TECHNICAL REPORT

ISO/IEC TR 24716

First edition
2007-12-15

Information technology — Programming languages, their environment and system software interfaces — Native COBOL Syntax for XML Support

Technologies de l'information — Langages de programmation, leur environnement et interfaces du logiciel système — Syntaxe COBOL native pour support XML

Reference number
ISO/IEC TR 24716:2007(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

In exceptional circumstances, the joint technical committee may propose the publication of a Technical Report of one of the following types:

— type 1, when the required support cannot be obtained for the publication of an International Standard, despite repeated efforts;

— type 2, when the subject is still under technical development or where for any other reason there is the future but not immediate possibility of an agreement on an International Standard;

— type 3, when the joint technical committee has collected data of a different kind from that which is normally published as an International Standard ("state of the art", for example).

Technical Reports of types 1 and 2 are subject to review within three years of publication, to decide whether they can be transformed into International Standards. Technical Reports of type 3 do not necessarily have to be reviewed until the data they provide are considered to be no longer valid or useful.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC TR 24716, which is a Technical Report of type 2, was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 22, *Programming languages, their environments and system software interfaces*, in collaboration with INCITS Techmical Committee J4, Programming language COBOL.

Introduction

This Technical Report provides extensions so that COBOL can process XML documents as easily as it can read files. The new syntax to process XML documents,

— is based on the familiar approach used with COBOL I/O support,

— provides Document Object Model (DOM) style parsing,

— handles multiple input sources to handle XML in an extremely flexible manner,

— reads, updates, and writes XML documents,

— checks that XML documents are well-formed, and

— provides an optional validity check of an XML document against a schema or Document Type Definition (DTD).

Technical Report ISO/IEC 24716 extends the COBOL specification defined in ISO/IEC 1989:2002, *Information technology — Programming languages — COBOL*. It provides new syntax to read, write, and update XML documents in COBOL.

Annex A forms a normative part of this Technical Report. Annex B and Annex C and the Bibliography are for information only.

Information technology — Programming languages, their environment and system software interfaces — Native COBOL Syntax for XML Support

1 Scope

This Technical Report specifies the syntax and semantics for XML support in COBOL. The purpose of this Technical Report is to promote a high degree of portability in implementations, even though some elements are subject to trial before completion of a final design suitable for standardization.

This specification builds on the syntax and semantics defined in ISO/IEC 1989:2002.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 1989:2002, *Information technology — Programming languages — COBOL*

Extensible Markup Language (XML) 1.0 (Fourth Edition), W3C Recommendation, 16 August 2006

Extensible Markup Language (XML) 1.1 (Second Edition), W3C Recommendation, 16 August 2006

Namespaces in XML 1.1, W3C Recommendation, 4 February 2004

XML Schema Part 1: Structures, W3C Recommendation, 28 October 2004

XML Schema Part 2: Datatypes, W3C Recommendation, 28 October 2004

3 Conformance to this Technical Report

This Technical Report is based on ISO/IEC 1989:2002. Conformance to this Technical Report does not require a full implementation of ISO/IEC 1989:2002. The interaction of the features of this technical report with features that are not provided by an implementation of ISO/IEC 1989:2002 is processor dependent.

4 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

4.1
document type definition
DTD
specification of the markup language that defines the elements, attributes, comments, and entities that a document may contain and specifies the relationships among them within the document

4.2
external document type definition
external DTD
DTD that is outside the file containing the XML document

4.3
internal document type definition
internal DTD
DTD that is in the same file as the XML document

4.4
XML document
unit of data that is well-formed as defined either in XML 1.0 or XML 1.1

4.5
XML element
portion of an XML document, the boundaries of which are either delimited by start-tags and end-tags, or, for empty elements, by an empty-element tag

NOTE Each element has a type, identified by name, and may have a set of attribute specifications.

4.6
XML file
file with XML organization

4.7
XML Schema
XML language for constructing schemas that describe the syntax of XML documents. Each schema defines a class of XML documents by constraining the structures and data types of instance documents that conform to the schema

5 Description techniques

Description techniques and language fundamentals are the same as those described in ISO/IEC 1989:2002.

6 Changes to ISO/IEC 1989:2002

These changes refer to clause and rule numbers in ISO/IEC 1989:2002.

6.1 Changes to 8, Language fundamentals

[a] Add the following reserved words to the list in 8.9, Reserved Words

DOCUMENT
END-OPEN
IDENTIFIED
VERSION-XML

[b] In 8.10, Context-sensitive words, change the context for ATTRIBUTE to read:

"IDENTIFIED clause, DELETE statement, READ statement, REWRITE statement, SET statement, START statement, and WRITE statement"

[c] In 8.10, Context-sensitive words, change the context for ONLY to read:

"Object-view, SHARING clause of the file control entry, SHARING phrase of the OPEN statement, USAGE clause, READ statement, REWRITE statement, and WRITE statement"

[d] Add the following context-sensitive words to the list in 8.10, Context-sensitive words

CHECK	TYPE clause of the file control entry
DISCARD	CLOSE statement
DTD	TYPE clause of the file control entry
ELEMENT	DELETE statement, READ statement, REWRITE statement, START statement, and WRITE statement

NAMESPACE	IDENTIFIED clause
RAW	IDENTIFIED clause
SCHEMA	TYPE clause of the file control entry
STACK	OPEN statement
VALIDITY	TYPE clause of the file control entry
XML	ACCESS clause and ORGANIZATION clause

6.2 Changes to 9, I-O, objects, and user-defined functions

[a] 9.1.3, File connector, second paragraph, second sentence, add "XML" to the end of the list of types of file organization in the parenthetical and to the end of the list of access modes in the parenthetical.

[b] 9.1.6, Fixed file attributes, change the third sentence to read:

"The file organizations are sequential, relative, indexed, and XML."

[c] 9.1.7, Organization,

[1] Change the first sentence to read:

"The file organizations are sequential, relative, indexed, and XML."

[2] add 9.1.7.4, XML, as follows:

"9.1.7.4 XML

XML is a file organization used for data that is in the format of an XML document or a sequence of XML documents. An XML document can be on a traditional file medium or it can be in memory. An XML document is a string of text that is given a structure by the presence of tags, which separate the document into elements. XML documents are organized in a hierarchical manner, similar to a COBOL record structure, where an element may contain other elements. Within the context of its superordinate elements, each subordinate element may be stored, retrieved, or deleted based on its tag name. The element position vector for an XML file associates an element position or an attribute position with each data item for which an IDENTIFIED clause is specified.

COBOL allows multiple documents in a file that is open in the input, output, or extend mode. COBOL allows only one document in a file that is open in the i-o mode. The XML 1.0 and 1.1 recommendations do not consider the processing of anything beyond a single document. Therefore, many XML processors might not consider multiple-document files to be well-formed.

The file format OPEN statement for an XML file establishes a connection to the physical file. The file position indicator maintains the document number of the current document in the XML file. An XML-document format OPEN statement creates an internal representation of the current document. The READ statement transfers data into associated data items described in the file section. The START statement positions the element position vector to attributes or elements within the internal representation. The DELETE statement, REWRITE statement, and WRITE statement change the internal representation, but do not change the physical file. The XML-document format CLOSE statement optionally writes an XML document based on the internal representation and then deletes the internal representation; the file remains in the open state. A file format CLOSE statement closes the physical file.

COBOL processes documents that conform to XML 1.0 or XML 1.1.

On input, COBOL resolves character references, predefined entity references, general entity references, and parameter entity references consistent with the TYPE clause of the file control entry, and makes the content of CDATA sections available to the program. COBOL does end-of-line normalization and attribute-value normalization as specified in the applicable XML recommendation. All required parts of the XML document are made available to the COBOL program with the exception of Processing Instructions.

COBOL, by default writes XML documents that are well-formed with respect to XML 1.0 or optionally to XML 1.1.

All XML documents written by COBOL start with an XML declaration that includes the version information and an encoding declaration. If a DTD is included, the XML declaration also includes a standalone document declaration with a value of “yes.”

XML documents written by COBOL contain predefined references and character references as required by the appropriate XML Recommendation.

XML documents written by COBOL do not contain general entity references, parameter entity references, CDATA sections, processing instructions, or white space. The implementor may include comments.

NOTE The term 'white space' is defined in XML 1.1."

[d] 9.1.8, Access modes,

[1] In the first paragraph, last sentence, add "XML" to the list of access modes.

[2] In the second paragraph, add the following as a new last sentence:

"A file with organization XML may be accessed only in XML access mode."

[3] Add the following:

"9.1.8.4 XML access mode

When a file is accessed in XML access mode, the order of access of the documents in that file is the sequential order of the documents in that file.

Within each document, START statements may be used to set the element position vector to a specific occurrence of an attribute or element."

[e] 9.1.11, File position indicator, second paragraph, insert the following text after "for a relative file,":

"the document number of the current document for an XML file,"

[f] Add the following after 9.1.11, File position indicator:

"9.1.11a Element position vector

The element position vector is a conceptual entity that exists for each XML file to maintain the positions of attributes and elements within each level in the hierarchy of an XML document. Position is maintained for each attribute or element in the COBOL record description that is described with an IDENTIFIED clause. Attributes and elements may be accessed by their attribute names or element names, respectively, as long as all containing elements are indicated by the element position vector.

The element position vector is set only by CLOSE, OPEN, READ, and START statements. The XML-document format OPEN statement for a file open in input mode or i-o mode sets the element position vector to the first XML element associated with each COBOL data item that has an IDENTIFIED clause and to the first attribute, if any, of each of those elements. The XML-document format OPEN statement for a file open in extend mode or output mode sets the element position vector at the start of a new document. The START statement positions the element position vector to a specific occurrence of an attribute or element. The READ statement accesses occurrences of elements sequentially and repositions the element position vector. The READ statement accesses occurrences of attributes sequentially when the associated IDENTIFIED clause contains the USING phrase and repositions the element position vector. The READ statement accesses the specified occurrence of an attribute when the associated IDENTIFIED clause contains the BY phrase, but does not reposition the element position vector. The CLOSE statement causes the element position vector to indicate that no valid position has been established."

[g] 9.1.12.1, Successful completion, add the following new entries:

“6) I-O status = 08. The input-output statement is successfully executed but a READ statement ignored one or more XML attributes or elements, or both, because there is no COBOL data item with which one or more attributes or elements can be associated.

7) I-O status = 09. The input-output statement is successfully executed but the referenced DTD or schema is not available; the requested validity check of the XML document was not done."

[h] 9.1.12.4, Invalid key condition with unsuccessful completion, I-O status 23, rule 3, change the period to "; or" at the end of 3d and insert the following text:

"e) an attempt is made to access a document that does not exist at the current file position in an XML file; or

f) an attempt is made to access an attribute or element that does not exist at the current position in the internal representation of an XML file."

[i] 9.1.12.4, Invalid key condition with unsuccessful completion, I-O status 24, rule 4, change to read:

"4) I-O status = 24. This condition exists because:

"a) an attempt is made to write beyond the externally-defined boundaries of a physical relative or indexed file; the implementor specifies the manner in which these boundaries are defined; or

b) a sequential WRITE statement is attempted for a relative file and the number of significant digits in the relative record number is larger than the size of the relative key data item described for the file; or

c) an attempt is made to close an XML document in an in-memory XML file and the XML document to be created is larger than the associated length of that memory."

