# TBX-Basic Implementation Guide

Validation – Import – Export

## Contents

# Introduction

TBX-Basic is intended to be the major localisation industry dialect for terminology exchange. It can be used to handle monolingual, bilingual, or multilingual glossaries or termbases. It is composed of three modules: Core, Min, and Basic. The Core module consists of the core structure which all valid TBX files share and which corresponds with the description of the TBX Core as found in ISO 30042. The combination of the data categories added to the Core by the Min and Basic modules constitutes the TBX-Basic dialect. This implementation guide is provided by LTAC Global as a public service. If you want to implement TBX-Basic, you are free to use it. If you want to create your own TBX dialect, you must obtain appropriate access to a purchased copy of the new version of ISO 30042 DIS (to be published in 2018).

[In a future draft, a basic introduction to TBX will be included.]

Implementing TBX-Basic in a translation tool that includes terminology management means supporting import and export of valid TBX-Basic documents instances

## Data Category as Attribute (DCA)

DCA is the "traditional" TBX style in which data categories are indicated as values of the @type attribute of one of the following classification elements: admin, adminNote, descrip, descripNote, ref, transac, transacNote, termNote, xref.

Example:

```
<descrip type="subjectField">finance</descrip>
```

The classification elements are all included in the Core module. This is the reason DCA only needs a single namespace. By default, any value of @type is permitted in the Core module, but when used in a dialect, the permitted values are restricted to only those data categories included by the modules which define that dialect.

## Data Category as Tag (DCT)

DCT is a new style of TBX which attempts to bring TBX more in line with modern XML practices. In DCT, data categories are indicated as XML tag names (specifically, generic identifiers). Due to this, there are as many possible tag names in DCT as there are data categories. Therefore, for the sake of simplicity, all conceivable DCT tags are not included in the Core namespace. Instead, each data category module has its own namespace for use with DCT. For example, TBX-Basic includes the Basic module, which has the following namespace: http://www.tbxinfo.net/ns/basic. Using this namespace solution, only the data categories which are included in the Basic module need to have tags defined for them (in the Basic namespace). These tags can then be introduced into a TBX document instance by calling upon the Basic namespace.

Example:

```
<subjectField xmlns="http://www.tbxinfo.net/ns/basic">
    …
</subjectField>
```

It is possible in an XML file for elements to declare their namespace in their start tag (as shown above). It is also possible (and the recommended practice in TBX) to assign a namespace prefix earlier on in the root element:

```
<tbx … xmlns:basic="http://www.tbxinfo.net/ns/basic">
    …
    <basic:subjectField>…</basic:subjectField>
    …
</tbx>
```

Therefore, a valid TBX-Basic file in DCT will have namespace declarations for each of the modules which are included in the TBX-Basic dialect definition (which can be found using either the TBXinfo website or the Validation API):

- Core – `urn:iso:std:iso:30042:ed:3.0`
- Min – `http://www.tbxinfo.net/ns/min`
- Basic – `http://www.tbxinfo.net/ns/basic`

These namespace declarations should most often be found in the root <tbx> element, but may in some files appear in the elements to which they apply.

Also, to preserve isomorphism between DCT and DCA, DCT elements may use the optional @metaType attribute to retain the classification element:

```
<basic:subjectField metaType="descrip">…</basic:subjectField>
```

## Validation

This section details the steps which must be taken to ensure an XML document instance (hereafter described as the "*candidate*"), which the *candidate* claims to be TBX-Basic compliant is actually a valid instance of the TBX-Basic dialect. If at any step the *candidate* fails a test, it is not valid TBX-Basic.

If the *candidate* does not even claim to be compliant with TBX-Basic (according to the type attribute on the root), use the Validation API or TBXinfo.net[1] to attempt to learn which modules the *candidate's* stated dialect is supposed to contain.

### TBX-Basic Validation Steps (First steps, both styles):

TBX-Basic compliance ultimately needs to be validated with schemas (see Validation via Schema). However, the following 8 steps are a human targeted guidance that suggests what simple (of gradually increasing complexity) manual checks can be performed before submitting a candidate instance to actual schema validation. The schema validation covers all the human targeting suggested validation steps, yet performing the steps will provide a very useful guidance for a human validator.

