

Processing text

Getting texts into good shape

DK-CLARIN WP 2.1 Technical Report.
Jørg Asmussen, DSL

Draft version of June 15, 2011¹

Document history

July 2, 2010: First version completed. Covers general procedures and import of Infomedia material.

Outline

1	General procedures	1
2	XML-formatted material	2
2.1	Infomedia	2

This technical report describes how existing text material is converted to the CTB-TEI format specified in [Asmussen et al. \(2011\)](#).

1 General procedures

If the text material to be converted already exists in an XML version, the conversion process will usually be straightforward:

¹The most recent version can be downloaded from:
http://korpus.dsl.dk/clarin/corpus-doc/converting_text.pdf
Additional material such as software documentation and source code is available via
<http://clarin.dsl.dk> (see menu item *Deliverables*).

1. The existing XML-shaped text material is uploaded to the text input queue of the CTB at `/db/ctb/text-input`. This is done by applying the eXist Java client shipped with eXist-db. The upload task should be performed by the DB admin only.²
2. A series of XQuery-based webservices are triggered from within a locally run application³ in order to perform the conversion process proper. Most of the web-services applied are standard (see below), at least one of them must be developed specifically for the particular conversion task in question. The conversion service must generate output in proper CTB-TEI format and place it at its final location in the CTB in a collection beneath `/db/ctb/text`.

The conversion process, indicated as the second step above, is broken down by applying a couple of specifically designed tools implemented as publicly accessible webservices, namely

1. `make-header` that helps constructing a valid CTB-TEI text header, see [Asmussen et al. \(2011\)](#)
2. `make-id` that returns valid CTB text and person IDs, see [Asmussen \(2011b\)](#)
3. `pretokenize` that converts ‘horizontal’ text into the ‘vertical’ format required by CTB-TEI, see [Asmussen \(2011a\)](#)

The CTB text input queue `/db/ctb/text-input` is subdivided into several branches reflecting the organisations responsible for gathering the corpus material, e.g. `dsl.dk` or `dsn.dk`.⁴ For each organisation there again is a sub-collection for each text supplier, e.g. Infomedia, that again may be subdivided into further sub-collections.

2 XML-formatted material

2.1 Infomedia

Infomedia material is delivered in a proprietary XML format. The files are uploaded by means of the GUI client shipped together with eXist-db to the CTB text input queue `/db/ctb/text-input` using the the eXist GUI client shipped with eXist-db.

²Currently Jørg Asmussen.

³WP 2.1 uses the self-developed Java application `RunTextImport`, see [2.1](#) below.

⁴Values for organisation names must be defined in the valueset

http://korpus.dsl.dk/clarin/corpus-doc/text-header/vs_organisationName.xml.

The CTB input queue comprises two branches: `dsn.dk` and `dsl.dk` for material gathered by these two institutions respectively. Below this level, there is one sub-collection for each source, in this case Infomedia.

The Infomedia source material comprises one text collection per month. The import routine imports one month per execution.⁵ From here texts are taken one-by-one, converted to the CTB format, and uploaded to their final destination in the CTB in a sub-collection below `/db/ctb/text`.

The tool that takes care of this process is the Java program `RunTextImport` of the CTB-Manager tool collection.⁶ In order to run the importer, the following variables need to be manually assigned with appropriate values directly in the code of the method `importInfomedia`:

```
String txtInPath DB-path to input collection given as URI,
    e.g. http://ctb.dsl.dk/text-input/dsn.dk/infomedia/2009-12/

String txtOutPath DB-path to output collection given as plain eXist path,
    e.g. /db/ctb/text/clarin.dsn.dk/infomedia/2009

String outputCollectionPrefix Prefix of output collection, given as the
    2-digit number of the month being processed, followed by at full stop
    character, e.g. 12.

String logFileName Path and name of logfile (in XML format) that is written
    on the local computer where the importer is executed,
    e.g. /DOT/infomedia-log/clarin-dsn-infomedia-2009-12.xml

int startTextId Start value of CTB text id series (10-digit number). The
    Infomedia importer does not use the make-id web-service7 in order
    to make the importer run somewhat faster. This means that prior to
    each execution of the importer, the initial text ID must be defined. It
    is then automatically incremented for each text processed, the used
    ID range can be seen from the logfile. It is crucial to ensure that no
    duplicate IDs are issued as there is no automatic check of this!
    An initial Infomedia text ID8 could be 2010551237

long startPersonId Start value of CTB person id series (12-digit number
    given as long integer). The same facts as given for startTextId apply.
    It is assumed that there is only one author of each text. In cases where
```

⁵Depending on the quantity of text material to be processed (figures can be found at <http://clarin.dsl.dk/wiki/clarin/doku.php?id=log>) processing takes around five hours per month.

⁶`RunTextImport` is located in the `dk.dsl.ja.textman` package. The code is available at <http://korpus.dsl.dk/clarin/java/CTB-Manager.zip>. Javadoc is available at <http://korpus.dsl.dk/clarin/javadoc/CTB-Manager>.

⁷<http://ctbws.dsl.dk/registry/make-id>

⁸Text ID series are documented in [Asmussen et al. \(2011\)](#) (see the `textId` variable here).

there are more, they are treated as one (as this is not explicitly indicated in the Infomedia input format).

An initial Infomedia person ID could be 201000551237L (L = long)

`int sizeOfSubcollection` Size of output sub-collections. Output is organized in sub-collections, each named by *outputCollectionPrefix* concatenated with a running 3-digit index (e.g. 07.001, 07.002, 07.003, etc.)

A good size value is 500

After these assignments have been made, the code needs to recompiled.

`RunTextImport` is then executed from a commandline. Once started, the application needs the command `info` to be entered in order to perform the requested Infomedia import.

OBS! The importer **must not** be executed with incorrect values assigned to the above variables. **Severe damage** of DB contents may result!

Further information on the web-services that `RunTextImport` applies, can be seen from its Javadoc and the code itself.⁹

⁹The specifically developed Infomedia importer web-service is located at <http://ctbws.dsl.dk/convert/infomedia.xq> and can be downloaded from <http://ctbws.dsl.dk/convert/infomedia.xq.zip>.

Bibliography

Asmussen, J. (2011a). CTB web-services. Technical report, DK-CLARIN, korpus.dsl.dk/clarin/corpus-doc/ctb-webservices.pdf.

Asmussen, J. (2011b). Text formatting. Technical report, DK-CLARIN, korpus.dsl.dk/clarin/corpus-doc/text-format.pdf.

Asmussen, J. et al. (2011). Text metadata. Technical report, DK-CLARIN, korpus.dsl.dk/clarin/corpus-doc/text-header.pdf.