_time*.

In Hibernate mapping file */.../sims/hibernate/Middle.hbm.xml* the same change has to be done:

```
        ...
        <property name="dt01" column="measurement_time"/>
        ...
```

Here dt01 means that this attribute has type DT and is first attribute (01) with DT type in Middle level table.

Mappings have to be added into Hibernate mapping file */.../sims/hibernate/Middle.hbm.xml* for all new attributes (changes are marked with red color):

```
        <?xml version="1.0"?>
        <!DOCTYPE hibernate-mapping PUBLIC
```

```
            "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
            "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping
            package="sims.hibernate">
            <class name="Middle" table="a2_characteristics">
                        <id name="id" column="sysid">
                                    <generator class="native"/>
                        </id>
                        <property name="visibleName" column="visit_id" not-null="true"/>
                        <many-to-one name="parent" column="person" not-null="true"/>
                        <many-to-one name="creator" column="created_by" not-null="true"/>
                        <many-to-one name="modifier" column="modified_by"/>
                        <many-to-one name="technology" column="technology"/>
                        <property name="createDate" column="created" not-null="true"/>
                        <property name="modifDate" column="modified"/>
                        <property name="protocolFiles" column="protocol_files"/>
                        <property name="dataFiles" column="data_files"/>
                        <property name="pendingX" column="is_pending"/>
                        <property name="batchId" column="batch_id"/>
                        <property name="comment" column="commentary"/>
                        <bag name="aliquots" lazy="true" inverse="true" cascade="save-update">
                                    <key column="from_visit"/>
                                    <one-to-many class="Bottom"/>
                        </bag>
<!-- configuration of integer types -->
                        <property name="it01" column="age_first_myocardial_infarction"/>
                        ...
                        <property name="it13" column="integer_characteristic"/>
<!-- configuration of float types -->

                        <property name="ft01" column="weight"/>
                        ...
                        <property name="ft27" column="float_characteristic"/>
<!-- configuration of string types -->
                        <property name="st02" column="glucose_timing"/>
                        ...
                        <property name="st23" column="string_characteristic"/>
<!-- configuration of timestamp types -->
                        <property name="dt01" column="measurement_time"/>
<!-- configuration of cv types -->
                        <many-to-one name="vt03" column="alcohol_status"/>
                        ...
                        <many-to-one name="vt30" column="list_characteristic"/>
<!-- configuration of cv types with identifiers-->
                        <many-to-one name="ivt01" column="sample_type"/>
<!-- the configurable part ends here -->
            </class>
</hibernate-mapping>
```

Changes that have been made in SIMS DB and Hibernate mapping files require also changes in systemConfig.xml file (see section “System Configuration”). Changes in java classes are only necessary if the maximum number of vocabulary tables (79) has been created. See in Section “Java classes”.

2. New vocabulary table.

New vocabulary table cv2_list_values is created in example (Figure 2). This vocabulary is mapped as **vt30** in */.../sims/hibernate/Middle.hbm.xml* hibernate mapping file. In such a case new hibernate mapping file */.../sims/hibernate/VT30.hbm.xml* is needed:

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
        "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping package="sims.hibernate">
        <class name="VT30" table="cv2_list_values">
                <id name="id" column="sysid">
                        <generator class="native"/>
                </id>
                <property name="visibleName" column="list_value" not-null="true"/>
                <many-to-one name="creator" column="created_by" not-null="true"/>
                <many-to-one name="modifier" column="modified_by"/>
                <property name="createDate" column="created" not-null="true"/>
                <property name="modifDate" column="modified"/>
                <property name="comment" column="commentary"/>
        </class>
</hibernate-mapping>
```

3. New ID influencing table

The mapping file for a table that influences ID generation is very similar to a regular vocabulary table mapping file. Example table cv2_sample_type contains the additional column “abbreviation” that is needed in creating ID. This vocabulary is mapped as **ivt01** in */.../sims/hibernate/Middle.hbm.xml* hibernate mapping file. Corresponding hibernate mapping file is */.../sims/hibernate/IVT01.hbm.xml* that contains:

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
        "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping package="sims.hibernate">
        <class name="IVT01" table="cv2_sample_types">
                <id name="id" column="sysid">
                        <generator class="native"/>
                </id>
```

```
            <property name="visibleName" column="sample_type" not-null="true"/>
            <property name="abbreviation" column="abbreviation" not-null="true"/>
            <many-to-one name="creator" column="created_by" not-null="true"/>
            <many-to-one name="modifier" column="modified_by"/>
            <property name="createDate" column="created" not-null="true"/>
            <property name="modifDate" column="modified"/>
            <property name="comment" column="commentary"/>
        </class>
    </hibernate-mapping>
```