[j] 9.1.12.4, Invalid key condition with unsuccessful completion, add the following text:

"5) I-O status = 25. An attempt is made to delete an XML attribute or element at an invalid position."

[k] 9.1.12.5, Permanent error condition with unsuccessful completion, rule 3, I-O status = 34, first sentence, insert the following text after "externally-defined boundaries of":

"an XML file or"

[l] 9.1.12.5, Permanent error condition with unsuccessful completion, add the following text:

"8) I-O status = 3A. An OPEN statement is attempted on an XML document that is not well-formed.

9) I-O status = 3B. A CLOSE statement is attempted on an XML document that is not valid.

10) I-O status = 3C. An OPEN statement with the I-O phrase is attempted on a multiple document file.

11) I-O status = 3D. An OPEN statement is attempted on a document in a file that is open in the input or i-o mode and the coded character set used for encoding the document could not be determined."

[m] 9.1.12.6, Logic error condition with unsuccessful completion, add the following text:

"10) I-O status = 4A. An attempt is made to write XML that is not well-formed by a document-format CLOSE statement, a REWRITE statement, or a WRITE statement.

11) I-O status = 4B. An attempt is made to reference a file connector that is not open by an XML-document format CLOSE statement or an XML-document format OPEN statement.

12) I-O status = 4C. The execution of a READ statement is unsuccessful because duplicate names are specified in IDENTIFIED clauses at the same level in the hierarchy of the associated record description entry.

13) I-O status = 4D. An attempt is made by an XML-document format CLOSE statement to reference a file connector that does not have an open document.

14) I-O status = 4E. An attempt is made by a WRITE or REWRITE statement to create XML text with an attribute name or element name that cannot be represented in the coded character set used for the encoding of the XML document."

6.3 Changes to 12, Environment division

[a] Add a new file control entry format to 12.3.4.1, General format:

"Format 5 (XML):

SELECT [OPTIONAL] file-name-1

ASSIGN { TO { { device-name-1 / literal-1 } … [USING data-name-1] / DATA data-name-9 / data-name-10 LENGTH IS data-name-11 } / USING data-name-1 }

[ACCESS MODE IS XML]

[FILE STATUS IS data-name-4]

ORGANIZATION IS XML

[TYPE IS { DTD / EXTERNAL { DTD / SCHEMA } { literal-3 / data-name-12 } } [CHECK VALIDITY ON { INPUT / OUTPUT }]]

[VERSION-XML IS literal-4].

NOTE ORGANIZATION is a required word in the XML format, rather than optional, so that XML can be a context-sensitive word. "

[b] Add the following new file control entry syntax rules to 12.3.4.2:

"FORMAT 5

"14) The physical file associated with file-name-1 shall have organization XML. The associated file description entry shall not be a sort-merge file description entry.

15) Data-name-9 shall reference a data item of category alphanumeric or national and shall not be subordinate to the file description entry for file-name-1.

16) Data-name-10 shall reference a data item of category data-pointer and shall not be subordinate to the file description entry for file-name-1.

17) Data-name-11 shall reference an unsigned integer data item described without the picture symbol 'P' and shall not be subordinate to the file description entry for file-name-1.

18) Data-name-12 shall reference a data item of category alphanumeric or national and shall not be subordinate to the file description entry for file-name-1.

19) The OPTIONAL phrase shall not be specified when the DATA or LENGTH phrase is specified."

[c] Change 12.3.4.3 file control entry general rule 1:

[1] subrule 1b, add data-name-9, data-name-10, and data-name-11 to both lists.

[2] add the following additional subrules:

"n) The same specification of the TYPE clause, when data-name-12, if specified, references an external data item.

o) The same specification of the VERSION-XML clause."

[d] Change 12.3.4.3 file control entry general rule 3:

[1] add the following text at the end of the first sentence:

"; or to a memory location referenced by either data-name-9 or the content of data-name-10"

[2] subrule 3a, change in part to read:

"When device-name-1 or literal-1 is specified and the USING phrase is omitted …"

[3] add the following new subrules:

"c) If data-name-9 is specified, one or more XML documents are contained in the data item referenced by data-name-9. The XML document is said to be in-memory.

d) If data-name-10 is specified, one or more XML documents begin at the address specified by the content of the data item referenced by data-name-10 and continue for the number of bytes specified by the content of the data item referenced by data-name-11. The XML document is said to be in-memory."

[e] Add the following new file control entry general rule to 12.3.4.3:

"FORMAT 5

"10) The XML format defines a file connector for an XML file. "

[f] In 12.3.4.4.1, general format of the ACCESS MODE clause, add "XML" to the stack in the curly braces.

[g] In 12.3.4.4.2, syntax rules of the ACCESS MODE clause, add the following new syntax rules:

"3) The DYNAMIC, RANDOM, and SEQUENTIAL phrases shall not be specified for an XML file.

4) The XML phrase may be specified only for an XML file."

[h] In 12.3.4.4.3, general rules of the ACCESS MODE clause,

[1] Change general rule 1 to read:

"1) If the ACCESS MODE clause is not specified,

a) if the organization is XML, XML access is assumed,

b) for other organizations, sequential access is assumed."

[2] add the following new general rule:

"5) If the access mode is XML, documents within the file are accessed in sequential order. Elements in the current document can be accessed via their element names only when all containing elements are indicated by the element position vector."

[i] In 12.3.4.9.1, General format [ORGANIZATION clause],

[1] Add the heading "FORMAT 1 (sequential-relative-indexed)" before the existing format.

[2] Add the following after the existing format:

"Format 2 (XML):

ORGANIZATION IS XML"

[j] In 12.3.4.9.2, General rules [ORGANIZATION clause], add the following:

"4a) The XML phrase specifies that the file organization is XML. XML organization is a permanent logical file structure in which each constituent record is an XML document."

[k] Make the following changes to 12.3.6, SAME clause

[1] Syntax rule 7, insert the following text after "a report file":

", an XML file,"

[2] Add the following new syntax rule:

"6a) A given file-name that represents an XML file may be specified in one file-area format SAME clause, and shall not be specified in a record-area format or sort-merge-area format SAME clause."

[l] Add the following after 12.3.4.15, SHARING clause:

"12.3.4.16 TYPE clause

The TYPE clause specifies the SCHEMA or DTD that describes the XML document.

12.3.4.16.1 General format

```
TYPE IS { DTD
        | EXTERNAL { DTD    } { literal-1   } }
                   { SCHEMA } { data-name-1 }
[ CHECK VALIDITY ON { INPUT  } ]
                    { OUTPUT }
```

12.3.4.16.2 Syntax rules

1) Literal-1 shall be an alphanumeric or national literal and shall not be a figurative constant.

2) Data-name-1 may be qualified.

12.3.4.16.3 General rules

1) If the TYPE clause is specified without the EXTERNAL phrase, then the document is described by an internal DTD.

2) If the EXTERNAL phrase and the DTD phrase are specified, the document is described by the external DTD specified by literal-1 or data-name-1.

3) If the SCHEMA phrase is specified, the document is described by the schema specified by literal-1 or data-name-1. Schemas are defined in XML Schema Part 1: Structures and XML Schema Part 2: Datatypes.

4) If the INPUT phrase is specified, when an XML-document format OPEN statement is executed, the specified DTD or schema is used to check the document for validity.

5) If the OUTPUT phrase is specified, when an XML-document format CLOSE statement is executed, the specified DTD or schema is used to check the document for validity.

12.3.4.17 VERSION-XML clause

The VERSION-XML clause specifies the version of the XML specification to which created XML files conform.

12.3.4.17.1 General format

```
VERSION-XML IS literal-1
```

12.3.4.17.2 Syntax Rules

1) Literal-1 shall be an alphanumeric or national literal with the value '1.0' or '1.1'.

12.3.4.17.3 General Rules

1) When an XML-document format CLOSE statement is executed for a file connector open in the extend or output mode,

   – if literal-1 is 1.1, it is written as the version number in the XML declaration; otherwise,

   – 1.0 is written as the version number in the XML declaration."

6.4 Changes to 13, Data division

[a] 13.3.4.1, General format of the file description entry, add the following new format:

"Format 4 (XML):

```
FD file-name-1
        [ IS EXTERNAL [ AS literal-1 ] ]
        [ IS GLOBAL ]
        [ CODE-SET IS { alphabet-name-3 } ] . "
                      { data-name-6     }
```

[b] 13.3.4.2, Syntax rules of the file description entry, add the following new rules:

"FORMAT 4

"9) Format 4 is the file description entry for an XML file.

10) One or more record description entries shall be associated with the XML file description entry.

11) Subordinate entries shall define data items that are category alphabetic, alphanumeric, boolean, national, or numeric."

[c] Add the following new clauses in alphabetical order to the format 1 data description entry, in 13.14.1:

"COUNT IN data-name-7"

"IDENTIFIED { BY { data-name-8 / literal-2 } / USING data-name-9 } IS { ATTRIBUTE / ELEMENT [RAW] } [NAMESPACE { IS { data-name-10 / literal-3 / NULL } / USING data-name-11 }] "

[d] CODE-SET clause, 13.16.13:

[1] In 13.16.13.1, General format, add the heading "FORMAT 1 (sequential):" before the current syntax diagram and add the following after that syntax diagram:

"Format 2 (XML):

CODE-SET IS { alphabet-name-3 / data-name-1 } "

[2] In 13.16.13.2, Syntax rules, add the heading "FORMAT 1" before the current syntax rules and the following after those rules:

"FORMAT 2:

4) Alphabet-name-3 shall reference an alphabet that defines a coded character set.

5) Data-name-1 may be qualified.

6) Data-name-1 shall reference either an alphanumeric or a national data item."

[3] In 13.16.13.3, General rules, old general rules 1, 6, and 7 apply to both formats. Old general rules 2 through 5 apply only to format 1. The following new general rules apply only to format 2:

"8) If the file connector associated with the file description in which this clause is specified is open in the extend or output mode, during the execution of an XML-document format OPEN statement the operating environment shall save the encoding referenced by the specified code-set. The execution of an XML-document format CLOSE statement referencing this file connector writes this saved encoding as the encoding declaration in the XML document.

If alphabet-name-3 is specified, the coded character set used to represent data on the storage medium is the one referenced by alphabet-name-3.

If data-name-1 is specified, the coded character set used to represent data is the one whose name is contained in the data item referenced by data-name-1 after the leading and trailing spaces are deleted.

9) If the file connector associated with the file description in which this clause is specified is open in the input, i-o, or extend mode, during the execution of an XML-document format OPEN statement:

a) If alphabet-name-3 is specified and the referenced encoding does not match the encoding declaration specified in the document, the EC-XML-CODESET exception is set to exist. If a RESUME statement with the NEXT STATEMENT phrase is executed in an associated declarative and the implementation can determine the coded character set used for encoding the document, processing continues with the OPEN statement and the OPEN statement is successful.

b) If data-name-1 is specified, the encoding declaration specified in the document is moved into the data item referenced by data-name-1.

10) The specified encoding is used for code-set conversion of all data items in the document. Both UTF-8 and UTF-16 shall be supported by the implementation. If the CODE-SET clause is not specified, UTF-8 is used by default."

[e] Add the following new clause 13.16.15a, COUNT clause:

"13.16.15a COUNT clause

The COUNT clause establishes a temporary integer data item. The XML-element format READ statement places the number of occurrences of an attribute or element read into this data item. The XML-element format WRITE and XML-element format REWRITE statements use the value of this data item to determine the number of occurrences to write or rewrite.

