J4/05-0049

WG4n0229

ISO/IEC JTC 1/SC 22

Date: 2005-02-16

ISO/IEC TR 24716:200x(E)

ISO/IEC JTC 1/SC 22/WG 4

Secretariat: ANSI

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

Document type: preliminary draft Technical Report
Document subtype: 2
Document stage: (2) WD under consideration
Document language: E

Contents

Foreword ... iv
Introduction ... v
1 Scope ... 1
2 Normative references ... 1
3 Conformance to this Technical Report ... 1
4 Terms and definitions ... 1
5 Description techniques ... 2
6 Changes to ISO/IEC 1989:2002 ... 2
6.1 Changes to 8, Language fundamentals ... 2
6.2 Changes to 9, I-O, objects, and user-defined functions ... 3
6.3 Changes to 12, Environment division ... 5
6.4 Changes to 13, Data division ... 9
6.5 Changes to 14, Procedure Division ... 14
Annex A (normative) **Language element lists** ... 29
Annex B (informative) **Unresolved technical issues** ... 30
Annex C (informative) **Concepts** ... 31
Bibliography ... 40

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.

Technical Reports are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Technical Reports adopted by the joint technical committee are circulated to national bodies for ballot. Publication as a Technical Report requires approval by at least 75 % of the member bodies that cast a vote.

Attention is drawn to the possibility that some of the elements of this Technical Report may be the subject of patent rights. Neither ISO nor IEC shall be held responsible for identifying any or all such patent rights.

Technical Report ISO/IEC 24716 was prepared by Technical Committee ISO/IEC JTC 1, *Information Technology*, Subcommittee SC 22, *Programming languages, their environments and system software interfaces*. INCITS Technical Committee J4, Programming language COBOL, contributed to the development.

Technical Report ISO/IEC 24716 extends the COBOL specification defined in ISO/IEC 1989:2002, Information technology — Programming languages, their environments and system software interfaces — Programming language COBOL. It provides new syntax to handle XML documents and schemas in Programming language COBOL.

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

Introduction

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

a. Is based on the familiar approach used with COBOL I/O support
b. Provides Document Object Model (DOM) style parsing
c. Handles multiple input sources to handle XML in an extremely flexible manner
d. Reads, updates, and writes XML documents
e. Checks that XML documents are well-formed
f. Provides an optional validity check of an XML document against a schema or Document Type Definition (DTD).

Information Technology —

Programming languages, their environments 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 normative documents contain provisions that, through reference in this text, constitute provisions of this preliminary draft Technical Report ISO/IEC 24716. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this preliminary draft Technical Report ISO/IEC 24716 are encouraged to apply the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

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

Extensible Markup Language (XML) 1.1, W3C Recommendation 04 February 2004

Namespaces in XML, W3C Recommendation, 14 January 1999

XML Schema Part 1: Structures, W3C Recommendation, 2 May 2001

XML Schema Part 2: Datatypes, W3C Recommendation, 2 May 2001

3 Conformance to this Technical Report

Conformance to this preliminary draft Technical Report requires conformance to ISO/IEC 1989:2002 as specified in clause 3, Conformance to this International Standard, and the normative specifications of this preliminary draft Technical Report.

4 Terms and definitions

For the purposes of this preliminary draft Technical Report, the following definitions apply:

4.1 **DTD; document type definition:** See document type definition.

4.2 **document type definition; DTD:** XML that defines the structure of a document with a list of valid elements.

4.3 **XML document**: A unit of data that is well-formed, as defined in Extensible Markup Language (XML) 1.0 Third Edition, W3C Recommendation 04 February 2004

4.4 **XML element:** A 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. Each element has a type, identified by name, and can have a set of attribute specifications.

4.5 **XML file:** A file or memory area that contains XML documents.

