ST.96 - ANNEX VI

TRANSFORMATION RULES AND GUIDELINES

Version 0.4

Proposal presented by the XML4IP Task Force for consideration at the CWS/2

Editorial Note

The Transformation Rules and Guidelines (Annex VI) is a draft which requires implementation and further test by Offices to refine it as necessary. Subsequently, a final proposal for this Annex VI will be prepared and submitted for consideration and approval by the Committee on WIPO Standards.

Table of Contents

1. INTRODUCTION

1.1 Overview

1. Before adoption of WIPO Standard ST.96, WIPO Standards ST.36, ST.66 and ST.86 have already been used by Industrial Property Offices (IPOs). Therefore, maintaining transformability with XML document instances conforming to the existing XML Standards, i.e., ST.36, ST.66 and ST.86 is one of the primary concerns for WIPO Standard ST.96.

2. For facilitation of data exchange and interoperability between an IPO using the existing XML Standards 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 the existing XML Standards. Because of improvements based on experience and advances in technology, ST.96 structures will differ in many ways from those defined in the existing XML Standards. 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 term "*data transformation*" refers to converting data from a source data format into destination data format. It can be divided into two steps: data mapping and code generation;
- the term "*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;
- the term "*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;
- the term "*forward transformatio*n" covers the capability to map elements/attributes of the ST.96 to elements/attributes defined in ST.36, ST.66 and/or ST.86; and
- the term "*backward transformation*" covers the capability to map elements/attributes of the ST.36, ST.66 and ST.86 to elements/attributes of ST.96.

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 MAPPING SPREADSHEETS

9. The mapping tables are the heart of 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 has not always been achieved for reasons explained elsewhere. The mapping tables have been created in the form of Excel spreadsheets as Appendix A to this document. Each spreadsheet contains two worksheets, one for backward transformation and the other for forward transformation.

10. The following columns exist in each worksheet:

- For each source and target item: Tag Name, Full Path, Data Type and Cardinality
- Mapping comments: Conversion from the existing standard(s) to ST.96; and Conversion from ST.96 to the existing standard(s)

11. The *Tag Name* column defines the atomic element or attribute. For ST.96 components, tag names include namespace prefix.

12. The *Full Path* column defines the full path for elements or attributes. Hierarchical levels represent the path using XPATH notation. The elements are listed in the order of use. For XPath notation, a slash "/" is used to separate the hierarchical levels. In cases where an element is defined with a type that is another element (e.g., `<element name="ApplicantAddressBook type ="com:AddressBookType"/>`) an additional "/" is included along with the element type name. In addition, *Full Path* for target data may contain [NOT USED]. If the path is shown as [NOT USED] then this element does contain data but there is no direct mapping available in the target.

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

14. The *Cardinality* column, designates 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

15. The *Mapping Comments* column contains specific mapping or conversion instructions relating to the source and target described in Section "Guidance for Data Conversion".

ST.96				**ST.36**				**Mapping Comments**
Tag Name	**Full Path**	**Data Type**	**Cardi-nality**	**Tag Name**	**Full Path**	**Data Type**	**Cardi-nality**	**Conversion From ST.96 to ST.36**
com:id	ReportCitation/CitedReference/@id	xsd:token	0..1	id	citation/@id	ID	0..1	If set, copy If not set, no attribute

Example of Mapping table for forward transformation

ST.36				**ST.96**				**Mapping Comments**
Tag Name	**Full Path**	**Data Type**	**Cardi-nality**	**Tag Name**	**Full Path**	**Data Type**	**Cardi-nality**	**Conversion From ST.36 to ST.96**
id	citation/@id	ID	0..1	com:id	ReportCitation/CitedReference/@id	xsd:token	0..1	If set, copy If not set, no attribute

Example of Mapping table for backward transformation

[TR-01] A mapping table SHOULD be developed for each Document level schema defined in ST.96, e.g., ApplicationBody.xsd, to facilitate data transformation.

[TR-02] Forward and backward 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).

3. GUIDANCE FOR DATA CONVERSION

16. 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 originated from ST.36, ST.66 and/or ST.86 have been redefined by simplifying data structure or using new XML technologies including data-oriented design approach. The following issues should be considered when converting XML instances and necessary guidance are provided to address the issues below:

- *Date format conversion*: 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, priority date. In ST.66, ST.86 and ST.96, dates are defined as `xsd:date` which requires the format of YYYY-MM-DD.
- *Type mismatches*: In ST.96, some elements or attributes have type restrictions, but ST.36 DTD's have very few typed attributes or elements. Only `#PCDATA`, ID, IDREF and CDATA for atomic elements or attributes are used.
- *Identity constraints*: ST.96 recommends using `xsd:key/xsd:unique/xsd:keyref` for identity constraints while ST.36 uses ID/IDREF.
- *Code values and enumeration values mapping*: ST.36, ST.66, ST.86 and ST.96 provide sets of codes and enumeration values. In some cases, one format has additional code values than the other. Moreover, the naming convention of enumeration values is different from ST.96 to ST.36, ST.66 or ST.86. For example, ST.36 allows only characters in lower case. ST.66 and ST.86 recommends Upper case character for each word, while ST.96 recommends only the first character to be in upper case.
- *Different data structure*: ST.96 provides another way to structure the information that was defined in ST.36, ST.66 and ST.86.

3.1 Date format

17. ST.36 does not specify a strict format for dates because of DTD limitations. It is assumed that each IPO has a specific business practice defining consistently the date format in its XML instance.

18. All 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-03] 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-04] 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 dashes.

