ST.96 - ANNEX VI

TRANSFORMATION RULES AND GUIDELINES

Version 7.0

Revision approved by the XML4IP Task Force of the Committee of WIPO Standards (CWS) on April 3, 2023

Table of Contents

ST.96 - ANNEX VI 1

1. Introduction 2
1.1 Overview 2
1.2 Scope 2
1.3 How to use this document 2
1.4 Terminology 2
1.5 Rule Identifiers 2

2. Guidance for Data Preparation 3

3. Guidance for Mapping Tables 3

4. Guidance for Data Conversion 4
4.1 Date and Time Formats 4
4.2 Boolean Values 5
4.3 Language and country codes 5
4.4 Code and Enumeration Values 5
4.5 Identity constraints 7
4.6 Different data structures 7
4.7 Type Mismatches 9

5. References 9
WIPO Standards 9
Industry Standards 9

***APPENDIXES** 10*

***APPENDIX A**: ELEMENT AND ATTRIBUTE MAPPING 10*

***APPENDIX B**: ENUMERATION LIST MAPPING 10*

***APPENDIX C**: SAMPLE XSLT CODES 11*

1. INTRODUCTION

1.1 Overview

1. Before the adoption of WIPO Standard ST.96, WIPO Standards ST.36, ST.66 and ST.86 had already been used by Intellectual Property Offices (IPOs); therefore, maintaining transformability with XML document instances conforming to WIPO Standards ST.36, ST.66 and ST.86 (hereinafter called "existing XML Standards") is one of the primary concerns for WIPO Standard ST.96.

2. For the facilitation of data exchange and interoperability between an IPO using ST.36/ST.66/ST.86 and an IPO using ST.96, bi-directional transformation is desirable. However, it is not realistic to expect perfect bi-directional conversion between instances conforming to ST.96 and instances conforming to ST.36/ST.66/ST.86. Due to improvements based on experience and advances in technology, ST.96 structures will differ in many ways from those defined in ST.36/ST.66/ST.86; therefore, this document aims at defining the necessary degree of transformability between ST.96 and ST.36, ST.66 or ST.86. Moreover, it is noted that bi-directional transformation would be determined on a case-by-case basis.

1.2 Scope

3. This document is intended to provide rules and guidelines for transformations between XML instances of ST.96 and instances of WIPO Standards ST.36, ST.66 or ST.86. It does not address either transformations for national implementations or transformation of XML instances of different versions of ST.96.

4. This document includes mapping tables for elements and attributes defined in the Standards in Appendix A of this document and mapping tables for enumerated values and codes specified in the Standards in Appendix B. The mapping tables will be updated in accordance with the evolution of the Standards.

5. This document also provides some examples of eXtensible Stylesheet Language Transformations (XSLT) in Appendix C of this document based on the mapping tables.

1.3 How to use this document

6. This document is intended to provide transformation guidance for IPOs that convert their data conforming to ST.36, ST.66 or ST.86 to data conforming ST.96 and *vice versa*.

1.4 Terminology

7. In this document, the following terms are used:

- "*data transformation*" refers to converting data from a source data format into a destination data format. It can be divided into two steps: data mapping and code generation;
- "*data mapping*" refers to mapping elements/attributes and codes/enumeration values from the source to the destination; and describing any transformation that must occur. The element/attribute mapping is provided in Appendix A of this document. The code/enumeration list mapping is provided in Appendix B of this document;
- "*code generation*" refers to creating transformations in XSLT based on an element mapping specification. Sample XSLT code is provided in Appendix C to this document;
- "*input XML instance*" is the XML instance that will be transformed; and
- "*output XML instance*" is the XML instance that is the result of transformation.

1.5 Rule Identifiers

8. All transformation rules are informative. Transformation rules are identified through a prefix of [TR-nn]. The value "nn" indicates the sequential number of the rule. For example, the rule identifier [TR-06] identifies the sixth transformation rule (TR).