1. Check whether *candidate* looks like XML.
2. Ensure the *candidate* is well-formed XML.

[1] http://www.tbxinfo.net/ [note: for now, consult the development website: http://tbxinfo-dev.byuling.net/]

3. The root start-tag should declare the default namespace as that of the Core module: `urn:iso:std:iso:30042:ed:3.0`
    a. Example:
    <tbx type="TBX-Basic" style="dca" xml:lang="en" xmlns="urn:iso:std:iso:30042:ed:3.0">
4. The root start-tag must contain a @type attribute, the value of which must be "TBX-Basic" (case-sensitive).
    a. Example:
    <tbx type="TBX-Basic" style="dca" xml:lang="en" xmlns="urn:iso:std:iso:30042:ed:3.0">
5. The root start-tag must contain a @style attribute, the value of which must be either "dca" or "dct" (case-sensitive). The next step of validation depends on the claimed style of TBX.
    a. Example:
    <tbx type="TBX-Basic" style="dca" xml:lang="en" xmlns="urn:iso:std:iso:30042:ed:3.0">
6. Use TBXinfo.net or the Validation API to find out which modules are included in TBX-Basic. As mentioned earlier, these are: Core, Min, and Basic.
    a. Data categories which are not included by these 3 modules must not be present in the *candidate*.
    b. Content of each data category must agree with the permissible values as described by these modules. Ex: /partOfSpeech/ must have one of these values (as defined in the Min module): "noun", "verb", "adjective", "adverb", or "other".

The next steps will depend on the style of the TBX document, whether it is DCA or DCT.

### Validation Continued (DCA):

7. There can be no elements from namespaces other than that of the Core module in TBX-Basic DCA style.
8. DCT style elements must not be present in a DCA style document instance.

### Validation Continued (DCT):

7. There can be no elements from namespaces other than that of the following modules in TBX-Basic DCT style: Core, Min, and Basic.
8. DCA style data categories must not be present in a DCT style document instance.

## Validation via Schema

The schemas discussed below can be used to address all the above steps with any off-the-shelf XML validator that supports the RelaxNG (RNG), Schematron (SCH), and (for DCT) NVDL languages (or whatever languages your equivalent schemas are written in). If you are not using these schemas, but are using equivalents, and a discrepancy should arise, the schemas here shall be considered authoritative.

**IMPORTANT NOTE: In practice (in software such as Oxygen), it is possible to simply point to the URL of these schemas (as found in the footnotes) directly from a TBX file, bypassing the need for a local file entirely. For DCT, it is only necessary point to the NVDL schema, as it ties the other DCT schemas together. If validation performance is slower than expected, downloading these schemas and pointing to them locally may be preferable.**

**Note: For some validation applications, such as a tool which helps design a TBX dialect, it is permissible to use the RNG for the Core module (see Schemas Needed for DCT style TBX for link) for partial validation. It will show whether a TBX file adheres to the core structure of TBX. However, this partial validation is NOT considered acceptable for Import/Export routines or any other validation routine which needs to validate a specific instance of a TBX dialect.**

## Schemas Needed for DCA style TBX

The following resources are needed to validate a TBX-Basic file of DCA style:

- TBX-Basic integrated RNG (or equivalent).[2]
    - The TBX-Basic RNG schema, found on TBXinfo.net is a modified version of the Core module RNG schema (see DCT section for link).
        - The Core module RNG schema contains "extension points" for DCA. These are the <define> elements with @name of "[classification].types":
            - **Example**: admin.types, adminNote.types, descrip.types, etc.
            - In the Core module RNG, these are all located toward the end of the document.
        - These extension points are used by replacing the <text/> element (in the <define> elements above) with either:
            - A list of permissible types (these are the data category names)
                - Example:

```
<value>subjectField</value>

<value>definition</value>
```

            - <empty/> (if no data categories for that classification are permitted)
                - Example: In TBX-Basic, <text/> has been replaced with <empty/> for descripNote.types, because TBX-Basic has no data categories of classification "descripNote".
        - More detailed examples of using these extension points are on TBXInfo.net and can also be seen by comparing the TBX-Basic DCA integrated RNG schema with the Core module RNG.
- TBX-Basic DCA style SCH (or equivalent).[3]