For example:

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

3.2 Code and enumeration values mapping

19. ST.36/66/86 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 some cases, however, ST.96 defines more or different code or enumerated values than the other Standards in order to reflect IPOs' practice and follow the ST.96 DRCs. Guidance is provided on the conversion of codes, depending on whether ST.96 supports the same codes or whether additional code values have been introduced.

20. Some ST.36 fields are defined as `#PCDATA` and do not define specific enumeration values. Although the format defined for the some fields in ST.36 is alpha-numeric (`#PCDATA`). In ST.96, an enumerated list or code is used in most cases.

[TR-23] To convert to ST.96, the ST.36 field value should be mapped to the list of ST.96 codes, and the mapped code should be inserted into the ST.96 field.

[TR-24] To convert to ST.36, the ST.96 code value should be copied as-is into the free-text field of ST.36.

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.

3.2.1 Same set of values

21. 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

22. In such case, the following recommendation applies:

[TR-07] 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-08] 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.

23. The fields that are marked with a plus sign (+) in Appendix B represent fields that are required in ST.96 and optional in ST.36, ST.66 and ST.86. In such case, the following conversion rules below apply:

[TR-09] 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-10] If the ST.36/ST.66/ST.86 component is not populated, the ST.96 component should acquire the “Undefined” value.

24. There is no case that 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.

3.2.2 Different set of values

25. 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

26. In some cases, ST.96 defines more codes or values than ST.36, ST.66 or ST.86. In these cases, backward transformation is not an issue. However, forward transformation is not ensured. In order to guarantee forward transformation, 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

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

[TR-11] 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 value has the same meaning.

[TR-12] 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 or ST.86 field, if the value has the same meaning.

3.3 Identity constraint

28. Some attributes in ST.36 employ the ID and IDREF types: For example, citation/id uses ID type. The ID and IDREF types permit values that are also permitted by the `xsd:token` type which is used for a similar purpose in ST.96. In either case, the value assigned in an instance to ID must be unique in the instance.

[TR-13] 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-14] To convert to ST.36/ST.66/ST.86, the value of the ST.96 field should be copied to the ST.36/66/86 field taking care to avoid duplication of ID values in the instance and revise any corresponding IDREF's.

3.4 Elements

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

3.4.1 Deletion

30. Some elements or attributes defined in ST.36, ST.66 or ST.86 have been deleted in ST.96. For example, citation/nplcit/article/book/text has no corresponding element in ST.96.

31. Some fields in ST.36/66/86 in Appendix A do not appear in ST.96 as they are no longer used. They are marked as "NOT USED" in Appendix A of this document.

[TR-15] 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-16] 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.

3.4.2 Addition

32. Some fields are new in ST.96. Guidance is provided on the way to handle such fields, depending on whether the ST.96 field has a counterpart in ST.36, ST.66 orST.86, or whether some mapping can be defined.

33. Some fields have been added in ST.96, with no counterpart in ST.36, ST.66 or ST.86. They are marked as "NEW" in Appendix A of this document.

[TR-19] 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-20] To convert to ST.36/ST.66/ST.86, the value of the ST.96 field should be ignored.

3.4.3 Renaming

34. 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-21] 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-22] To convert to ST.36/ST.66/ST.86, the value of the ST.96 field should be copied to the ST.36/66/86 field.

3.4.4 Change of elements order within sequence

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

3.4.5 Empty element

36. Since empty element is not allowed in ST.96, a corresponding element to empty element of ST.36, ST.66 and ST.86 cannot be defined in ST.96 in the way of one to one mapping. Furthermore there are various cases of empty elements and the way of transformation differs from one case to the other.

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

38. For another example, in ST.66 model schema and ST.86 model schema, most elements have no mandatory child element. It means the parent elements can have empty content. In order to avoid empty content, `sequence` construct in the elements is changed to multiple `choice` construct in corresponding elements defined in ST.96. The kind of structural changes should be considered when transformation is performed.

39. It is not worthwhile to provide specific transformation rule and guideline for the said structural changes because transformation should be addressed in case-by-case basis. In this regard, no further guidance is given in this document.

3.5 Field data type

3.5.1 Code or enumeration list

3.5.2 Pattern restriction

40. 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-25] 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 parse successfully.

[TR-26] To convert to ST.36, the ST.96 field value should be copied as-is into the free-text field of ST.36.

3.5.3 W3C Built-in data types

41. In principle, untyped 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 datatypes in ST.96.

[TR-27] 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 parse successfully.

[TR-28] To convert to ST.36, the ST.96 field value should be copied as-is into the free-text field of ST.36.

APPENDIX A: ELEMENT AND ATTRIBUTE MAPPING

The Appendix A aims at providing a model one-to-one mapping between ST.96 the elements and attributes and 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, this Appendix A is intended to provide mapping between ST.96 and ST.36, ST.66 or ST.86 in necessary degree.

The following mapping table is in the form of MS-Excel spreadsheets. There are two worksheets, as Appendix A to this document. Each spreadsheet contains two worksheets, one for mapping between ST.96 and ST.36, the other one is for mapping between ST.96 and ST.66 or ST.86.

- Model Mapping Table for Elements and Attributes of “Contact” [XLS]

APPENDIX B: ENUMERATION LIST MAPPING

The 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.

- Model Mapping Table for Enumeration List of “Contact” [XLS]

APPENDIX C: SAMPLE XSLT CODES

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

- Sample XSLT Codes for “Contact”: 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*. [ZIP]

[End of Annex VI]