2. GUIDANCE FOR DATA PREPARATION

9. It is expected that an IPO's XML instances are valid against its implementation XML DTD or schema. The implementation XML schema or DTD may define office-specific elements, attributes, types and namespace. In order to utilize the data transformation that converts the XML instances conforming to ST.36, ST.66 or ST.86 to instances conforming to ST.96 and *vice versa,* the Office's XML instances are likely to need to be modified for validation against the corresponding WIPO Standard.

10. In order to prepare valid instances,

- if an IPO uses its own component names instead of using names as defined in the standard, e.g., `WOApplicationBody`, for data preparation, the Office component names in the instances should be renamed to the corresponding component names defined in the Standard; and

- if no namespace is declared in the instances, the Office should add the namespace declaration in the instances as defined in corresponding Standard.

[TR-01] An Input XML instance should validate against the corresponding XML DTD (xx-patent-document.dtd defined in ST.36) or Schema (ST.66, ST.86 and ST.96) of WIPO Standards.

3. GUIDANCE FOR MAPPING TABLES

11. The mapping tables are vital to this document. The goal was to specify a one-to-one mapping between each of the elements and attributes of ST.96 and each of the elements and attributes of ST.36, ST.66 and ST.86. This will not be always feasible as explained in Section 1.1. The mapping tables have been created in Appendix A to this document.

12. For each mapping direction (i.e. ST.36 to ST.96, ST.66 to ST.96, ST.86 to ST.96, ST.96 to ST.36, ST.96 to ST.66 and ST.96 to ST.86), a different file is defined. Each file contains two sections, as Appendix A and Appendix B to this document.

13. The following columns exist in each mapping table:

- Input Node
- Output Node
- Type [Cardinality] for Input and Output nodes
- Condition

14. The *Input Node* and *Output Node* column define the atomic element or attribute. For ST.96 components, tag names include namespace prefix. For that reason, the full path for elements or attributes is used. Hierarchical levels represent the path using XPATH notation. The elements are listed in the order of declaration in the corresponding DTD or Schema. For XPath notation, a slash "/" is used to separate the hierarchical levels.

15. The *Type [Cardinality]* column designates:

- the type used by elements or attributes. As ST.36 is defined using DTDs, only the following types are used: `ID`, `CDATA` and `#PCDATA`. For ST.66, ST.86 and ST.96, W3C built-in data-types and user defined data Types are shown in this column. For ST.96 components and other referenced external standards, data Types include the namespace prefix, and;

- the cardinality of elements or attributes:

  - 1..1 = mandatory only one occurrence
  - 0..1 = optional only one occurrence
  - 1..n(∞) = mandatory one or many
  - 0..n(∞) = optional or many

16. The *Condition* column contains specific mapping or conversion instructions/rules relating to the source and target described in Section "Guidance for Data Conversion".

Input Node	Type [Cardinality]	Condition	Output Node	Type [Cardinality]
ReportCitation/CitedReference/@id	xsd:token [0..1]	TR-23	citation/@id	ID [0..1]

Example of Mapping table

[TR-02] A mapping table SHOULD be developed for each Document level schema defined in ST.96, e.g., ApplicationBody_V6_0.xsd, to facilitate data transformation.

[TR-03] Bi-directional transformations SHOULD ensure that the integrity of the data is maintained and that limitations are fully described and supported by IPOs (e.g. in case some data is lost during the conversion).

4. GUIDANCE FOR DATA CONVERSION

17. In addition to the tables for one-to-one mappings, guidance for data conversion is necessary since data formats used by elements or attributes and their content structures in ST.96 may differ from the corresponding elements or attributes in ST.36, ST.66 or ST.86. In ST.96 many elements and attributes originating from ST.36, ST.66 and/or ST.86 have been redefined by simplifying the data structure or using new XML technologies including a data-oriented design approach. The following issues should be considered when converting XML instances and necessary guidance is provided to address the issues below. Details concerning these issues are provided in the subsections that follow:

- Date and time formats;
- Boolean values;
- Language and country codes;
- Code and enumeration values;
- Identity constraints;
- Different data structures; and
- Type mismatches.