## Schemas Needed for DCT style TBX

The following resources are needed to validate a TBX-Basic file of DCT Style:

- Core module RNG (or equivalent)[4]
- Min module RNG (or equivalent)[5]
- Min module SCH (or equivalent)[6]

---

[2] https://raw.githubusercontent.com/LTAC-Global/TBX-Basic_dialect/master/DCA/TBXcoreStructV03_TBX-Basic_integrated.rng

[3] https://raw.githubusercontent.com/LTAC-Global/TBX-Basic_dialect/master/DCA/TBX-Basic_DCA.sch

[4] https://raw.githubusercontent.com/LTAC-Global/TBX_Core_RNG/master/TBXcoreStructV03.rng

[5] https://raw.githubusercontent.com/LTAC-Global/TBX_min_module/master/Min.rng

[6] https://raw.githubusercontent.com/LTAC-Global/TBX_min_module/master/Min.sch

- Basic module RNG (or equivalent)[7]
- Basic module SCH (or equivalent)[8]
- TBX-Basic DCT style SCH (or equivalent)[9]
- TBX-Basic NVDL (or equivalent)[10]

### Validation API

A simple TBX validation RESTful API has been created and can be accessed at http://validate.tbxinfo.net/. It can be used to programmatically get the absolute URLs to the various schemas of dialects and modules, which can then be downloaded and parsed (or pointed to directly). It can also be used to get a list of available modules, or the list of modules which are used to define a dialect. The response from the API is in JSON format. See Appendix I: API Example for TBX-Basic for a look at example output from the API along with instructions on how to use the JSON response.

## Import

This section details the process of importing a TBX file. This guide does not cover how to deal with duplicate entries (*i.e.*, same term as an entry already in the database, which may or may not designate the same concept as the existing entry).

1. Validate the file to be imported according to the instructions from the Validation section.
    a. If the file fails validation, it should be rejected.
    b. Providing a report of some kind indicating the reason for failure is recommended.
2. Using the Validation API or TBXinfo.net, retrieve the list of modules for the dialect of the file (TBX-Basic will be used here for demonstration).
    a. Using either the prose definition, the RNG and SCH schemas (or equivalent), or the TBXMD file for each module, you can learn exactly which data categories to expect on import. The TBXMD file would be the simplest to parse to learn this information. The description of the TBXMD file can be found on the TBXinfo.net.[11]
    b. For TBX-Basic these modules will be: Core, Min, and Basic.
3. If any module is unsupported by the importing software, you may:
    a. omit them with a message that the data categories from said module are unsupported;
    b. convert them to /note/ data categories via a converter.
        i. It is recommended that the original data category name be preserved as a part of the note, so that conversion back to the appropriate data category at some future point is made possible.
4. Any data categories from modules which are supported may be imported into the termbase normally.

## Export

This section details the process of exporting a file instance of a certain dialect (TBX-Basic in this case).

1. Select the dialect to which you will be exporting. (TBX-Basic)
    a. **Note: Generic exports of "TBX" files without a clearly stated dialect are not valid.**

[7] https://raw.githubusercontent.com/LTAC-Global/TBX_basic_module/master/Basic.rng
[8] https://raw.githubusercontent.com/LTAC-Global/TBX_basic_module/master/Basic.sch
[9] https://raw.githubusercontent.com/LTAC-Global/TBX-Basic_dialect/master/DCT/TBX-Basic_DCT.sch
[10] https://raw.githubusercontent.com/LTAC-Global/TBX-Basic_dialect/master/DCT/TBX-Basic.nvdl
[11] http://tbxinfo-dev.byuling.net/wp-content/uploads/2017/11/TBX%20Module%20Description.pdf