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

IDENTIFIED
NAMESPACE

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

ATTRIBUTE	IDENTIFIED clause
CHECK	TYPE clause of the file control entry
DOCUMENT	CLOSE statement and OPEN statement
DTD	TYPE clause of the file control entry
ELEMENT	DELETE statement, READ statement, REWRITE statement, START statement, and WRITE statement
SCHEMA	TYPE clause of the file control entry
VALIDITY	TYPE clause of the file control entry
XML	ACCESS clause, ORGANIZATION clause, and SAME 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.

[b] 9.1.6, Fixed file attributes, third sentence, add "XML" to the end of the list of file organizations.

[c] 9.1.7, Organization,

1) Change "three" to "four" and add "XML" to the end of the list of file organizations.

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. 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 can contain other elements. Within the context of its superordinate elements each subordinate element can be stored, retrieved, or deleted based on its tag name. When there are multiple occurrences of a TAG name in the document, the containing group must be made visible before its subordinate elements are visible. The element position indicator for XML files contains the position within each subordinate group.

COBOL allows multiple documents per file. The file position indicator maintains the document number of the current document in the XML file.

The file format OPEN statement for an XML file establishes a connection to the physical file. An XML-document format OPEN statement deletes the previous internal representation, if any, creates an internal representation of a document, and moves data into associated data items described in the file section. The READ statement and the START statement traverse 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 writes an XML document based on the current internal representation. The file remains in the open state. A file format CLOSE statement closes the physical file.

The implementor defines where white space characters are inserted.

All XML documents written by COBOL are created in compliance with *Extensible Markup Language (XML) 1.1*, W3C Recommendation 04 February 2004.

[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 XML organization may be accessed only in XML mode."

3. Add the following new section:

"9.1.8.4 XML access mode

When a file is accessed in XML 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 can be used to set the element position indicator to a specific occurrence of an element. However, the position can only be set in a top-down manner within the hierarchy of the elements in the document."

[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 indicator

The element position indicator is a conceptual entity that exists for each XML file to maintain the position within each group in a hierarchy. The element position indicator is set by OPEN and START statements and updated by READ statements. Each position is associated with a data item in the COBOL record description that is described with an IDENTIFIED clause. Elements can be accessed by their tag names as long as all containing elements are indicated by the element position indicator. When there are multiple elements with the same tag name in the document, the position may be set to a specific occurrence with a START statement or the occurrences can be read sequentially."

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

7) I-O status = 08. The input-output statement is successfully executed but an OPEN or READ statement ignored one or more XML tags because there is no COBOL data item with which the tags can be associated.

8) I-O status = 09. The input-output statement is successfully executed but the referenced DTD or schema is not available so 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, 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

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

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

5) I-O status = 25. A DELETE statement attempted to delete an XML element at an invalid position."

[j] 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"

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

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

9) I-O status = 3B. An XML-document format CLOSE statement attempted to write an XML document that is not well-formed.

10) I-O status = 3C. An OPEN statement attempted to open a multiple document file in the I-O mode."

[l] 9.1.12.6, Logic error condition with unsuccessful completion, item 7, I-O Status = 47, add OPEN to the list of statements.

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

"10) I-O status = 4A. The execution of a REWRITE statement or a WRITE statement attempted to create XML that is not well-formed.

11) I-O status = 4B. The execution of an XML-document format CLOSE statement attempted to reference a file connector that is not open in the extend, I-O, or output mode.

14 I-O status = 4C. The execution of an OPEN statement or a READ statement is unsuccessful because duplicate names are specified in IDENTIFIED clauses at the same level in the hierarchy."

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

         ⎧     ⎧ ⎧ device-name-1 ⎫                             ⎫ ⎫
         ⎪     ⎪ ⎩ literal-1     ⎭  . . . [ USING data-name-1 ] ⎪ ⎪
         ⎪  TO ⎨                                                ⎬ ⎪
    ASSIGN ⎨   ⎪ 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