4.1 Date and Time Formats

18. An ST.36 DTD does not specify a strict data format for dates because of DTD limitations even though the date format of "YYYYMMDD" is recommended. In ST.36, date is defined as #PCDATA even though the format of day/month/year is recommended for some kind of dates, e.g. date of mailing or priority date. In ST.66, ST.86 and ST.96, dates are defined as xsd:date which requires the format of YYYY-MM-DD.

19. All of the date fields using a format in which day, month and year information are mandatory can be expressed in a format that is appropriate for ST.96. In such case, the following conversion rules should apply:

[TR-04] To convert to ST.96, the date value should be copied after conversion from the date field of ST.36 into the field of ST.96.

For example, Input in ST.36: date field: 20081025; date format: YYYYMMDD; Output in ST.96: date field 2008-10-25

[TR-05] To convert to ST.36, the date value should be copied from the field of ST.96 into the date field of ST.36 but without the dashes.

For example, Input in ST.96 date value: 2008-10-25, and Output in ST.36: date field: 20081025.

20. ST.36 does not specify a strict format for time fields because of DTD limitations. In ST.36, time is defined as `#PCDATA` even though the format of HHMM is recommended. In ST.66, ST.86 and ST.96, time fields are defined as `xsd:time` which requires the format of hh:mm:ss.

21. All of the time fields using a format in which hour, minute and second information is mandatory can be expressed in a format that is appropriate for ST.96. In such cases, the following conversion rules should apply:

[TR-06] To convert to ST.96, the ST.36 time value should be copied, after conversion, into the time value of ST.96.

For example, Input in ST.36: `time value 1030; time format: HHMM`; Output in ST.96: `time value 10:30:00`

[TR-07] To convert to ST.36, the ST.96 time value should be copied into the time value of ST.36 but without colons or seconds.

For example, Input in ST.96: `time value: 10:30:00`, and Output in ST.36: `time value: 1030`.

4.2 Boolean Values

22. ST.36 does not specify a strict format for Boolean values because of DTD limitations. It is assumed that each IPO has a specific business practice to consistently define the Boolean format in its XML instances. The following values, '0' / '1', 'no' / 'yes', or 'false' / 'true', are commonly used as Boolean values in ST.36 XML instances. In ST.96, 'false' / 'true' Boolean values are used. For the conversion of Boolean values between ST.36 and ST.96 instances, the following rules should apply:

[TR-08] To convert to ST.36, a Boolean value should be copied from the ST.96 field into the Boolean field of ST.36. If necessary, IPOs should adapt the values according to their practices.

[TR-09] To convert to ST.96, a Boolean value from the ST.36 field should be converted to either 'true' or 'false' and copied into the Boolean field of ST.96.

23. In order to facilitate the conversion of Boolean values, the sample XSLT code in Appendix C includes a conversion scheme so that the following common values: '0' / '1', 'no' / 'yes', or 'false' / 'true' in an ST.36 XML instance are converted to 'true' or 'false' in ST.96. If an ST.36 instance contains the other values, IPO should convert the values to one of common values and run the XSLT script.

4.3 Language and country codes

24. In ST.36, the language element/attribute is defined as alpha-numeric (`#PCDATA`), even though ISO two-letter codes are recommended in the description of the corresponding element/attribute. Therefore, if ISO two-letter language codes are not used in ST.36 XML instances, the data of language code in the instances should be converted into corresponding two-letter code before transformation.

25. In ST.36, the country element/attribute is defined as alpha-numeric (#PCDATA), even though ISO two-letter country codes are recommended in the description of corresponding element/attribute. While both standards use ISO Country codes, in ST.36, lower case country codes are allowed, while in ST.96 upper case codes are mandatory; therefore the codes' case should be changed to match that required by each standard during transformation.