2. Decide which style of TBX the export will be (DCA or DCT).
    a. Mixing styles in a single TBX document instance is not permissible.
3. Use the Validation API or TBXinfo.net to retrieve the list of modules included by the target dialect.
    a. TBX-Basic includes: Core, Min, and Basic
    b. Note that if DCT is used, each of these modules will use their own namespaces.
4. If the data model of the exporting software already maps directly to the data categories (including permissible content values) included by the modules, a simple export should suffice.
    a. It is permissible to export subsets of the terminological data contained in the exporting software.
5. If the data model of the exporting software does not directly map to the data categories included by the module, a manual mapping wizard may be necessary.
    a. If only some of the modules of the target dialect are supported, the unsupported data categories may either be converted to /note/s or omitted (preferably with a report to the user in either case).
6. In either situation, the output TBX file must use the data category names which are defined by the modules for that dialect and not the internal data model names for said data categories.
    a. Example: Internally, the software may call /subjectField/ “domain”, but on export it must be converted to “subjectField”.
    b. This same rule applies to defined picklist values. If the software uses the abbreviated /partOfSpeech/ content value “adj” internally, on export, it must be changed to match whatever values are declared in the module definition. In the case of TBX-Basic, “adj” would need to be changed to “adjective”.
7. The export file must be a well-formed XML file with a “.tbx” file extension
8. The export file must also be valid instance of the target dialect (TBX-Basic) according to the schemas of that dialect and style (DCA or DCT).

# Appendix I: API Example for TBX-Basic

## Using the response:

For DCA, only the schemas pointed to by “dca_rng” and “dca_sch” are needed. For DCT, the schemas pointed to by “dct_nvdl”, “dct_sch”, and each of the “rng” and “sch” schemas for each module of the “modules” list. As mentioned before in the DCT section, it is only necessary in practice to use the “dct_nvdl” schema link, since by default, it points to the absolute URLs of the other schemas needed in DCT.

## Example call to API (GET) and response (JSON):

*GET Request:*

URL: http://validate.tbxinfo.net/dialects/TBX-Basic

*JSON Response:*


```
[{
    "name": "TBX-Basic",
    "definition": "https://github.com/LTAC-Global/TBX-
Basic_dialect/blob/master/TBX-Basic%20Definition.pdf",
    "dca_rng": "https://raw.githubusercontent.com/LTAC-Global/TBX-
Basic_dialect/master/DCA/TBXcoreStructV03_TBX-Basic_integrated.rng",
    "dca_sch": "https://raw.githubusercontent.com/LTAC-Global/TBX-
Basic_dialect/master/DCA/TBX-Basic_DCA.sch",
    "dct_nvdl": "https://raw.githubusercontent.com/LTAC-Global/TBX-
Basic_dialect/master/DCT/TBX-Basic.nvdl",
    "dct_sch": "https://raw.githubusercontent.com/LTAC-Global/TBX-
Basic_dialect/master/DCT/TBX-Basic_DCT.sch",
    "id": 2,
    "modules": [
        {
            "name": "Min",
            "definition": "https://github.com/LTAC-
Global/TBX_min_module/blob/master/Min%20Module%20Definition.pdf",
            "rng": "https://raw.githubusercontent.com/LTAC-
Global/TBX_min_module/master/Min.rng",
            "sch": "https://raw.githubusercontent.com/LTAC-
Global/TBX_min_module/master/Min.sch",
            "tbxmd": "https://raw.githubusercontent.com/LTAC-
Global/TBX_min_module/master/Min.tbxmd",
            "id": 1
        },
        {
            "name": "Basic",
            "definition": "https://github.com/LTAC-
Global/TBX_basic_module/blob/master/Basic%20Module%20Definition.pdf",
            "rng": "https://raw.githubusercontent.com/LTAC-
Global/TBX_basic_module/master/Basic.rng",
            "sch": "https://raw.githubusercontent.com/LTAC-
Global/TBX_basic_module/master/Basic.sch",
            "tbxmd": "https://raw.githubusercontent.com/LTAC-
Global/TBX_basic_module/master/Basic.tbxmd",
            "id": 2
        },
```

```
        {
            "name": "Core",
            "definition": "",
            "rng": "https://raw.githubusercontent.com/LTAC-
Global/TBX_Core_RNG/master/TBXcoreStructV03.rng",
            "sch": "",
            "tbxmd": "",
            "id": 4
        }
    ]
}]
```