13.16.15a.1 General format

COUNT IN data-name-1

13.16.15a.2 Syntax rules

1) The COUNT clause may be specified only in a record description subordinate to a file with organization XML.
2) Data-name-1 shall not be defined elsewhere in the source element.

13.16.15a.3 General rules

1) Each entry in which a COUNT clause is specified establishes a temporary integer data item that is implicitly defined with usage binary-long signed. If the COUNT clause is specified in a data description entry with an OCCURS clause, the dimension of the temporary integer data item referenced by data-name-1 is one less than the dimension of the subject of the entry. If the COUNT clause is specified in a data description without an OCCURS clause, the dimension of the temporary integer data item referenced by data-name-1 is the same as the dimension of the subject of the entry. If the subject of the entry is subordinate to an entry described with the EXTERNAL clause, this temporary data item is external. If the subject of the entry is subordinate to an entry described with the GLOBAL clause, the entry-name specified for the subject of the entry is a global name.
2) When an XML-element format REWRITE statement or an XML-element format WRITE statement is executed for the file, the content of the data item referenced by data-name-1 specifies the number of occurrences of the subject of the entry to write. If the specified number of occurrences is greater than the number of occurrences described for the associated item, the EC-XML-COUNT exception condition is set to exist and no occurrences are written.
3) The execution of a REWRITE or WRITE statement and the unsuccessful execution of a READ statement do not alter the content of the data item referenced by data-name-1.
4) After the successful execution of a READ statement, the content of the data item referenced by data-name-1 indicates the number of occurrences of the subject of the entry that were just read.

NOTE If the subject of the entry is not subject to an OCCURS clause, the data item referenced by data-name-1 contains either zero or one, to indicate whether that element was present in the XML document. If the subject of the entry is subject to an OCCURS clause, the data item referenced by data-name-1 indicates how many occurrences were read. The number of occurrences read is limited by the maximum size of the table and may be smaller than the number of occurrences in the document."

[f] In the general rules of the EXTERNAL clause, 13.16.20.3, general rule 2.

[1] Rule 2a, delete the "and" at the end.

[2] Rule 2b, change in part to read: "as external, subject to general rules 3 and 4; and

[3] Add 2c as follows:

"c) the internal representation of an XML file is external and may be accessed by any runtime element in the run unit that describes the same file and records as external, subject to general rules 3 and 4."

[g] Add the following:

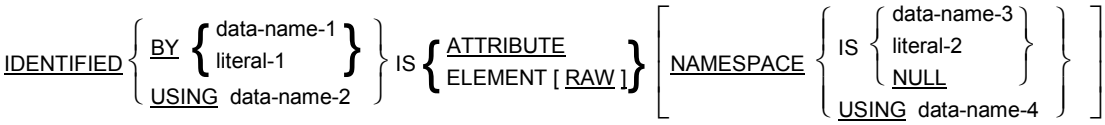
"13.16.28a IDENTIFIED clause

The IDENTIFIED clause maps an XML element name or attribute name to a COBOL data name.

The RAW phrase of the IDENTIFIED clause indicates that data is to be transferred into the data item without decomposition and transferred out of the data item intact, without processing or validation.

The NAMESPACE phrase of the IDENTIFIED clause sets or identifies the namespace that applies to the subject of the entry.

13.16.28a.1 General Format



13.16.28a.2 Syntax Rules

1) The IDENTIFIED clause may be specified only in a record description subordinate to a file with organization XML.

2) If the subject of the entry is described with an IDENTIFIED clause, all superordinate data description entries shall contain an IDENTIFIED clause.

3) Literal-1 and literal-2 shall be of category alphanumeric or national.

4) Data-name-1, data-name-2, data-name-3, and data-name-4 shall reference a data item of category alphanumeric or national that is not described with an IDENTIFIED clause.

5) The content of literal-1 shall be a valid XML name.

6) The content of literal-2 shall be a valid namespace as specified by Namespaces in XML 1.1.

7) Data-name-1, data-name-2, data-name-3, and data-name-4, if subordinate to a file with organization XML, shall be directly subordinate to the subject of the entry.

8) If a namespace is associated with the subject of the entry:

   a) If the ATTRIBUTE phrase is specified, the value of literal-1 shall be unique within the values of literal-1 for attributes in the applicable namespace that are specified at the same level as the subject of the entry.

   b) If the ELEMENT phrase is explicitly or implicitly specified, the value of literal-1 shall be unique within the values of literal-1 for elements in the applicable namespace that are specified at the same level as the subject of the entry.

9) If a namespace is not associated with the subject of the entry:

   a) If the ATTRIBUTE phrase is specified, the value of literal-1 shall be unique within the values of literal-1 for attributes without an associated namespace that are specified at the same level as the subject of the entry.

   b) If the ELEMENT phrase is explicitly or implicitly specified, the value of literal-1 shall be unique within the values of literal-1 for elements without an associated namespace that are specified at the same level as the subject of the entry.

10) The USING phrase shall not be specified at any given level in a record description if the BY phrase is specified at that level.

11) If the ATTRIBUTE phrase is specified, no subordinate data description entries may specify the IDENTIFIED clause.

12) There shall be at most one data description entry that is directly subordinate to the subject of the entry and that has no IDENTIFIED clause in its description and that is not referenced by this IDENTIFIED clause.

    NOTE This subordinate entry describes the data item associated with the data content of the subject of the entry.

13) If the RAW phrase is specified, no subordinate data description entries shall contain an IDENTIFIED clause.

14) If the RAW phrase is specified, the subject of the entry shall be category alphanumeric or national.

13.16.28a.3 General Rules

1) The IDENTIFIED clause associates the subject of the entry with the XML element identified by literal-1 or the content of data-name-1 or data-name-2. This association exists only when each superordinate entry is associated with a corresponding XML parent element.

If the subject of the entry is an elementary item, this elementary item is associated with the data content of the subject of the entry.

If the subject of the entry is a group item, the data description entry, if any, that is directly subordinate to the subject of the entry and that has no IDENTIFIED clause in its description and that is not referenced by this IDENTIFIED clause describes the data item associated with the data content of the subject of the entry.

2) On input, if literal-1 is specified, the subject of the entry is associated only with an XML attribute or element with the name specified by literal-1.

3) On input, if data-name-1 is specified, the subject of the entry is associated only with an XML attribute or element with the name specified by data-name-1.

4) On input, if data-name-2 is specified, the subject of the entry is associated with the next XML attribute or element as specified by the element position vector, whichever is specified in the IDENTIFIED clause, and the XML attribute name or element name is moved into data-name-2. If one input statement reads several elements or attributes whose associated data description entries reference the same data-name-2 in IDENTIFIED clauses with the USING phrase, the data item referenced by data-name-2 is updated for each of these entries in the order the elements occur in the XML document.

   NOTE Multiple references to data-name-2 can occur when the subject of the entry has subordinate data items or when data-name-2 is subject to an OCCURS clause.

5) On output, the content of data-name-1 or data-name-2 after trailing blanks are trimmed shall be a valid XML name. If the JUSTIFIED clause is specified in the description of data-name-1 or data-name-2, leading, rather than trailing, spaces are trimmed.

   NOTE The version of XML specified in the file control entry determines the characters that are allowed in an XML name.

6) On output, the value of literal-1, the content of data-name-1, or the content of data-name-2 specifies the XML attribute name or element name that is written.

7) If the ATTRIBUTE phrase is specified, the value of literal-1, the content of data-name-1, or the content of data-name-2 shall identify an attribute; otherwise, they shall identify an element.

8) When the subject of the entry is directly or indirectly referenced in a READ statement and a namespace is associated with the subject of the entry:

   a) If the ATTRIBUTE phrase is specified, the value of literal-1 or the content of data-name-1 shall be unique within the values of literal-1 and the content of data-name-1 for attributes in the applicable namespace that are specified at the same level as the subject of the entry within a given record description entry; otherwise, the I-O status in the associated file connector is set to '4C' and the execution of the statement is unsuccessful.

   b) If the ELEMENT phrase is explicitly or implicitly specified, the value of literal-1 or the content of data-name-1 shall be unique within the values of literal-1 and the content of data-name-1 for elements in the applicable namespace that are specified at the same level as the subject of the entry within a given record description entry; otherwise, the I-O status in the associated file connector is set to '4C' and the execution of the statement is unsuccessful.

9) When the subject of the entry is directly or indirectly referenced in a READ statement and a namespace is not associated with the subject of the entry:

   a) If the ATTRIBUTE phrase is specified, the value of literal-1 or the content of data-name-1 shall be unique within the values of literal-1 and the content of data-name-1 for attributes without an associated namespace that are specified at the same level as the subject of the entry within a given record description entry; otherwise, the I-O status in the associated file connector is set to '4C' and the execution of the statement is unsuccessful.

   b) If the ELEMENT phrase is explicitly or implicitly specified, the value of literal-1 or the content of data-name-1 shall be unique within the values of literal-1 and the content of data-name-1 for elements without an associated namespace that are specified at the same level as the subject of the entry within a given record description entry; otherwise, the I-O status in the associated file connector is set to '4C' and the execution of the statement is unsuccessful."

10) The NAMESPACE phrase specifies the namespace for the subject of the entry and for each subordinate item for which the IDENTIFIED clause with the explicit or implicit ELEMENT phrase is specified.

11) If a NAMESPACE phrase is specified in an item subordinate to a group item for which a NAMESPACE phrase is specified, the NAMESPACE phrase in the subordinate item takes precedence for that subordinate item.

12) If the NULL phrase is specified, the namespace is the empty string, which is equivalent to specifying no namespace.

13) Execution of an XML-document format WRITE statement or XML-document format REWRITE statement uses the content of the data item referenced by data-name-3 or data-name-4 or the value of literal-2 as the namespace of the attribute or element in the internal representation. If the content of that data item or the value of that literal is spaces, the namespace is the empty string.

NOTE The namespace for an XML element can be the default namespace, but attributes use an explicit prefix. Per Namespaces in XML 1.1, “Default namespace declarations do not apply directly to attribute names; the interpretation of unprefixed attributes is determined by the element on which they appear.”

14) When the associated file connector is open in the input or I-O mode, the namespace declarations in the XML document specify the namespaces of the document.

If literal-2 or data-name-3 is specified and does not match the corresponding namespace attribute specified in the document, the EC-XML-NAMESPACE exception condition is set to exist. If data-name-3 is specified and no namespace is specified in the document or the namespace specified is the empty string, data-name-3 is compared to spaces.

If data-name-4 is specified, the namespace attribute specified in the document is moved into the data item referenced by data-name-4. If no namespace is specified in the document or the namespace specified is the empty string, spaces are moved into the data item referenced by data-name-4.

15) If the RAW phrase is specified, the XML-value in 14.6a.1, Transfer of input data, consists of all the XML character data, including, for example, subordinate markup, processing instructions, character data and white space, in the same order as specified in the XML document. On output the data is transferred intact with no processing or validation.

NOTE The XML-document format CLOSE statement checks that the resulting document is well-formed.

16) If the RAW phrase is not specified, the characters & < > ' " in the content of the sending COBOL data item are converted to the strings "&" "<" ">" "'" """ respectively. If the RAW phrase is specified, these characters are not converted and are interpreted as XML control characters.

NOTE In order to transfer data that includes an ampersand, less than symbol, greater than symbol, apostrophe, or quotation mark when the RAW phrase is specified, the programmer can specify '&' '<' '>' ''' '"', respectively, in the content of the sending COBOL data item."