[TR-10] To convert to ST.96, the ST.36 field value for country or language should be case-transformed to the corresponding values defined in ST.96, and the result code should be inserted into the ST.96 field.

[TR-11] To convert to ST.36, the ST.96 code value for country or language should be inserted into the ST.36 field.

4.4 Code and Enumeration Values

26. ST.36, ST.66, ST.86 and ST.96 provide sets of codes and enumerated values. In many cases, ST.96 has the same code or enumerated values as defined in ST.36, ST.66 or ST.86. In other cases, however, ST.96 defines additional or different codes/ enumerated values from the other Standards, in order to reflect IPOs' practice and/or follow the ST.96 DRCs. Guidance is provided on the conversion of the codes, depending on whether ST.96 supports the same codes or whether additional code values have been introduced.

27. Some ST.36 fields are defined as `#PCDATA` rather than as a list of specific enumeration values whereas in ST.96, an enumerated list or code is used in most cases. In ST.66/ST.86, some types are defined as a union of a free format and an enumerated list.

[TR-12] To convert to ST.96, the ST.66/ST.86 field value should be mapped to the list of ST.96 codes, and the mapped code should be inserted into the ST.96 field. The free format value should be ignored unless it is possible to map it to an ST.96 code.

[TR-13] To convert to ST.96, the ST.36 field value should be ignored unless it is possible to map it to an ST.96 code. If the value does not conform to the expected data type in ST.96, it may have to be reformatted so that the instance will validate successfully.

[TR-14] To convert to ST.36/ST66/ST86, the ST.96 code value should be mapped to the field of corresponding Standard.

WARNING: When copying values from ST.36, ST.66 or ST.86 to ST.96 (or the reverse), or when revising ST.36, ST.66 or ST.86 to incorporate values from ST.96, consider the possibility that a newly added value might violate a business rule in the new context.

Same Set of Values

28. In most cases, ST.36/66/86 and ST.96 expect the same set of codes or enumerated values for some specific elements or attributes listed in Appendix B of this document.

For example:

ST.66 component	ST.66 allowed values	ST.96 component	ST.96 allowed values
MarkKind	Individual	MarkKind	Individual
	Collective		Collective
	Certificate		Certificate
	Guarantee		Guarantee
	Defensive		Defensive
	Other		Other

29. In this case, the following recommendation applies:

[TR-15] To convert to ST.96, the field value should be copied from the ST.36/66/86 field into the ST.96 field. The formatting of the value should follow the conventions specified in the ST.96 DRCs.

[TR-16] To convert to ST.36/66/86, the field value should be copied from the ST.96 field into the ST.36/66/86 field. The formatting of the value should follow the conventions specified in ST.36, ST.66 or ST.86.

30. In the case that the elements or attributes are required in ST.96 and optional in ST.36, ST.66 and ST.86, the following conversion rules below apply:

[TR-17] If the ST.36/ST.66/ST.86 component is populated, the field value should be copied from the ST.36/ST.66/ST.86 field into the ST.96 field. The formatting of the value should follow the conventions specified in the ST.96 DRCs.

[TR-18] If the ST.36/ST.66/ST.86 component is not populated, the ST.96 component should be populated with the “Undefined” value.

31. There is no case where the element or attribute required in ST.36, ST.66 and ST.86 is defined as optional in ST.96; no rule, therefore, is specified in this document for this circumstance.

Different Set of Values

32. In some cases listed in Appendix B of this document, ST.96 defines different codes or enumerated values from ST.36, ST.66 or ST.86, but the codes or values have the same meaning. For example:

ST.36 component	ST.36 allowed values	ST.96 component	ST.96 allowed values
Orient	Port	orientation	Portrait
	Land		Landscape

33. In some cases, ST.96 defines more codes or values than in ST.36/ST.66/ST.86. In these cases, transformation from ST.36/ST.66/ST.86 to ST.96 is not an issue. However, transformation from ST.96 to ST.36/ST.66/ST.86 is not ensured. In order to guarantee the transformation from ST.96 to ST.36/ST.66/ST.86, the existing XML Standards should be revised to capture the additional codes or values defined in ST.96. For example:

ST.36 component	ST.36 allowed values	ST.96 component	ST.96 allowed values
Color	color	ColourMode	Colour
	Bw		Black and white
	-		Greyscale

34. In general, an ST.96 component does not define fewer values than the corresponding components defined in ST.36, ST.66 or ST.86. For this reason, rules regarding this case are not provided in this document.