4. **New dependency**

In the example we have a table cv3m_sample_type_sample_subtype that defines which subtype entries correspond to which type entries. Mapping file for it is */.../sims/hibernate/DVT01.hbm.xml* and contains:

```
    <?xml version="1.0"?>
    <!DOCTYPE hibernate-mapping PUBLIC
        "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
    <hibernate-mapping package="sims.hibernate">
        <class name="DVT01" table="cv2m_sample_types_sample_subtypes">
            <id name="id" column="sysid">
                <generator class="native"/>
            </id>
            <many-to-one name="server" column="sample_type" class="IVT01" not-null="true"/>
            <many-to-one name="client" column="sample_subtype" class="VT16" not-null="true"/>
            <many-to-one name="creator" column="created_by" not-null="true"/>
            <many-to-one name="modifier" column="modified_by"/>
            <property name="createDate" column="created" not-null="true"/>
            <property name="modifDate" column="modified"/>
            <property name="comment" column="commentary"/>
        </class>
    </hibernate-mapping>
```

In this example the same table that influences the generated ID also has a dependant table. It is not mandatory. It is possible to have a vocabulary table that has a dependant but does not influence the generated ID and vice versa.

3 Meta-data

Meta-data can be changed from SIMS interface (Admin Tables) or by using DB tables:

a1_person_meta_data, a2_visit_meta_data, a3_sample_meta_data in default configuration.

The structure of meta-data tables have to remain as it is for normal SIMS functioning. However, values have to correspond to desirable SIMS configuration.

In example (Figure 2) attribute *time_of_visit* of table *a2_characteristic* has been changed to *measurement_time* and number of new attributes has been added. In such a case context of meta-data table for Middle level has to be changed (new records are colored with red and bold, changed parts are colored red):

Property name	Short name	Long name	...	Upload name
...				
dt01	Measurement time	Measurement time		Time of measurement
...				
it13	**Integer Value**	**Integer Characteristic**		**Integer Characteristic**
ft27 **st23**	**Float Value** **String Value**	**Float Characteristic** **String Characteristic**		**Float Characteristic** **String Characteristic**
vt30	**List Value**	**List**		**List**
ivt01	**Type**	**Type**		**Type**

Table 3 ***Context of meta-data table***

Values of attributes **short name**, **long name** and **upload name** will be used in SIMS forms.

4 System Constants

There are several constants that need to be specified. These are kept in /conf/systemConst.xml file. The file begins with an opening tag:

```
<SIMS_Constants version="1.17">
```

There are upload names for static fields:

```
<UploadNames>
<UploadName column="visibleName" upload="ID"/>
<UploadName column="comment" upload="COMMENT"/>
<UploadName column="dataFiles" upload="DATA_F"/>
<UploadName column="protocolFiles" upload="SUPP_F"/>
<UploadName column="dataFilesType" upload="DF_TYPE"/>
<UploadName column="protocolFilesType" upload="SF_TYPE"/>
<UploadName column="createDate" upload=""/>
```

```
<UploadName column="creator" upload=""/>
<UploadName column="modifDate" upload=""/>
<UploadName column="modifier" upload=""/>
</UploadNames>
```

This is where you would change the level names of the application. Below each of the names of the three levels of the application are the numbers of each type of fields in that level. You should check that the number of entries in the corresponding level mapping file match the ones provided in this file:

```
<MainNames>
<Bottom plural="Aliquots" singular="Aliquot" prefix="Aliquot."
 string_numb="1" integer_numb="0" float_numb="2" date_numb="2"/>
<Middle plural="Characteristics" singular="Characteristic" prefix="Characteristic."
 string_numb="3" integer_numb="0" float_numb="0" date_numb="1"/>
<Top plural="Persons" singular="Person" prefix="Person."
 string_numb="2" integer_numb="6" float_numb="4" date_numb="5"/>
</MainNames>
```

There is also a list of all the vocabularies. The seq_numb value must correspond to the value of VTxx.hbm.xml file of the corresponding vocabulary. The table and item values are going to be used in the application.

```
<Vocabularies>
 <Vocabulary seq_numb="01" table="Aliquot Concentration units" item="Concentration unit"/>
…
 <Vocabulary seq_numb="29" table="Sample subtypes" item="Sample subtype"/>
 <Vocabulary seq_numb="30" table="Sample transport conditions" item="Sample transport condition"/>
</Vocabularies>
```

The tables that will influence the generation of IDs are configured separately. First there is a list of the table mappings similar to that of vocabularies above.

```
<Identificies>
<Identificy seq_numb="1" table="Sample timings" item="Sample timing"/>
</Identificies>
```

Then the desired structure of the generated ID is defined. The level of the ID in question is defined, static strings (s1, s2) are defined, which table to be used (ivt1), and what counter (numbers or letters) (ptype).

```
<IdStructure>
 <Middle s1="-" ivt1="1" s2="" ptype="1"/>
</IdStructure>
```

Table dependencies are configured as follows. Again, the seq_numb corresponds to the number of DVTxx.hbm.xml file. The name of the dependency table, independent and dependent will be used in the application. And the names of the mapping files of the related vocabularies have to be provided.

