SDTM-ETL 5.x: MedDRA and WHODrug Coding

Author: Jozef Aerts, XML4Pharma

Last update: **2026-02-15**

MedDRA and WHODrug coding

Does SDTM-ETL provide MedDRA and WHODrug coding?
It is a question that we regularly get …
The answer is: "Yes" and "No".

The reason is that most companies who generate SDTM already have their own system to generate MedDRA codes and names for several variables in CM (Concomitant Medications) and AE (Adverse Events) and for WHODrug names for variables such as CMDECOD and AGDECOD. Furthermore, the licensing restrictions on MedDRA forbid us to redistribute the MedDRA files or derived work (such as MedDRA-XML or MedDRA-JSON).

However: SDTM-ETL already has features that enable to connect to MedDRA and/or WHODrug (or ATC) systems of the users. There are different possibilities:

A. Using RESTful Web Services

SDTM-ETL allows to define functions that use RESTful Web Services. This feature is explained in the tutorial "Using RESTful Web Services". There are already build-in functions for:
- getting LBTESTCD, LBTEST, LBSPEC, LBMETHOD etc. from the LOINC code
- performing unit conversions, also between "US-conventional" and "SI" units and vice versa based on the LOINC code, or the provided molecular weight of the substance.

Users of the software can however add their own functions to use RESTful Web Services by adding short XSLT scripts in the file "functions_restful_web_services.xsl" in the folder "stylesheets"[1].

One of the existing functions in this file is:

```
<xsl:function name="sdtm-etl:loinc2sdtmlb">
    <xsl:param name="loincode"></xsl:param>
    <xsl:param name="varname"></xsl:param>
    <xsl:message>Trying to find value for variable <xsl:value-of select="$varname"/> for LOINC code <xsl:value-of select="$loincode"/></xsl:message>
    <!-- 2022-06-28: try using XML file in CDISC-CT/LOINC2SDTM_cached.xml first -->
    <!--xsl:message>Function loinc2cdisc: LOINC code = <xsl:value-of select="$loincode"/> - SDTM/SEND variable = <xsl:value-of select="$varname"/></xsl:message-->
    <xsl:variable name="xmlfile" select="concat('file:///',$LOINC2SDTMLB_XMLFILELOCATION)"/>
    <xsl:choose>
        <!-- remark the starting directory is "/temp", so we need "../CDISC-CT/LOINC2SDTM_cached.xml -->
        <xsl:when test="document($xmlfile)//LOINC2SDTMMapping[LBLOINC/text()=$loincode] != ''">
            <!--xsl:message>Searching for LOINC <xsl:value-of select="$loincode"/> in cached XML file</xsl:message-->
            <xsl:variable name="CDISCCODE" select="document($xmlfile)/LOINC2SDTMMapping[LBLOINC/text()=$loincode]/*[name()=$varname]/text()"/>
            <xsl:message>Found <xsl:value-of select="$varname"/> = <xsl:value-of select="$CDISCCODE"/> in XML file with cached mappings for LOINC code <xsl:value-of select="$loincode"/>
            </xsl:message>
            <xsl:value-of select="$CDISCCODE"/>
        </xsl:when>
        <xsl:otherwise>
            <xsl:message>Using RESTful web service to find <xsl:value-of select="$varname"/> value for LOINC code <xsl:value-of select="$loincode"/></xsl:message>
            <xsl:variable name="RESTRESPONSE" select="concat('http://www.xml4pharmaserver.com:8080/CDISCCTService2/rest/LOINC2SDTMLB/',$loincode,'.xml')"/>
            <!-- now get the value for the variable name, e.g. LBTESTCD,LBTEST,LBMETHOD,LBANMETH,LBFAST,LBSPEC -->
            <xsl:variable name="CDISCCODE" select="document($RESTRESPONSE)//Response/LOINC2SDTMMapping[1]/*[local-name()=$varname]/text()" />
            <xsl:choose>
                <xsl:when test="$CDISCCODE != ''"><xsl:message>Value for <xsl:value-of select="$varname"/> for LOINC code <xsl:value-of select="$loincode"/> = <xsl:value-of select=
                "$CDISCCODE"/></xsl:message></xsl:when>
                <xsl:otherwise>No value for <xsl:value-of select="$varname"/> for LOINC code <xsl:value-of select="$loincode"/> could be obtained from the RESTful Web Service</xsl:otherwise>
            </xsl:choose>
            <xsl:value-of select="$CDISCCODE"/>
        </xsl:otherwise>
    </xsl:choose>
</xsl:function>
```

[1] The folder name "stylesheets" is somewhat confusing, but we want to keep it for historical reasons.

where the parameter "varname" can be "LBTESTCD", "LBTEST", "LBMETHOD", "LBSPEC" etc..