[TR-19] To convert to ST.96, a value should be copied from the ST.36, ST.66 or ST.86 field into the ST.96 field, if the both values have the same meaning.

[TR-20] To convert to ST.36/ST.66/ST.86, a value should be copied from the ST.96 field into the ST.36/ ST.66/ST.86 field, if the both values have the same meaning.

[TR-21] If the corresponding value does not exist in destination standard, the value should be copied unchanged in to the output instance even though the value is invalid against the destination standard schema or DTD.

4.5 Identity constraints

35. ST.96 recommends using `xsd:key`/`xsd:unique`/`xsd:keyref` and/or `xsd:ID`/`xsd:IDREF`/`xsd:IDREFS` for identity constraints while ST.36 uses ID/IDREF/IDREFS. Some attributes in ST.36 employ the ID and IDREF/IDREFS types: For example, citation/id uses ID type. The values starting with a numeric character are not allowed in `xsd:ID`/ `xsd:IDREF`/`xsd:IDREFS` field.

[TR-22] To convert to ST.96, the value of the ST.36/ST.66/ST.86 field associated with `xsd:ID`/`xsd:IDREF`/`xsd:IDREFs` type should be copied to the corresponding ST.96 field.

[TR-23] To convert to ST.36/ST.66/ST.86, the `xsd:ID`/`xsd:IDREF`/`xsd:IDREFS` value of the ST.96 field should be copied to the corresponding ST.36/ ST.66/ ST.86 field. The `xsd:key`/`xsd:unique`/`xsd:keyref` value of the ST.96 field should be copied to the ST.36/ ST.66/ ST.86 field ensuring no duplication of ID values in the instance and revising any corresponding IDREF and IDREFS. The `xsd:key`/`xsd:unique`/`xsd:keyref` values starting with a numeric character should be converted to non-numeric character values..

4.6 Different data structures

36. ST.96 provides different structures for some components defined in ST.36, ST.66 or ST.86. Guidance is provided on conversion from one structure to the other. Three major changes can occur in this context: deletion, addition, and renaming.

Deletion

37. Some elements or attributes defined in ST.36, ST.66 or ST.86 have been deleted in ST.96 as they are no longer used. They are marked as “NOT USED” in Appendix A of this document. For example, the following elements

`citation`, `nplcit`, `article`, `book` and `text` are no longer used and therefore have no corresponding element in ST.96.

[TR-24] To convert to ST.96, the ST.36/ST.66/ST.86 field should be ignored as there is no counterpart component in ST.96.

[TR-25] To convert to ST.36/ST.66/ST.86 the field should not be provided in ST.36/ST.66/ST.86 as the related field is optional in ST.36/ST.66/ST.86.

Addition

38. Some elements or attributes have newly been added in ST.96 which have no counterpart in ST.36, ST.66 or ST.86.

[TR-26] All IPO specific elements and attributes should be ignored in the transformation process.

[TR-27] In the case of the addition of an optional element or attribute, to convert to ST.96, the ST.96 field should not be populated, as there is no counterpart component in ST.36/ST.66/ST.86.

[TR-28] When adding a mandatory element or attribute, a mandatory element should not be created if the corresponding optional element or attribute is not present in the input instance. An error message should be given in the validation process of the output instance. The IPOs can decide how to resolve the differences on a case-by-case basis.

Renaming

39. Almost all fields have been renamed in ST.96. A one-to-one mapping is provided in Appendix A of this document. For example, the `absno` element in ST.36 is mapped to ST.96 `AbstractNumber`.