[h] In 13.16.31.3, general rules for Level-number, change general rule 3 to read:

"Multiple level 1 entries subordinate to a CD, FD with organization other than XML, or SD entry represent implicit redefinitions of the same area. Multiple level 1 entries subordinate to a report description entry or an FD with organization XML do not represent implicit redefinitions of the same area."

[i] In 13.16.36.2, syntax rules for the OCCURS clause, add the following new syntax rule after the FORMAT 2 heading:

"17a) The OCCURS clause shall not be specified subordinate to a file description entry associated with a file with organization XML."

[j] In 13.16.42.2, syntax rules for the REDEFINES clause, replace syntax rule 3 with the following:

"This clause shall not be specified in level 1 entries in the file section, in any entry subordinate to a file description entry that contains a FORMAT clause, or in any entry subordinate to a file description entry associated with a file that has organization XML."

[k] In 13.16.43.2, syntax rules for the RENAMES clause, add a new syntax rule as follows:

"3a) The RENAMES clause shall not be specified subordinate to a file description entry associated with a file with organization XML."

[l] In 13.16.55.2, syntax rules for the TYPE clause, add a new syntax rule that applies to format 1 as follows:

"8a)When the TYPE clause is specified in a record description subordinate to a file with organization XML, the STRONG phrase shall not be specified in the description of type-name-1."

6.5 Changes to 14, Procedure division

[a] In 14.4, Procedural statements and sentences, Table 13, Procedural statements, in the row for OPEN, add "[NOT] AT END" to the second column and "END-OPEN" to the third column.

[b] In 14.5.12.1, Exception conditions, fifth paragraph, third sentence, add EC-XML to the list of level-2 exception-names.

[c] In 14.5.12.1.5, Exception-names and exception-conditions, add the following new exception conditions to Table 14, Exception-names and exception-conditions:

Exception-name	Cat	Description
EC-DATA-INFINITY	Fatal	INF (infinity) cannot be assigned to the receiving item
EC-DATA-NEGATIVE-INFINITY	Fatal	-INF (negative infinity) cannot be assigned to the receiving item
EC-DATA-NOT-A-NUMBER	Fatal	NaN (not a number) cannot be assigned to the receiving item
EC-XML		XML error
EC-XML-CODESET	NF	The encoding declaration in the XML declaration in the XML document does not match the encoding specified in the CODE-SET clause
EC-XML-CODESET-CONVERSION	NF	One or more characters in an XML attribute or element name transferred from an XML document or one or more characters in XML data transferred to or from an XML document had no corresponding character in the target coded character set
EC-XML-COUNT	NF	A REWRITE or WRITE statement was executed on an XML file, but the number in a temporary data item defined by a COUNT clause exceeds the number of associated items that are described
EC-XML-DOCUMENT-TYPE	NF	The schema or DTD referenced in the XML document does not match that specified in the TYPE clause
EC-XML-IMPLICIT-CLOSE	NF	A file format CLOSE statement was executed on an XML file without closing the document. The updates made to the internal representation of the XML document are about to be discarded
EC-XML-INVALID	NF	The XML document does not conform to the DTD or schema
EC-XML-NAMESPACE	NF	A namespace attribute value in the XML document did not match the namespace specified in the corresponding NAMESPACE phrase of the IDENTIFIED clause
EC-XML-STACKED-OPEN	Fatal	An XML-document format OPEN statement with the STACK phrase failed because the specified identifier was not associated with a valid element position by the element position vector
EC-XML-RANGE	NF	The exponent of a floating point input value is greater than the implementation supports

[d] Add the following:

"14.6a Common XML Processes

This clause describes common processes that pertain to XML input and output. Clause 14.6a.1, Transfer of input data, pertains to the READ statement. Clause 14.6a.2, Transfer of output data, pertains to the REWRITE and WRITE statements.

14.6a.1 Transfer of input data

Processing of input data is described as though the encoding of the internal representation is UTF-16.

Transfer of input data is defined for one sending operand and one receiving operand. The sending operand is an elementary national character string, XML-value, of the length of the value made available by the READ statement. The value made available by the READ statement shall include all the character data including the white space. If the RAW phrase is specified for an element, XML-value shall include all the XML character data between the opening delimiter and the closing delimiter of the element, including, for example, subordinate markup, processing instructions, character data and white space, in the same order as specified in the XML document. If the RAW phrase is not specified for an element, XML-value shall include all the character data between the opening delimiter and the closing delimiter of the element except for processing instructions and subordinate elements. The receiving operand is the COBOL data item associated with the data content of the COBOL data item associated with the XML attribute or element.

If the sending operand is of length zero, the receiving operand is initialized as though the following statement were processed: INITIALIZE receiving-data-item TO DEFAULT.

Otherwise, the rules for transferring XML-value to the receiving data item are determined by the class of the receiving COBOL data item, as described in the following clauses.

14.6a.1.1 Alphabetic receiving items

XML-value is converted from national representation to the runtime alphanumeric coded character set as described in 14.6a.3, Codeset conversion for input data transfer.

The resultant XML-value is a temporary data item of class and category alphabetic of the exact length necessary to hold the entire converted value. That XML-value is moved to the receiving data item in accordance with the rules of the MOVE statement for an elementary sending data item of category alphabetic, as described in 14.8.24.3, MOVE statement, general rule 4c.

14.6a.1.2 Alphanumeric receiving items

XML-value is converted from national representation to the runtime alphanumeric coded character set as described in 14.6a.3, Codeset conversion for input data transfer.

The resultant XML-value is a temporary data item of class and category alphanumeric of the exact length necessary to hold the entire converted value. That XML-value is moved to the receiving data item in accordance with the rules of the MOVE statement for an elementary sending data item of category alphanumeric, as described in 14.8.24.3, MOVE statement, general rule 4a.

14.6a.1.3 Boolean receiving items

XML-value is moved to the receiving data item in accordance with the rules of the MOVE statement for an elementary sending data item of category boolean, as described in 14.8.24.3, MOVE statement, general rule 4d.

14.6a.1.4 National receiving items

XML-value is moved to the receiving data item in accordance with the rules of the MOVE statement for an elementary sending data item of category national, as described in 14.8.24.3, MOVE statement, general rule 4a.

14.6a.1.5 Fixed-point numeric receiving items

The fixed-point numeric value of XML-value is computed and assigned to the receiving data item as though the following statement were executed:

COMPUTE receiving-data-item = FUNCTION NUMVAL-C (XML-value ANYCASE).

NOTE A single currency sign is supported, either the default or a single currency sign defined in the SPECIAL-NAMES paragraph.

14.6a.1.6 Floating-point numeric receiving items

If XML-value is plus infinity (INF), the EC-DATA-INFINITY exception condition is set to exist and the content of the receiving data item is undefined.

If XML-value is negative infinity (-INF), the EC-DATA-NEGATIVE-INFINITY exception condition is set to exist and the content of the receiving data item is undefined.

If XML-value is Not a Number (NaN), the EC-DATA-NOT-A-NUMBER exception condition is set to exist and the content of the receiving data item is undefined.

If XML-value contains an exponent that has a value greater than the implementation supports, the EC-XML-RANGE exception condition is set to exist and the content of the receiving data item is undefined.

If none of these exception conditions is set to exist, the floating-point numeric value of XML-value is computed and assigned to the receiving data item as though the following statement were executed:

```
COMPUTE receiving-data-item = FUNCTION NUMVAL-F (XML-value)
```

14.6a.2 Transfer of output data

Transfer of output data is defined for one sending operand and one receiving operand. The sending operand is a COBOL data item; the receiving operand is a temporary data item of class and category national, XML-value, of the exact length necessary to hold the sending data item.

If the sending data item is of zero-length, XML-value is of zero-length.

The rules for formatting the sending COBOL data item are determined by the class of the sending data item, as described in the following clauses.

Each formatted XML-value is checked for character compliance with the coded character set used for the XML document as described in 14.6a.4, Codeset correspondence for output data transfer.

14.6a.2.1 Alphabetic or alphanumeric sending items

XML-value is trimmed as follows:

- If the sending data item is described with the JUSTIFIED clause, the characters of the sending data item beginning with the leftmost nonspace character and continuing through the rightmost character of the item are assigned to XML-value.
- If the sending data item is described without the JUSTIFIED clause, the characters of the sending data item beginning with the leftmost character and ending with the rightmost nonspace character are assigned to XML-value.

If the trimmed XML-value is of zero length, a single space character is assigned to XML-value.

NOTE The trimmed value is of zero length when the sending item contained only spaces.

The resulting XML-value is converted to usage NATIONAL as described in 14.6a.4, Codeset correspondence for output data transfer.

14.6a.2.2 Boolean sending items

XML-value is used without alteration.

If the resulting XML-value is of zero length, a single boolean value zero is assigned to XML-value.

If the class of XML-value is other than national, XML-value is converted to usage NATIONAL as described in 14.6a.4, Codeset correspondence for output data transfer.

14.6a.2.3 National sending items

XML-value is trimmed as follows:

- If the sending data item is described with the JUSTIFIED clause, the national characters of the sending data item beginning with the leftmost nonspace character and continuing through the rightmost character of the item are assigned to XML-value.
- If the sending data item is described without the JUSTIFIED clause, the national characters of the sending data item beginning with the leftmost character and ending with the rightmost nonspace character are assigned to XML-value.

If the trimmed XML-value is of zero length, a single national space character is assigned to XML-value.

NOTE The trimmed value is of zero length when the sending item contained only spaces.

14.6a.2.4 Fixed-point numeric sending items

If the sending data item is described with a picture character-string that includes the symbol 'P', the sending data item is assigned to a temporary data item that has the same description as the sending data item with each symbol 'P' replaced by a symbol '9'.

If the sending data item is an integer, the value of the item is assigned to a temporary data item that has the following description:

PICTURE -(n)9 USAGE NATIONAL

where n is the greater of one and the number of digits in the value of the sending data item. Assignment is in accordance with the rules of the MOVE statement for a numeric sending operand and a numeric-edited receiving operand.

If the sending data item is not an integer, the value of the item is assigned to a temporary data item that has the following description:

PICTURE -(n)9.9(m) USAGE NATIONAL

where n is the greater of one and the number of digits to the left of the decimal separator; and m is the number of digits to the right of the decimal separator. The DECIMAL-POINT IS COMMA clause does not apply to this picture character-string because XML always uses the period as the decimal separator. Assignment is in accordance with the rules of the MOVE statement for a numeric sending operand and a numeric-edited receiving operand. Trailing zeros to the right of the decimal separator are discarded. If all digits to the right of the decimal separator are zero, the decimal separator is also discarded.

The resulting temporary data item, beginning with the leftmost nonspace character, is assigned to XML-value. XML-value is converted to usage NATIONAL as described in 14.6a.4, Codeset correspondence for output data transfer.

14.6a.2.5 Floating-point numeric sending items

The value of the sending data item is formatted as described for a floating-point literal in 8.3.1.2.2.2, Floating-point numeric literals, and normalized to one significant digit before the decimal point. The sign is included for a signed significand and for a signed exponent.

A zero floating-point value is formatted as 0E+0.

The resulting XML-value is converted to usage NATIONAL as described in 14.6a.4, Codeset correspondence for output data transfer.