In this example the independent table is the same one that participates in ID generation. It is not mandatory; the independent table can be any of the other vocabularies, as linked in the data base.

```
<Dependencies>
<Dependency seq_numb="1" table="Sample type and subtype" independent="Type" dependent="Subtype"
        independent_class="IVT01" dependent_class="VT16"/>
</Dependencies>
```

The file ends with a closing tag.

```
</SIMS_Constants>
```

This file is read by the application only once at startup. For any changes thereafter to take place the application needs to be reloaded by Tomcat server.

5 Java classes

In the current version it is very unlikely that you would have to edit and recompile Java classes. Java classes have to be changed when there is more than maximum number of attributes of some particular type in new configuration. In the current version the maximum number of string, integer, float, date and vocabulary types is 79, maximum number of ID related vocabularies and maximum number of vocabulary dependencies is 9. If your configuration requires more, then you have to edit source files and recompile.

In such case following classes have to be changed:

1. *sims/hibernate/Top.java* (if first level table is changed: *a1_persons* in default configuration),
2. *sims/hibernate/Middle.java* (if second level table is changed: *a2_vists* in default configuration),
3. *sims/hibernate/Bottom.java* (if third level table is changed: *a3_samples* in default configuration)
4. *sims/hibernate/VTxx.java* (if more than 79 vocabularies are configured).

In example (Figure 2) float type is exceeded default maximum **ft27** (maximum is 26) and vocabulary type is exceeded default maximum **vt30** (maximum is 29).

2. Middle.java

Also java class *sims/hibernate/Middle.java* has to be changed (changes/new code is colored with red) and recompiled:

```
package sims.hibernate;
...
...
//---------- Float parameters
    private Double ft01;
    public Double getFt01() {
            return ft01;
          }
          public void setFt01(Double val) {
                  ft01 = val;
          }
          ...
          private Double ft27;
          public Double getFt27() {
                  return ft27;
          }
          public void setFt27(Double val) {
                  ft27 = val;
          }
...
//---------- vocabularies
    private VT01 vt01;
    public VT01 getVt01() {
      return vt01;
    }
    public void setVt01(VT01 val) {
            vt01 = val;
    }
    ...
    private VT30 vt30;
    public VT30 getVt30() {
            return vt30;
    }
    public void setVt30(VT29 val) {
            vt30= val;
    }
...
```

3. VT30.java

New class *sims/hibernate/VT30.java* has to be created and compiled for new vocabulary:

```
package sims.hibernate;
public final class VT30 extends Persistent {
}
```

6 Application forms for record editing/inserting - JSPs

There are six JSPs that have to be changed to complete new SIMS configuration, all found in the *jspf* folder:

1. *AddTopE.jspf* and *AddTopB.jspf* (if first level table is changed: *a1_persons* in default configuration),
2. *AddMiddleE.jspf* and *AddMiddleB.jspf* (if second level table is changed: *a2_vists* in default configuration),
3. *AddBottomE.jspf* and *AddBottomB.jspf* (of third level table is changed: *a3_samples* in default configuration).

When new attributes are added to a level, they have to be added to the edit page too. It is done in the editable part marked by commentary of *Add<level>E.jsp*. In the example (Figure 2) a new float characteristic was added to middle level. The new code to add would look similar to this:

```
…
<!-- ******* The editable part starts here ******* -->
…
<%--Float characteristic:--%>
    <tr>
    <td class="dialh"><%=sims.getLongName("ft27")%></td>
    <td><%=sims.getInput("ft27",10)%> </td>
   </tr>
…
<!-- ******** The editable part ends here ******* -->
…
```