[TR-29] To convert to ST.96, the value of the ST.36/ST.66/ST.86 field should be copied to the ST.96 field.

[TR-30] To convert to ST.36/ST.66/ST.86, the value of the ST.96 field should be copied to the ST.36/ST.66/ST.86 field.

Change of elements order within a sequence construct

40. In ST.96 Schema, some elements have a different order for child elements from the corresponding elements defined in ST.36, ST.66 or ST.86. In a sequence construct, the order of child elements is important therefore the changed order of child elements within the sequence should be considered when transformation is performed.

Empty element

41. Since an empty element is not allowed in ST.96, a corresponding one-to-one mapping element to empty element within ST.36, ST.66 and ST.86 cannot be defined. Empty elements have been defined in various ways in the individual standards; therefore the transformation of these elements will differ on a case by case basis.

42. In ST.36, in some cases, elements indicating the existence of information are empty elements and the corresponding elements in ST.96 are defined as `xsd:boolean` Type. In other cases, the presence of some empty elements in an ST.36 XML instance can be mapped to an enumeration value in the ST.96 XML instance.

43. In the ST.66 model schema and ST.86 model schema, most elements have no mandatory child element, which means the parent elements can have empty content. In order to avoid empty content the `sequence` construct in the elements is changed to a multiple `choice` construct in corresponding elements defined in ST.96. This kind of structural change should be considered when the transformation is designed.

Change of cardinality

44. An element may be defined as optional in one Standard and as mandatory in another.

[TR-31] When changing the cardinality of an element from optional to mandatory, a mandatory element should not be created if the corresponding optional element is not present in the input instance. An error message should be given in the validation process of the output instance. The IPOs can decide how to resolve the differences.

Change of structure

45. For clarity, ST.96 proposes a different structure from the one defined in ST.36/ST.66/ST.86. For these cases, some conditions should be set for transformation on a case-by-case basis. For example,

- if doc-page is used in the context of drawings, check the type attribute.
  - If `@type= jpg` or `tif`, map with `PageImage`.
  - If `@type=pdf`, map with `DocumentURI`.
- in other contexts, check the presence of the ocr attribute.
  - If the `ocr` is defined, map with `DocumentURI`,
  - Otherwise map with `PageImage`.

4.7 Type Mismatches

46. In ST.96, some elements or attributes have type restrictions, but the ST.36 DTD has very few typed attributes or elements. Only `#PCDATA, ID, IDREF` and `CDATA` for atomic elements or attributes are used.

Pattern restriction

47. Some ST.36 fields are defined as `#PCDATA` and therefore do not contain pattern restrictions. These fields can easily accept the more restricted values of ST.96.

[TR-32] To convert to ST.96, the ST.36 field should be copied as-is to the ST.96 field. If the value does not conform to a pattern restriction in ST.96, it may have to be reformatted so that the instance will validate successfully.

[TR-33] To convert to ST.36, the ST.96 field value should be copied as-is into the free-text field of ST.36.

W3C Built-in data types

48. In principle, un-typed atomic elements or attributes defined in a DTD are mapped to `xsd:string` where a type is required. ST.96 uses the following W3C Built-in data types: `xsd:token, xsd:positiveInteger, xsd:boolean` and `xsd:string`. Although the format defined for some fields in ST.36 is alpha-numeric `(#PCDATA)`, their values are expected to follow the W3C built-in data types in ST.96.

[TR-34] To convert to ST.96, the ST.36 field should be copied as-is to the ST.96 field. If the value does not conform to the expected data type in ST.96, it may have to be reformatted so that the instance will validate successfully.

[TR-35] To convert to ST.36, the ST.96 field value should be copied as-is into the free-text field of ST.36.