14.6a.3 Codeset conversion for input data transfer

When the coded character set used for the internal representation of the XML document (UTF-16) differs from the runtime coded character set used for a given data item, XML-value is converted to the runtime coded character set used for the data item before transferring XML-value to the receiving data item. If no correspondence exists for a given character, an implementor-defined substitution character is used as the corresponding character and the EC-XML-CODESET-CONVERSION exception condition is set to exist.

If a RESUME statement that has the NEXT STATEMENT phrase is executed from a declarative associated with the EC-XML-CODESET-CONVERSION exception condition, processing resumes with the next XML-value, if any.

14.6a.4 Codeset correspondence for output data transfer

After formatting and conversion to usage NATIONAL, the characters of each XML-value are checked for correspondence with the coded character set used for the XML document. If no correspondence exists for a given character, an implementor-defined substitution character is used as the corresponding character and the EC-XML-CODESET-CONVERSION exception condition is set to exist.

NOTE Conversion is expected to be successful for all runtime coded character sets when the output coded character set is UTF-8 or UTF-32. Conversion is expected to be successful for all runtime coded character sets when the output coded character set is UTF-16 when no surrogate pairs are used.

If the implementation cannot determine or verify the identity of the coded character set used for an XML document, an implementor-defined default is assumed.

If a RESUME statement that has the NEXT STATEMENT phrase is executed from a declarative associated with the EC-XML-CODESET-CONVERSION exception condition, processing resumes with the next sending data item, if any."

[e] CLOSE statement, 14.8.6:

[1] Add the following to the initial paragraph of the CLOSE statement, 14.8.6:

"The XML-document format CLOSE statement completes the processing of an XML document or portion of a document."

[2] 2. Label the existing general format of the CLOSE statement as "Format 1 (file)" and add the following:

"Format 2 (XML-document)

CLOSE DOCUMENT file-name-1 [WITH DISCARD]"

[3] Add the heading "FORMAT 1" before the existing syntax rules and add the following to the syntax rules of the CLOSE statement, 14.8.6.2:

"3) The LOCK phrase shall not be specified for an XML file.

FORMAT 2

4) File-name-1 shall have organization XML."

[4] General rules 1 and 4 apply to both formats. The other existing general rules apply only to format 1.

[5] Add the following new general rule for format 1 of the CLOSE statement:

"11) If the file connector referenced by file-name-1 has organization XML and an XML-document format CLOSE statement has not been executed for the file connector referenced by file-name-1 since an XML-document format OPEN statement was executed for that file connector, an implicit XML-document format CLOSE statement with the DISCARD phrase is executed for that file connector and the EC-XML-IMPLICIT-CLOSE exception condition is set to exist before the file format CLOSE statement is executed. If this exception condition results in the execution of a RESUME statement with the NEXT STATEMENT phrase, processing resumes with continuation of the execution of this CLOSE statement."

[6] Add the following general rules for format 2:

"12) If the file connector referenced by file-name-1 is not open, the I-O status associated with file-name-1 is set to '4B' and the execution of the CLOSE statement is unsuccessful.

13) If an XML document in the file referenced by file-name-1 is not open, the I-O status associated with file-name-1 is set to '4D' and the execution of the CLOSE statement is unsuccessful.

14) If the file connector referenced by file-name-1 is open in the extend, i-o, or output mode, the document is checked to verify that it is well-formed. If the XML document that would be written is not well-formed, the I-O status associated with file-name-1 is set to '4A' and the execution of the CLOSE statement is unsuccessful.

NOTE Unless the RAW phrase is specified in an IDENTIFIED clause in a data description subordinate to file-name-1, the fact that the XML would not be well-formed would have been detected during the execution of a WRITE statement or a REWRITE statement.

NOTE 2 A programmer can use CDATA in an item described with the RAW phrase to create XML data that would otherwise not be valid.

15) If the file connector referenced by file-name-1 is open in the extend, i-o, or output mode and the OUTPUT phrase is specified in the TYPE clause of the file control entry for file-name-1, the specified DTD or schema is used to check the document for validity. If the XML document that would be written is not valid, the I-O status associated with file-name-1 is set to '3B' and the execution of the CLOSE statement is unsuccessful.

16) If the DISCARD phrase is specified, the changes made to the internal representation are not made to the file referenced by file-name-1, and the following processing occurs:

a) If the STACK phrase was specified on the most recent XML-document format OPEN statement that was executed for the file connector referenced by file-name-1, the internal representation and

the element position vector are restored to the state they had at the start of the execution of that OPEN statement.

b) If the STACK phrase was not specified on the most recent XML-document format OPEN statement that was executed for the file connector referenced by file-name-1, the internal representation is unavailable and the element position vector is set to indicate that no valid position has been established.

17) If the DISCARD phrase is not specified, the following processing occurs:

a) If the STACK phrase was specified on the most recent XML-document format OPEN statement that was executed for the file connector referenced by file-name-1, the internal representation and the element position vector are restored to the state they had when that OPEN statement was initiated, except that when the file connector referenced by file-name-1 is open in the extend, i-o, or output mode any changes made to the internal representation since that OPEN statement are made to the restored internal representation.

b) If the STACK phrase was not specified on the most recent XML-document format OPEN statement that was executed for the file connector referenced by file-name-1, the following processing occurs:

1. If the file connector referenced by file-name-1 is open in the extend, i-o, or output mode, an XML document is created from the internal representation and released to the operating environment.

NOTE 1 The content of the XML document can be affected by the CODE-SET clause in the file description, the VERSION-XML clause in the file control entry, and the NAMESPACE phrases of the IDENTIFIED clauses of data items subordinate to an XML file description.

NOTE 2 Updates to data items subordinate to a file description entry are not included in the document if they were not written with XML-element format WRITE or XML-element format REWRITE statements.

If the XML document is in-memory and is larger than the associated length of that memory, the I-O status associated with file-name-1 is set to '24' and the execution of the CLOSE statement is unsuccessful.

2. The internal representation is unavailable and the element position vector is set to indicate that no valid position has been established.

18) The execution of the CLOSE statement does not close the file.

19) The file position indicator is not affected by the execution of a CLOSE statement.

20) If the execution of the CLOSE statement is unsuccessful, no update of the file takes place, the element position vector is unchanged, and the content of the internal representation is unchanged.

21) If the TYPE clause is specified in the file control entry for file-name-1, the following is written to the file when a CLOSE statement is executed:

a) if the DTD phrase is specified without the EXTERNAL clause, a DTD that describes the document that is contained in the internal representation;

b) if the DTD phrase is specified with the EXTERNAL clause, a reference to the specified external DTD;

c) if the SCHEMA phrase is specified, a reference to the specified schema."

[f] Add the following second paragraph to 14.8.9, DELETE statement:

"The DELETE statement deletes an XML attribute or element from the internal representation of an XML document. If an element is deleted, all its attributes and subordinate elements are deleted."

[g] In the general format of the DELETE statement, 14.8.9.1, General format:

[1] Add the heading "FORMAT 1 (relative-indexed)" before the current general format.

[2] Add the following after the current general format:

"Format 2 (XML):

```
DELETE file-name-1 RECORD

{ ATTRIBUTE }
{ ELEMENT   }  data-name-1

[| INVALID KEY imperative-statement-1       |]
[| NOT INVALID KEY imperative-statement-2   |]

[ END-DELETE ] "
```

[h] In the syntax rules of the DELETE statement, 14.8.9.2, Syntax rules:

[1] Add the heading "FORMAT 1" before the current syntax rules.

[2] Add the following after the current syntax rules:

"FORMAT 2:

3) File-name-1 shall reference an XML file.

4) The description of data-name-1 shall include an IDENTIFIED clause and shall be subordinate to the description of file-name-1.

5) If the ATTRIBUTE phrase is specified on the DELETE statement, the ATTRIBUTE phrase shall be specified in the IDENTIFIED clause in the description of data-name-1.

6) If the ELEMENT phrase is specified, the ELEMENT phrase shall be explicitly or implicitly specified in the IDENTIFIED clause in the description of data-name-1."

[i] In the general rules of the DELETE statement, 14.8.9.3, General rules:

[1] General rules 9 through 11 apply to both formats.

[2] Add the following new general rules:

"FORMAT 2:

12) The file connector referenced by file-name-1 shall be open in i-o mode.

13) After the successful execution of a DELETE statement with the ELEMENT phrase, the element identified by data-name-1, along with all of its attributes and child elements, is removed from the internal representation of the document and is no longer accessible to the program. All other elements in the document are unchanged.

14) After the successful execution of a DELETE statement with the ATTRIBUTE phrase, the attribute identified by data-name-1 is removed from the internal representation of the document and is no longer accessible to the program. All other attributes in the document are unchanged.

15) The content of the element position vector at the start of the execution of the DELETE statement is used to determine which attribute or element is to be deleted in accordance with the following rules:

a) If the ELEMENT phrase is specified and the element position vector was established by a prior READ or START statement, the element that is deleted is the element associated with the data item referenced by data-name-1 that is indicated by the element position vector.

b) If the ATTRIBUTE phrase is specified and the element position vector was established by a prior READ or START statement, the attribute that is deleted is the attribute associated with the data item referenced by data-name-1 that is indicated by the element position vector.

If an attribute or element is found that satisfies this general rule, that attribute or element is deleted.

If the element position vector indicates that no valid position has been established or if no attribute or element is found that satisfies this general rule, the DELETE statement is unsuccessful, the I-O status value associated with file-name-1 is set to '25', and the invalid key condition is set to exist."

[j] In syntax rule 9 of the MERGE statement, 14.8.23.2, change in part to read:

"… file description entry that is not for an XML file and that is not for a report …"

[k] 14.8.26, OPEN statement, add a second paragraph as follows:

"The OPEN statement creates an internal representation of an XML document."

[l] 14.8.26.1, OPEN statement,

[1] Label the existing format of the OPEN statement as "Format 1 (file):"

[2] Add " [END-OPEN] " to the end of format 1.

[3] Add the following new format:

"Format 2 (XML-document):

OPEN DOCUMENT file-name-1 [AT identifier-1 [STACK]]

[RETURNING identifier-2]

[| AT END imperative-statement-1 |]
[| NOT AT END imperative-statement-2 |]

[END-OPEN] "

[m] In the syntax rules of the OPEN statement, 14.8.26.2:

[1] Add the heading "FORMAT 1" before the existing syntax rules.

[2] Change syntax rule 2 to read:

"2) The EXTEND phrase shall be specified only if either:

a) the access mode of the file connector referenced by file-name-1 is sequential and the LINAGE clause is not specified in the file description entry for file-name-1, or

b) the access mode of the file connector referenced by file-name-1 is XML."

[3] Add the following after the existing syntax rules:

"FORMAT 2

8) File-name-1 shall have XML organization.

9) The description of identifier-1 shall include an IDENTIFIED clause with the explicit or implicit ELEMENT phrase and shall be subordinate to the description of file-name-1.

10) Identifier-2 shall reference an alphanumeric or national data item."

[j] In the general rules of the OPEN statement, 14.8.26.3:

[1] Insert the heading "FORMAT 1" before the first general rule.

[2] First sentence of general rule 2, insert after "file-name-1 with a" the following text:

"memory location or "

[3] 3. Append the following to the bottom of Table 21, Permissible I-O statements by access mode and open mode:

[Access mode Statement Input Output I-O Extend]

"

XML	READ	X		X	
	WRITE		X	X	X
	REWRITE			X	
	START	X		X	
	DELETE			X	
	OPEN DOCUMENT	X	X	X	X
	CLOSE DOCUMENT	X	X	X	X

"

[4] Add the following new last sentence to general rule 12:

"When the organization is XML, the file position indicator is set to indicate the first document in the file."