For example, LBSPEC, the function is then defined by:

```
<xsl:function name="rws:loinc2lbspec">
    <!-- Returns the LBSPEC value from the LOINC code -->
    <xsl:param name="loinccode"/>
    <!-- 2024-08-08: we delegate it to the function sdtm-etl:loinc2sdtmlb passing the SDTM variable name -->
    <xsl:value-of select="sdtm-etl:loinc2sdtmlb($loinccode,'LBSPEC')"/>
</xsl:function>
```

and when used in an SDTM-ETL mapping script:

The Transformation Script

```
# Mapping using ODM element ItemData with ItemOID IT.LOINCODE
# Generalized for all StudyEvents
# Using categorization as a CodeList is associated with the SDTM CodeList
# but no CodeList is associated with the ODM data
$LOINCCODE = xpath(/StudyEventData/FormData[@FormOID='F_LAB']/ItemGroupData[@ItemGroupOID='IG_LB_LOINC']/ItemData[@ItemOID='IT.LOINCODE']/@Value);
$LB.LBSPEC = rws:loinc2lbspec($LOINCCODE) ;
```

Suppose we have a company-specific system for MedDRA mapping. It can be e.g. be based on AI (Artificial Intelligence) or "word similarity"[2]. All that is then needed to have an API defined on top of it that can be used as a RESTful Web Service.
For example, this could be an intranet call to the RESTful Web Service server like:

https://myMedraSystem?term=pain%20in%20the%head

where the part after "?term=" is essentially "pain in the head" (in HTTP requests require that blanks are replaced by "%20".

e.g. returning either as XML (or JSON):

```
<MyMedraSystem inputTerm="pain in the head">
    <PreferredTerm>headache</PreferredTerm>
    <PreferredTermCode>10019211</PreferredTermCode>
    <LowLevelTerm>headache</LowLevelTerm>
    <LowLevelTermCode>10019211</LowLevelTermCode>
    <HighLevelTerm>Headaches NEC</HighLevelTerm>
    <HighLevelTermCode>10019212</HighLevelTermCode>
    ...
</MyMedDRASystem>
```

It is then very easy to add a function in SDTM-ETL as XSLT for e.g. getting the "high level code" as e.g.:

```
<xsl:function name="rws:HLTCodeMedDRA">
    <!-- Returns the High Level Code for the provided term -->
    <xsl:param name="term"/>
    <xsl:variable name="RESTRESPONSE" select="concat('https://MyMedDRASystem?term=',$term)"/>
    <!-- Retrieve the value of the element HighLevelTermCode -->
    <xsl:value-of select="document($RESTRESPONSE)/MyMedDRASystem/HighLevelTermCode"/>
</xsl:function>
```

[2] e.g. based on the "Levenshtein distance".

and used in the mapping script e.g. as:

The Transformation Script

```
1 # Mapping using ODM element ItemData with ItemOID I_AE_TERM
2 $ADVERSETERM = xpath(/StudyEventData[@StudyEventOID='AE']/FormData[@FormOID='F_AE']/ItemGroupData[@I
3 # use the RESTful Web Service function to get the HLT-code
4 $AE.AEHTLCD = rws:HLTCodeMedDRA($ADVERSETERM);
5
```

It's as simple as that …
If the RESTful Web Service is designed to answer JSON, the function is slightly more longer, as the function must then parse JSON, which is also possible in XSLT.
The other possibility is of course to adapt the RESTful Web Service to also return XML transformed from the JSON, e.g. using
https://myMedraSystem?term=pain in the%head&output=xml

Such functions are extremely easy to develop, require however a minimum of XSLT knowledge. In case such XSLT knowledge is lacking in your company, we will of course always be happy to help.

The same essentially applies to WHODrug coding and ATC coding: all one needs to have is one own's system to do the coding with a RESTful Web Service installed on top of it.

The great thing of this methodology is that such additions can be done by the customer: they do not require any change in or update of the software itself: in most cases we at XML4Pharma even do not know about about such additions by our customers.

B. Using mapping files

SDTM-ETL has the possibility to read external files and treat the content. An example is the "SDTM Relationships" from the file "ct-relationships-1-0-sdtmig_3-3_3-4.json" in the folder "CDISC-CT". This also allows integration with propriety MedDRA and WHODrug mapping files. Such an integration does however a "tailoring" of the software for the specific customer.