5. REFERENCES

WIPO Standards

- WIPO Standard ST.36 — Processing of patent information using XML
- WIPO Standard ST.66 — Processing of trademark information using XML;
- WIPO Standard ST.86 — Processing of industrial design information using XML;

Industry Standards

- W3C XSLT: https://www.w3.org/TR/xslt/

APPENDIXES

The Appendixes are based on the version 7.0 of Annex III of WIPO Standard ST.96 (XML Schema V7_0) and their details are available here (AnnexVI_Appendices_A_B_C_V7_0.zip).

APPENDIX A: ELEMENT AND ATTRIBUTE MAPPING

Appendix A aims at providing a model one-to-one mapping between ST.96 elements and attributes and the corresponding elements and attributes of ST.36, ST.66 and ST.86. The one-to-one mapping is not always achieved due to reasons explained in Annex VI to ST.96, Transformation Rules and Guidelines; therefore, Appendix A is intended to provide a mapping between ST.96 and ST.36, ST.66 or ST.86 to the necessary degree.

The following mapping tables are provided in the corresponding file:

- Mapping Table for Elements and Attributes regarding the transformation of ST.36 `application-body` to ST.96 `ApplicationBody`
- Mapping Table for Elements and Attributes regarding the transformation of ST.96 `ApplicationBody` to ST.36 `application-body`
- Mapping Table for Elements and Attributes regarding the transformation of ST.36 `bibliographic-data` to ST.96 `BibliographicData`
- Mapping Table for Elements and Attributes regarding the transformation of ST.96 `BibliographicData` to ST.36 `bibliographic-data`
- Mapping Table for Elements and Attributes regarding the transformation of ST.66 `Transaction` to ST.96 `TrademarkTransaction`
- Mapping Table for Elements and Attributes regarding the transformation of ST.96 `TrademarkTransaction` to ST.66 `Transaction`
- Mapping Table for Elements and Attributes regarding the transformation of ST.86 `Transaction` to ST.96 `DesignTransaction`
- Mapping Table for Elements and Attributes regarding the transformation of ST.96 `DesignTransaction` to ST.86 `Transaction`

APPENDIX B: ENUMERATION LIST MAPPING

Appendix B aims at providing a model one-to-one mapping for codes or enumerated values between WIPO Standard ST.96, and WIPO Standards ST.36, ST.66 or ST.86. The one-to-one mapping is not always achieved due to reasons explained in Annex VI to ST.96, Transformation Rules and Guidelines.
The following mapping tables are provided in the corresponding file:

- Mapping Table for Enumeration List regarding the transformation of ST.36 `application-body` to ST.96 `ApplicationBody`
- Mapping Table for Enumeration List regarding the transformation of ST.96 `ApplicationBody` to ST.36 `application-body`
- Mapping Table for Enumeration List regarding the transformation of ST.36 `bibliographic-data` to ST.96 `BibliographicData`
- Mapping Table for Enumeration List regarding the transformation of ST.96 `BibliographicData` to ST.36 `bibliographic-data`

- Mapping Table for Enumeration List regarding the transformation of ST.66 `Transaction` to ST.96 `TrademarkTransaction`
- Mapping Table for Enumeration List regarding the transformation of ST.96 `TrademarkTransaction` to ST.66 `Transaction`
- Mapping Table for Enumeration List regarding the transformation of ST.86 `Transaction` to ST.96 `DesignTransaction`
- Mapping Table for Enumeration List regarding the transformation of ST.96 `DesignTransaction` to ST.86 `Transaction`

APPENDIX C: SAMPLE XSLT CODES

Appendix C aims at providing sample XSLT (Extensible Stylesheet Language Transformations) codes for data conversion between an ST.96 instance and an ST.36, ST.66 or ST.86 instance based on Appendixes A and B.

- Sample XSLT Codes for "`ApplicationBody`", "`BibliographicData`", "`TrademarkTransaction`" and "`DesignTransaction`": The conversion stylesheets consist of a set of files that can be used to convert ST.96 instances to ST.36, ST.66, or ST.86 instances and vice versa.

[Annex VII follows]