[5] Add the following new last sentence to general rule 13:

"The last logical record for an XML file is the last document in the file."

[6] Add the following the end of the general rules of the OPEN statement:

"26) If the INPUT, I-O, or EXTEND phrase is specified on the OPEN statement and a data-name is specified in the TYPE clause for file-name-1, execution of the OPEN statement moves the schema or DTD name specified in the XML document into the data item referenced by that data-name. If a literal is specified in the TYPE clause for file-name-1, execution of the OPEN statement checks that that literal matches the schema or DTD specified in the XML document and if not, the EC-XML-DOCUMENT-TYPE exception condition is set to exist.

27) If the CHECK phrase is specified in the TYPE clause of the file control entry for file-name-1 and the specified DTD or schema is not available or is not valid, the I-O status value associated with file-name-1 is set to '09' and no validity check of the document is performed. The implementor defines the conditions that determine whether a DTD or schema is available.

28) If the I-O phrase is specified and the file referenced by file-name-1 has organization XML, the file shall contain at most one document. If the file contains multiple documents, the I-O status value associated with file-name-1 is set to '3C' and the OPEN statement is unsuccessful.

FORMAT 2

29) If the file connector referenced by file-name-1 is not open, the execution of the OPEN statement is unsuccessful and the I-O status associated with file-name-1 is set to '4B'.

30) If identifier-1 is not specified and an XML-document format OPEN statement that specifies file-name-1 has been executed and no XML-document format CLOSE statement that specifies file-name-1 has been subsequently executed, an implicit XML-format CLOSE statement with the DISCARD phrase is executed for that file connector and the EC-XML-IMPLICIT-CLOSE exception condition is set to exist before the open operation is executed. If this exception condition results in the execution of a RESUME statement with the NEXT STATEMENT phrase, processing resumes with continuation of the execution of this OPEN statement.

31) If identifier-1 is not specified and the file connector referenced by file-name-1 is open in the input or i-o mode, the OPEN statement creates an internal representation of the document. The value of the file position indicator at the start of the execution of the OPEN statement is used to determine the root node of the XML document that is made available in accordance with the following rules:

a) If the file position indicator indicates that no valid position has been established or that an optional input file is not present, execution of the OPEN statement is unsuccessful.

b) If the file position indicator was established by a prior OPEN statement, the document that is selected is the first document in the physical file after the location indicated by the file position indicator.

If the implementation cannot determine the coded character set that is used for the document, the OPEN statement is unsuccessful and the I-O status value is set to '3D'.

If the document is not well-formed, the I-O status value associated with file-name-1 is set to '3A' and the execution of the OPEN statement is unsuccessful. A document is terminated by the XML declaration of the next document, the root tag of the next document, or the end of the file. Processing instructions, comments and white space between documents are ignored by the COBOL processor.

If the INPUT phrase is specified in the TYPE clause in the description of file-name-1, the specified DTD or schema is available, and the document does not conform to the specified DTD or schema, the EC-XML-INVALID exception condition is set to exist.

If a document is found that satisfies this general rule, the document is made available and the file position indicator is set to indicate the document made available.

If no document is found that satisfies this general rule, the file position indicator is set to indicate that no next record exists, the I-O status value associated with file-name-1 is set to '10', the at end condition exists, and execution proceeds as specified in general rule 35.

32) If identifier-1 is specified, the data item referenced by identifier-1 shall be associated by the element position vector with a valid element position in the internal representation; otherwise the EC-XML-STACKED-OPEN exception condition is set to exist and the execution of the OPEN statement is unsuccessful. The position in the internal representation associated with the data item referenced by identifier-1 provides the root node and its subordinate structure for the new internal representation of file-name-1. If the STACK phrase is specified, the previous internal representation and element position vector are preserved in a stack so that they can be restored when an XML-document format CLOSE statement referencing file-name-1 is executed; this previous internal representation cannot be accessed until an XML-document format CLOSE statement is executed for file-name-1. If the STACK phrase is not specified, the previous internal representation and element position vector are discarded.

33) If the file connector referenced by file-name-1 is open in the input or i-o mode, the element position vector is set to indicate the first XML element that is associated with each COBOL data item that has an IDENTIFIED clause and to the first attribute, if any, of each of those elements when that attribute is associated with a COBOL data item that has an IDENTIFIED clause.

34) If the at end condition does not occur during the execution of an OPEN statement, the AT END phrase is ignored, if specified, and the following actions occur:

a) The I-O status associated with file-name-1 is updated and, if the record operation conflict condition did not occur, the file position indicator is set.

b) If an exception condition that is not an at end condition exists, control is transferred in accordance with the following rules:

1. If a USE AFTER EXCEPTION procedure is associated with the file connector referenced by file-name-1, control is transferred in accordance with the rules for the specific exception condition and the rules for the USE statement. If this exception condition results in the execution of a RESUME statement with the NEXT STATEMENT phrase, processing resumes with continuation of the execution of this OPEN statement.

2. If there is no USE AFTER EXCEPTION procedure associated with the file connector referenced by file-name-1, control is transferred in accordance with the rules for the specific exception condition, except that when the exception condition is not a fatal exception condition, processing resumes with continuation of the execution of this OPEN statement.

c) If no exception condition exists, the internal representation of the XML document is created. Control is transferred to the end of the OPEN statement, or to imperative-statement-2, if specified. If a procedure branching or conditional statement that causes explicit transfer of control is executed, control is transferred in accordance with the rules for that statement; otherwise, upon completion of the execution of imperative-statement-2, control is transferred to the end of the OPEN statement.

35) If the at end condition exists, the following occurs in the order specified:

a) The I-O status value associated with file-name-1 is set to '10' to indicate the at end condition.

b) If the AT END phrase is specified in the statement that caused the condition, control is transferred to the AT END imperative-statement-1. Any USE AFTER EXCEPTION procedure associated with the file connector referenced by file-name-1 is not executed. Execution then continues in accordance with the rules for each statement specified in imperative-statement-1. If a procedure branching or conditional statement that causes explicit transfer of control is executed, control is transferred in accordance with the rules of that statement; otherwise, upon completion of the execution of imperative-statement-1, control is transferred to the end of the OPEN statement and the NOT AT END phrase, if specified, is ignored.

c) If the AT END phrase is not specified and a USE AFTER EXCEPTION procedure is associated with the file connector referenced by file-name-1, that procedure is executed. Control is then transferred to the end of the OPEN statement. The NOT AT END phrase is ignored, if it is specified.

d) If the AT END phrase is not specified and there is no USE AFTER EXCEPTION procedure associated with the file connector referenced by file-name-1, control is transferred to the end of the OPEN statement. The NOT AT END phrase is ignored if it is specified.

When the at end condition exists, execution of the OPEN statement is unsuccessful.

36) Regardless of the method used to overlap access time with processing time, the concept of the OPEN statement is unchanged; the internal representation is available to the runtime element prior to the execution of imperative-statement-2, if specified, or prior to the execution of any statement following the OPEN statement, if imperative-statement-2 is not specified.

37) Unless otherwise specified, at the completion of any unsuccessful execution of an OPEN statement, the content of the internal representation is undefined, the element position vector is set to indicate that no valid element position has been established, and the file position indicator is set to indicate that no valid record position has been established.

38) If the RETURNING phrase is specified and the open mode is INPUT or I-O, the name of the root element of the document is moved to the data item referenced by identifier-2."

[n] Add the following last sentence to 14.8.29, READ statement:

"For XML access, the READ statement transfers data content associated with a specified XML attribute or element to the associated COBOL record area. "

[o] In the general format of the READ statement, 14.8.29.1, add the following after the current general formats:

"Format 3 (XML-element):

```
READ file-name-1

{ ATTRIBUTE          }
{ [ ONLY ] ELEMENT   }  data-name-2

[| AT END imperative-statement-1       |]
[| NOT AT END imperative-statement-2  |]

[ END-READ ] "
```

[p] In the syntax rules of the READ statement, 14.8.29.2, Syntax rules:

[1] Change the heading "ALL FORMATS" to "FORMATS 1 AND 2".

[2] Add the following after the current syntax rules:

"FORMAT 3:

12) File-name-1 shall have XML organization.

13) Data-name-2 may be qualified.

14) The description of data-name-2 shall include an IDENTIFIED clause and shall be subordinate to the description of file-name-1.

15) If the ATTRIBUTE phrase is specified on the READ statement, the ATTRIBUTE phrase shall be specified in the IDENTIFIED clause in the description of data-name-2.

16) If the ELEMENT phrase is specified, the ELEMENT phrase shall be explicitly or implicitly specified in the IDENTIFIED clause in the description of data-name-2."

[q] In the general rules of the READ statement, 14.8.29.3, General rules:

[1] General rules 1, 2, 12, 14, 15, and 21 apply to both this new format and the existing formats.

[2] add the following after the existing general rules:

"FORMAT 3:

30) The READ statement finds the internal counterpart associated with the data item specified by data-name-2 and transfers the content of the next occurrences of that associated attribute or element from the internal representation to the data item associated with the data content of the data item referenced by data-name-2 in accordance with general rules 31 through 36. If the ELEMENT phrase is specified without the ONLY phrase, the READ statement also transfers data from any subordinate XML elements and their attributes into the associated subordinate data items, moving as much data as possible subject to the occurrence limitations of the associated data items. Data items specified in any associated COUNT clauses are updated to indicate the number of occurrences, if any, moved into the associated data items in the COBOL record.

If there is no next occurrence, the I-O status value associated with file-name-1 is set to '10', the at end exception condition is set to exist, and the READ statement is unsuccessful.

NOTE When there are more associated attributes or elements in the document than fit into the COBOL record, the application can perform successive reads that specify the associated COBOL data item until the at end condition exists. This can be repeated at each level where multiple attributes or elements are possible.

31) The content of the element position vector at the start of the execution of the READ statement is used to determine the XML data that is made available in accordance with the following rules:

a) If the element position vector indicates that no valid position has been established, execution of the READ statement is unsuccessful and execution proceeds as specified in the last paragraph of this general rule.

b) If the ELEMENT phrase is specified and the element position vector was established by a prior READ statement with an ELEMENT phrase and an ONLY phrase that specifies a data item superordinate to data-name-2, an OPEN statement, or a START statement, the element that is selected is the first element associated with the data item referenced by data-name-2 that has a location greater than or equal to the location indicated by the element position vector for that level and that is subordinate to the current groups that are indicated by the element position vector.

c) If the ELEMENT phrase is specified and the element position vector was established by a prior READ statement that specified data-name-2 in the ELEMENT phrase or that specified an ELEMENT phrase without the ONLY phrase that specified a data item superordinate to data-name-2, the element that is selected is the first element associated with the data item referenced by data-name-2 that has a location greater than the location indicated by the element position vector for that level and that is subordinate to the current groups that are indicated by the element position vector.

d) If the ATTRIBUTE phrase is specified and the IDENTIFIED clause for data-name-2 specifies the BY phrase, the attribute that is selected is the attribute associated with the data item referenced by data-name-2 that is subordinate to the current groups that are indicated by the element position vector.

e) If the ATTRIBUTE phrase is specified and the IDENTIFIED clause for data-name-2 specifies the USING phrase and the element position vector was established by a prior READ statement with an ELEMENT phrase, an OPEN statement, or START statement, the attribute that is selected is the first existing attribute associated with the data item referenced by data-name-2 that has a location greater than or equal to the location indicated by the element position vector for that level and that is subordinate to the current groups that are indicated by the element position vector.

f) If the ATTRIBUTE phrase is specified and the IDENTIFIED clause for data-name-2 specifies the USING phrase and the element position vector was established by a prior READ statement with an ATTRIBUTE phrase, the attribute that is selected is the first existing attribute associated with the data item referenced by data-name-2 that has a location greater than the location indicated by the element position vector for that level and that is subordinate to the current groups that are indicated by the element position vector.