The *Add<level>E.jspf* are used for creating new entries or editing a single entry. *The Add<level>B.jspf* are used for batch edit. The editable part of batch edit files differs from the single edit files with the additional checkbox next to each field. Therefore when adding new attributes, it is convenient to adjust the edit file and then just copy the new parts to the batch file and add the checkbox code next to each entry. The part that differs from the simple edit file is emphasized:

```
…
<%--Float characteristic:--%>
    <tr>
    <td><%=sims.changeMark("ft27")%></td>
    <td class="dialh"><%=sims.getLongName("ft27")%></td>
    <td><%=sims.getInput("ft27",10)%> </td>
   </tr>
…
```

Following methods are used for SIMS edit/insert form creation:

Method	Description	Example
Sims.changeMark	Used for the batch edit files, for all types of attributes	*<%--String characteristic:--%>* *<tr>* *<td><%=**sims.changeMark**("st23")%></td>* *<td class="dialh"><%=**sims.getLongName**("st23")%></td>* *<td><%=sims.getInput("st23",20)%> </td>* *</tr>*
Sims.getLongName	Print LongName from meta-data table. Is used with all types of attributes.	

Sims.getCheckBox	Create check box HTML element, checked if value of attribute is 1 and unchecked if value is 0. Only for IT type.	*<%--Integer characteristic:--%>* *<tr>* *<td class="dialh"><%=sims.getLongName("it13")%></td>* *<td><%=**sims.getCheckBox**("it13") %></td>* *</tr>*
Sims.getPromterVt	Create dropdown-menu HTML element with values from appropriate vocabulary table. Only for VT type.	*<%--Listed Characteristic:--%>* *<tr>* *<td class="dialh"><%=sims.getLongName("vt30")%></td>* *<td><%=**sims.getPrompterVt**(VT30.class,"vt30", false) %></td>* *</tr>*
Sims.getInput	Create input HTML element with value of attribute. For ST, DT, IT and FT types.	*<%--Integer value characteristic:--%>* *<tr>* *<td class="dialh"><%=sims.getLongName("it14")%></td>* *<td><%=**sims.getInput**("it14",10)%></td>* *</tr>*
Sims.getYear	Create input HTML element with year /month /day /hour /minute from a DT attribute.	*<%-- date attribute (yyyy-mm-dd-hh-mm):--%>* *<tr>* *<td class="dialh"><%=sims.getLongName("dt01")%></td>* *<td><%=**sims.getYear**("dt01", true)%><b>-</b><%=**sims.getMonth**("dt01", true)%><b>-</b>* *<%=**sims.getDay**("dt01", true)%><b>-</b><%=**sims.getHour**("dt01", true)%><b>-</b><%=**sims.getMinute**("dt01", true)%></td>* *</tr>*
Sims.getMonth		
Sims.getDay		
sims.getHour		
sims.getMinute		
Sims.getTextArea	Create text-area HTML element.	*<%--string attribute:--%>* *<tr>* *<td class="dialh"><%=sims.getLongName("st24")%></td>* *<td><%=**sims.getTextArea**("st24", 50, 5) %></td>* *</tr>*

Table 4 ***Methods for attributes processing***

The examples in *Table 4* are for a simple edit page all except the first. It is not necessary to list all the available units of a DT attribute; more often just the first three are used. It is also possible to combine several different characteristics:

```
<%--Glucose medication--%>
<tr>
   <td class="dialh"><%=sims.getLongName("it09")%></td>
   <td><%=sims.getCheckBox("it09") %>
          &nbsp;&nbsp;<%=sims.getLongName("vt11")%>
          &nbsp;&nbsp;<%=sims.getPrompterVt(VT11.class,"vt11", false) %>
          &nbsp;&nbsp;<%=sims.getLongName("ft21")%>
          &nbsp;&nbsp;<%=sims.getInput("ft21",10)%>
          &nbsp;&nbsp;<%=sims.getPrompterSt(VT14.class,"st16",false)%>
   </td>
</tr>
```

To properly display dependent vocabularies in a single edit page, the code should look something like this:

```
<%--Sample types/subtypes (Type/Collection metadata):--%>
      <tr>
        <td class="dialh"><%=sims.getLongName("ivt01")%></td>
        <td><select name="ivt01"
              <%if ( sims.usedBy( Bottom.class, "parent", myMiddle)){ %>
              disabled="disabled"<%} %>
          onchange="processMiddleIdentif( this.value, 'ivt01', 'IVT01'); processDependent(
this.value,'vt16','DVT01');">
          <%=sims.getObjectPrompter( IVT01.class, typeId)%>
        </select>
          &nbsp;<%=sims.getDependentPrompter( "vt16", DVT01.class, false,"ivt01",
IVT01.class)%>&nbsp;<%=sims.getInput("st01",20)%></td>
      </tr>
```

Dependant values cannot be edited in batch mode and there is nothing to post in batch edit page.