If an attribute or element is found that satisfies this general rule, the attribute or element is made available and the element position vector is set to indicate the attribute or element made available.

If the ELEMENT phrase is specified and the satisfying element has subordinate elements:

- If the ONLY phrase is specified, the element position vector is set to indicate the first element associated with each subordinate COBOL data item that has an IDENTIFIED clause and to the first attribute, if any, of each of those elements.
- If the ONLY phrase is not specified, the element position vector is set to indicate the last attribute or element that is made available for each subordinate COBOL data item that has an IDENTIFIED clause.

If no attribute or element is found that satisfies this general rule, the element position vector is set to indicate that no next element exists, the I-O status value associated with file-name-1 is set to '10', the at end condition exists, and execution proceeds as specified in general rule 21.

32) If the description of the data item referenced by data-name-2 contains an OCCURS clause, the number of table elements made available is the maximum number allowed by the OCCURS clause or the number of unread associated XML attributes or elements, whichever is less.

33) If data-name-2 is described with an explicit or implicit ELEMENT phrase without the RAW phrase in its IDENTIFIED clause, any attributes or elements in the XML document subordinate to the element associated with data-name-2 that have no associated data items in the COBOL record are ignored. If an attribute or element is ignored, the data associated with it is ignored. If the ELEMENT phrase is specified without the ONLY phrase in the READ statement and attributes or elements are ignored and there is no error that causes the execution of the READ statement to be unsuccessful, the execution of the READ statement is successful and the I-O status value associated with file-name-1 is set to '08'.

34) If a group item is described with an IDENTIFIED clause, any directly subordinate data item without an IDENTIFIED clause in its description that is not referenced by an IDENTIFIED clause on the group item is associated with the character data content of the identified XML attribute or element.

If a group item described with an IDENTIFIED clause has no such associated subordinate item, the character data associated with the attribute or element described by the IDENTIFIED clause is ignored.

35) When the READ statement has made XML data available, a single XML-value exists for each attribute or element that is made available. XML-values are transferred one at a time to the data item associated with the character data content of the data item referenced by data-name-2 and to any associated subordinate items in hierarchical order of the description of data-name-2 in accordance with the following rules:

a) When either the ATTRIBUTE phrase or the ONLY phrase is specified in the READ statement, the attribute's or element's XML-value is transferred to the associated data item as described in 14.6a.1, Transfer of input data, and processing continues at general rule 36.

b) When the ELEMENT phrase is specified and the ONLY phrase is not specified in the READ statement,

1. The XML-value of an element at a given level is transferred to the associated data item as described in 14.6a.1, Transfer of input data. For a multiple-occurrence data item, this step is repeated for each occurrence. For each attribute of each of these elements, the XML-value is transferred to the associated data item as described in 14.6a.1, Transfer of input data.
2. Step 35b1 repeats for each data item at a given level and for each level subordinate to data-name-2. Any such data item that does not have an associated attribute or element in the XML document is initialized as though the statement INITIALIZE data-item TO DEFAULT were executed.

NOTE Because the element position vector maintains position for each attribute or element in the COBOL record description that is described with an IDENTIFIED clause, the order of attributes and elements in the XML document need not match the order of the associated attributes and elements in the record description.

36) If the READ statement triggers execution of a RESUME statement with the NEXT STATEMENT phrase in a declarative associated with an EC-DATA-INFINITY, EC-DATA-INCOMPATIBLE, EC-DATA-NEGATIVE-INFINITY, EC-DATA-NOT-A-NUMBER, or EC-XML-CODESET-CONVERSION exception condition, processing resumes with the next XML-value, if any, and execution of the READ statement is successful."

[r] Add the following new paragraph to 14.8.33, REWRITE statement:

"The REWRITE statement modifies an XML element in the internal representation of an XML document."

[s] In the general format of the REWRITE statement, 14.8.33.1, General format:

[1] Add the heading "FORMAT 1 (sequential-relative-indexed)" before the current general format.

[2] Add the following after the current general format:

"Format 2 (XML-element):

```
REWRITE record-name-1

    { ATTRIBUTE          }
    { [ ONLY ] ELEMENT   }  data-name-1

    [| INVALID KEY imperative-statement-1       |]
    [| NOT INVALID KEY imperative-statement-2  |]

[ END-REWRITE ] "
```

[t] In the syntax rules of the REWRITE statement, 14.8.33.2, Syntax rules:

[1] Syntax rule 4 applies to both formats.

[2] Add the following after the current syntax rules:

"FORMAT 2:

13) Record-name-1 shall reference a record subordinate to a file description for a file with XML organization.

14) The description of data-name-1 shall include an IDENTIFIED clause and shall be either record-name-1 or subordinate to the description of record-name-1.

15) If the ATTRIBUTE phrase is specified on the REWRITE statement, the ATTRIBUTE phrase shall be specified in the IDENTIFIED clause in the description of data-name-1.

16) If the ELEMENT phrase is specified, the ELEMENT phrase shall be explicitly or implicitly specified in the IDENTIFIED clause in the description of data-name-1."

[u] In the general rules of the REWRITE statement, 14.8.33.3, General rules:

[1] Old general rules 1, 2, 12, 13, and 14 apply to both formats.

[2] Add the following after the current general rules:

"FORMAT 2:

25) Execution of the REWRITE statement updates the internal representation of the XML document and does not modify the file. The attribute or element associated with data-name-1, and optionally child elements, is updated. All other attributes and elements in the internal representation are unchanged. The element position vector indicates the position of the referenced attribute or element within the hierarchy of the document. If there is no attribute or element associated with the data item referenced by data-name-1 so located, the invalid key condition exists, the execution of the REWRITE statement is unsuccessful and the I-O status associated with the rewrite file connector is set to the invalid key condition '23'.

The content of the data item associated with the character data content of the specified XML attribute or element and its attributes are formatted as described in 14.6a.2, Transfer of output data, and are updated in the internal representation of the XML document. If the ONLY phrase is specified, the contained elements and their attributes are unchanged; otherwise, the contained elements and attributes are also individually formatted and updated.

If the ONLY phrase is not specified, each data item that is directly or indirectly subordinate to the data item referenced by data-name-1 and that has an IDENTIFIED clause in its description is treated as specified for data-name-1 in the previous paragraphs of this rule. This treatment is done in the order that the data items are specified in the COBOL record description.

NOTE The order in which elements are rewritten may differ from the order in the original document.

Before an attribute or element is transferred to the internal representation, the characters of the element name or attribute name are checked for correspondence with the coded character set used for the XML document. If no correspondence exists for a given character, the I-O status value is set to '4E', the attribute or element is not transferred, no further elements or attributes are transferred, and the REWRITE statement is unsuccessful.

If the RAW phrase is not specified in the IDENTIFIED clause in the description of data-name-1 and the resultant XML is not well-formed, the I-O status associated with the rewrite file connector is set to '4A' and the execution of the REWRITE statement is unsuccessful.

If an EC-DATA-INCOMPATIBLE exception condition or an EC-XML-CODESET-CONVERSION exception condition is set to exist during processing of an XML attribute or element and a RESUME statement with the NEXT STATEMENT phrase is executed in an associated declarative, processing resumes with the next attribute or element, if any, and execution of the REWRITE statement is successful.

26) When a data-name is specified in the IDENTIFIED clause in the description of data-name-1 or one of its subordinate items and the value of that data item changes between the execution of the READ statement and the execution of the REWRITE statement, an XML attribute or element with the new name replaces the previous XML attribute or element."

[v] In the syntax rules of the SORT statement, 14.8.36.2,

[1] Syntax rule 8, change in part to read:

"… file description entry that is not for an XML file and that is not for a report …"

[2] Add the following new syntax rule for the format 2 SORT statement:

"13a) Data-name-2 shall not be subordinate to a record description of an XML file."

[w] In the general format of the START statement, 14.8.37.1, General format:

[1] Add the heading "FORMAT 1 (sequential-relative-indexed)" before the current general format.

[2] Add the following after the current general format:

"Format 2 (XML):

```
START file-name-1

{ ATTRIBUTE } data-name-2 [ INDEX IS { data-name-3 } ]
{ ELEMENT  }              [          { integer-1   } ]
                          [ KEY EQUAL { data-name-4 } [ WITH LENGTH arithmetic-expression-1 ] ]
                          [           { literal-1   } ]

[| INVALID KEY imperative-statement-1     |]
[| NOT INVALID KEY imperative-statement-2 |]

[ END-START ]"
```

[x] In the syntax rules of the START statement, 14.8.37.2, Syntax rules:

[1] Add the heading "FORMAT 1" before the current syntax rules.

[2] Add the following after the current syntax rules:

"FORMAT 2:

8) File-name-1 shall have organization XML.

9) Data-name-2, data-name-3, and data-name-4 may be qualified.

10) The description of data-name-2 shall include an IDENTIFIED clause and shall be subordinate to the description of file-name-1.

11) Data-name-3 shall reference an integer.

12) The data item referenced by data-name-4 and the value of literal-1 shall have the same class and category as the data item that receives the data associated with data-name-2 and, if the LENGTH phrase is not specified, shall have a length that is not greater than the length of the data item that receives the data associated with data-name-2.

13) The LENGTH phrase shall not be specified if literal-1 is numeric.

14) If the ATTRIBUTE phrase is specified on the START statement, the ATTRIBUTE phrase shall be specified in the IDENTIFIED clause in the description of data-name-2.

15) If the ELEMENT phrase is specified on the START statement, the ELEMENT phrase shall be explicitly or implicitly specified in the IDENTIFIED clause in the description of data-name-2.

16) If the ATTRIBUTE phrase is specified, the INDEX phrase shall not be specified."

[y] In the general rules of the START statement, 14.8.37.3, General rules:

[1] Add the following after the current general rules:

"XML FILES

22) The element position vector is set to indicate an attribute or element with the attribute name or element name, respectively, specified in the IDENTIFIED clause of the description of data-name-2. If the ELEMENT phrase is specified and data-name-2 references a group item, the element position vector is set to indicate the first XML element associated with each subordinate COBOL data item that has an IDENTIFIED clause and to the first attribute, if any, of each of those elements. ATTRIBUTE and ELEMENT phrases in subsequent statements that reference a data item subordinate to the indicated element do so within that occurrence of the element.

If the specified occurrence of the element does not exist, the I-O status associated with file-name-1 is set to '23', the invalid key condition exists, the element position vector is set to indicate that no valid position has been established, and the execution of the START statement is unsuccessful.

23) If neither the INDEX phrase nor the KEY phrase is specified, execution of the START statement sets the element position vector to indicate the first occurrence of an attribute or element with the name specified in the IDENTIFIED clause of the description of data-name-2 that is at the correct level within all superordinate groups indicated by the element position vector.

24) If the INDEX phrase is specified, integer-1 or the value of the data item referenced by data-name-3 specifies an ordinal position among the elements whose tag name is equal to the value specified in the IDENTIFIED clause of the description of data-name-2. The element position vector is set to indicate this element.

25) If the KEY phrase is specified:

a) If the LENGTH phrase is specified, the value of arithmetic-expression-1 specifies the number of character positions to be used in the comparison, with padding or truncation of the operands as necessary.

If arithmetic-expression-1 does not evaluate to a positive non-zero integer, the I-O status associated with file-name-1 is set to '23', the invalid key condition exists, and the execution of the START statement is unsuccessful.

b) Data associated with an attribute or element whose attribute name or element name, respectively, is equal to the value specified in the IDENTIFIED clause of the description of data-name-2 is compared for equality with the content of the data item referenced by data-name-4 or the value of literal-1. Comparison proceeds as specified for items of the class of data-name-2 in 8.8.4.1.1.6, Comparison of alphanumeric operands, or 8.8.4.1.1.8, Comparison of national operands. Then, either:

1. The element position vector is set to indicate the first attribute or element that satisfies the comparison, or
2. If the comparison is not satisfied by any attribute or element, the invalid key condition exists and the execution of the START statement is unsuccessful.

NOTE 1 Execution of the START statement sets the element position vector such that a subsequent XML-element format READ statement reads the specified occurrence of the data item referenced by data-name-2.

NOTE 2 If a START statement is followed immediately by an XML-element format WRITE statement, the element is inserted into the internal representation immediately before the element that would have been read if a START statement had been followed by an XML-element format READ statement.

NOTE 3 To insert an element after the element specified in the START statement, instead of before it, execute a READ statement that specifies the same element as specified in the START statement before executing the WRITE statement."

[z] Add the following new second paragraph to 14.8.47, WRITE statement:

"The WRITE statement adds an XML element to the in-memory representation of an XML document."

[aa] In the general format of the WRITE statement, 14.8.47.1, General format, add the following after the current general format:

"Format 3 (XML-element):

```
WRITE record-name-1

    { ATTRIBUTE                }
    { [ ONLY ]  ELEMENT        }  data-name-1

    [| INVALID KEY imperative-statement-1      |]
    [| NOT INVALID KEY imperative-statement-2  |]

[ END-WRITE ]"
```

[bb] Add the following syntax rules to the WRITE statement, 14.8.47.2, Syntax rules:

"22) If the organization of the write file is XML, format 3 shall be specified.

23) The description of data-name-1 shall include an IDENTIFIED clause and shall be either record-name-1 or subordinate to the description of record-name-1.

24) If the ATTRIBUTE phrase is specified on the WRITE statement, the ATTRIBUTE phrase shall be specified in the IDENTIFIED clause in the description of data-name-1.

25) If the ELEMENT phrase is specified, the ELEMENT phrase shall be explicitly or implicitly specified in the IDENTIFIED clause in the description of data-name-1."

[cc] Add the following new general rules to the WRITE statement, 14.8.47.3:

"XML FILES

38) Execution of the XML-element format WRITE statement manipulates the internal representation of the XML document and does not modify the file. The attribute or element associated with data-name-1 is inserted immediately before the attribute or element, respectively, at the position indicated by the element position vector at the level associated with data-name-1. Any containing elements that are required for the specified attribute or element are also inserted into the internal representation. The specified element, its attributes, and the data items in the COBOL record associated with the character data content of those attributes and elements are formatted as described in 14.6a.2, Transfer of output data, and are added to the internal representation of the XML document. If the ONLY phrase is specified, contained elements and their attributes are not added; otherwise, the contained elements and their attributes are also individually formatted and added.

If the ONLY phrase is not specified, each data item that is directly or indirectly subordinate to the data item referenced by data-name-1 and that has an IDENTIFIED clause in its description is treated as

specified for data-name-1 in the previous paragraphs of this rule. This treatment is done in the order that the data items are specified in the COBOL record description

Before an attribute or element is transferred to the internal representation, the characters of the element name or attribute name are checked for correspondence with the coded character set used for the XML document. If no correspondence exists for a given character, the I-O status value is set to '4E', the attribute or element is not transferred, no further elements or attributes are transferred, and the WRITE statement is unsuccessful.

If the RAW phrase is not specified in the IDENTIFIED clause in the description of data-name-1 and the resultant XML is not well-formed, the I-O status associated with the write file connector is set to '4A' and the execution of the WRITE statement is unsuccessful.

If an EC-DATA-INCOMPATIBLE exception condition or an EC-XML-CODESET-CONVERSION exception condition occurs during processing of an XML attribute or element and a RESUME statement with the NEXT STATEMENT phrase is executed in an associated declarative, processing resumes with the next attribute or element, if any, and the execution of the WRITE statement is successful."

6.6 Changes to Annex F (informative) Substantive changes list

[a] In F.1, Substantives changes potentially affecting existing programs, item 6, Reserved words, add the following to the list in alphabetical order:

DOCUMENT

END-OPEN

IDENTIFIED

VERSION-XML

Annex A
(normative)

Language element lists

A.1 Implementor-defined element list

The following is a list of the language elements within this Technical Report that depend on implementor definition to complete the specification of the elements. Each element is defined as required, optional, or conditionally required. Furthermore, each element is defined as requiring (or not requiring) user documentation. These terms have the following meaning:

— Required: The element shall be provided by the implementor. When the element is part of a feature that is optional or processor-dependent, the item is not required if the optional or processor-dependent feature is not implemented.

— Optional: The element may be provided at the implementor's option.

— Conditionally required: If the associated feature or language element is implemented, this element is also required.

— Documentation required: If the element is provided by the implementor, the implementor's user documentation shall document the element or shall reference other documentation that fulfills this requirement.

A short header and informative optional parenthetical text provide a paraphrase of the normative detailed specification located in the body of this Technical Report and direct the reader to that detail. A cross-reference is provided for all items.

1) Computer's coded character set (substitution character used when a character in an XML document has no corresponding character in the runtime coded character set). This item is required. This item shall be documented in the implementor's user documentation. (14.6a.3, Codeset conversion for input data transfer)

2) Computer's coded character set (substitution character used when a character to be written to an XML document has no corresponding character in the coded character set used for the XML document). This item is required. This item shall be documented in the implementor's user documentation. (14.6a.4, Codeset correspondence for output data transfer)

3) Coded character set of an XML document (default used when the implementation cannot determine or verify the identity of the coded character set used). This item is required. This item shall be documented in the implementor's user documentation. (14.6a.4, Codeset correspondence for output data transfer)

4) DTD or schema (conditions that determine whether available). This item is required. This item shall be documented in the implementor's user documentation. (14.8.26.3, OPEN statement, general rule 27)

A.2 Undefined language element list

The following are language elements within this Technical Report that are explicitly undefined.

1) Open statement. The content of an associated internal representation is undefined after unsuccessful execution of an XML-document format OPEN statement. (14.8.26.1, OPEN statement, general rule 37)

2) Transfer of XML input data to floating-point numeric receiving items. The content of the receiving data item is undefined when XML-value is plus infinity (INF), negative infinity (-INF), or Not a Number (NaN) or when XML-value contains an exponent that has a value greater than the implementation supports. (14.6a.1.6, Floating-point numeric receiving items)

A.3 Processor-dependent language element list

A processor consists of the hardware and associated software that are used to translate a compilation group or to execute a run unit.

The following is a list of the COBOL language elements within this Technical Report that depend on specific devices or on a specific processor capability, functionality, or architecture. A processor-dependent element may relate to capability or functionality of hardware or of software, or both. An element described with a device-specific term may be implemented in hardware, software, or a combination of hardware and software.

1) The interaction of the features of this Technical Report with features that are not provided by an implementation of ISO/IEC 1989:2002 is dependent on the capabilities of the processor."

Annex B
(informative)

Unresolved technical issues

B.1 General

The following technical issues are presented here in order to obtain feedback from reviewers and early implementors.

1. In this Technical Report, the IDENTIFIED clause maps XML data to a COBOL data item. There is no way to determine the number of characters of XML data that are transferred into the data item. Is there a need to know this length and whether the data was truncated?

This restriction is temporary because when this TR is included in a future edition of ISO/IEC 1989, the ANY LENGTH clause can be specified on the data items that receive the XML data and then there will be no limit to the size and the size can be determined.

2. The XML 1.0 and 1.1 recommendations allow specification of the values plus or negative infinity (INF and -INF) and Not a Number (NaN) for floating point items.

When this technical report is incorporated into the standard, these 3 values will become valid values for data items with the IEEE floating point types.

Until the IEEE floating point types are available during XML processing, these values are not valid for any receiving data item. When these values are encountered in an XML document, should the receiving data item be unchanged, undefined, implementer-defined, or be a defined value, such as 0E+0? This Technical Report sets a fatal exception condition to exist and leaves the receiving data item unchanged, which is what a MOVE statement does when the sending operand sets the EC-DATA-INCOMPATIBLE exception condition to exist. It is currently intended that when this Technical Report is incorporated into the standard and the receiving item is an IEEE floating point type that supports these values, the value will be assigned to the receiving data item and a nonfatal exception condition will be set to exist.

3. The requirement in this TR to check whether the generated XML is well-formed during the execution of an XML-format WRITE statement and an XML-format REWRITE statement necessitates a runtime overhead. Before this TR is included in a future edition of ISO/IEC 1989, an option should be considered to postpone this checking until an XML-format CLOSE statement is executed.

Annex C
(informative)

XML processing concepts

This Annex presents an overview of XML and examples of working with XML using the COBOL syntax extensions specified by this Technical Report.

C.1 What is XML?

XML is a self-describing textual representation of data. Each element of data is preceded by a tag that gives the name of the element. This tag is surrounded by angle-brackets, for example <tag-name>. The element of data is followed by the same tag preceded by a slash (/), for example </tag-name>. Elements can be nested within elements to create entire structures of data.

There are several ways to describe the tags that are allowed in an XML document and the type of data and structure that can be associated with those tags. This Technical Report allows validation of XML documents described by either the XML Schema Description Language or a Document Type Definition (DTD).

Valid XML conforms to the schema or DTD that describes the document. An XML document that conforms to the rules for XML is considered well formed, even if there is no schema or DTD to make it a valid document.

The easiest way to gain an understanding of XML is through an example. Here is a small example from the sample application that is discussed later:

```
<customer xmlns="http://www.MyKingdom.com/PrinceCharming">
  <cust-name>Prince Charming</cust-name>
  <cust-age>55</cust-age>
  <cust-policy-value>500000</cust-policy-value>
  <cust-policy-cost>25000</cust-policy-cost>
</customer>
```

The root element in this example document is named *customer*. It references the XML namespace with the URI http://www.MyKingdom.com/PrinceCharming. (A Uniform Resource Identifier (URI) is a unique identifier for the namespace. It is not necessarily a web location.) The use of this URI does not imply that there is anything at that location that defines the namespace, instead the URI is just a unique identifier used in qualification of element and attribute names. Nested within the customer element is the sequence of elements *cust-name*, *cust-age*, *cust-policy-value*, and *cust-policy-cost*. Each element is terminated by a tag that matches the beginning tag and begins with a backslash. Between the two tags is the value, which is XML character data.

Data for an XML element can be specified as an attribute. For example:

```
<person sex="female">
</person>
```

Here sex is an attribute of person. However, the same information can also be conveyed as data associated with an element as follows:

```
<person>
        <sex> female </sex>
</person>
```

You can relate COBOL data structures with XML element definitions using the COBOL syntax in the file section.