    ⎡          ⎧ DTD                                              ⎫ ⎡                         ⎧| INPUT  |⎫ ⎤⎤
    ⎢ TYPE IS  ⎨                                                  ⎬ ⎢ CHECK VALIDITY ON       ⎨          ⎬ ⎥⎥ .
    ⎣          ⎩ EXTERNAL ⎧ DTD    ⎫ ⎧ literal-3     ⎫            ⎭ ⎣                         ⎩| OUTPUT |⎭ ⎦⎦
                          ⎩ SCHEMA ⎭ ⎩ data-name-12  ⎭
```

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.

16) Data-name-10 shall reference a data item of category data-pointer.

17) Data-name-11 shall reference an unsigned integer data item described without the picture symbol 'P'.

18) 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 subrule:

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

[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) When data-name-9 is specified, the XML data is contained in the data item referenced by data-name-9.

d) When data-name-10 is specified, the XML data begins at the address specified by the content of the data item referenced by data-name-10 and continues for the length specified by the content of the data item referenced by data-name-11.

[e] Add the following new file control entry general rules 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,

a. 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."

b. add the following new general rule:

"5) If the access mode is XML, documents within the file may be accessed in sequential order. Elements in the current document may be accessed via their tag names as long as all containing elements are indicated by the element position indicator."

[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 12.3.4.9.3, General rules [ORGANIZATION clause], add the following:

"

6) 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] Add the following after section 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

$$\left[\underline{\text{TYPE}}\ \text{IS}\left\{\begin{array}{l}\underline{\text{DTD}}\\ \underline{\text{EXTERNAL}}\left\{\begin{array}{l}\underline{\text{DTD}}\\ \underline{\text{SCHEMA}}\end{array}\right\}\left\{\begin{array}{l}\text{literal-1}\\ \text{data-name-1}\end{array}\right\}\end{array}\right\}\left[\underline{\text{CHECK}}\ \text{VALIDITY ON}\left\{\begin{array}{l}|\ \underline{\text{INPUT}}\ |\\ |\ \underline{\text{OUTPUT}}\ |\end{array}\right\}\right]\right]$$

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 shall reference an alphanumeric or national data item.

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.

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

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

1. In the initial paragraph, add "XML processing" to the list.
2. Add a new general format as follows:

   "Format 4 (XML-area):

   $\underline{\text{SAME}}\ \ \underline{\text{XML}}\ \ \text{AREA FOR file-name-1}\ \{\text{file-name-2}\}\ldots$
3. Make the following changes to the syntax rules in the standard:
   a) In syntax rule 5, add XML-area format to the list of prohibitions.

   b) In syntax rule 6, add XML-area format to the prohibitions.

   c) In syntax rule 7, insert the following text after "a report file":

   ", an XML file,"
4. Add the following new syntax rules:

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

   11) If the XML phrase is specified, file-name-1 and file-name-2 shall have XML organization.

   12) If the XML phrase is specified and any of the file control entries for file-name-1 and file-name-2 include a TYPE clause, all file control entries for file-name-1 and file-name-2

shall include identical TYPE clauses, with the exception that the CHECK phrases may vary.

5. Add the following new general rule:

   "5) An XML-area format SAME clause specifies that two or more files referenced by file-name-1 and file-name-2 are to share the internal representation for processing the associated XML document. All of those files may be in the open mode at the same time, except that only one file that is also specified in a file-area format SAME clause may be open at that time. The internal representation in the shared XML area is available to each file open in the output mode and to the most recently-opened file in the input mode. The internal representation is available to the runtime element when any file connector referenced in the XML-area format SAME clause is open. When none of the file connectors is open, the internal representation is not available to the runtime element."

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)

$$\begin{array}{l}\underline{\text{FD}}\ \text{file-name-1}\\ \qquad [\ \text{IS}\ \underline{\text{EXTERNAL}}\ [\ \underline{\text{AS}}\ \text{literal-1}\ ]\ ]\\ \qquad [\ \text{IS}\ \underline{\text{GLOBAL}}\ ]\\ \qquad \left[\ \underline{\text{CODE-SET}}\ \text{IS} \left\{\begin{array}{l}\text{alphabet-name-3}\\ \text{data-name-6}\end{array}\right\}\right]\ .\ "\end{array}$$

[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."

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

$$\text{"}\underline{\text{COUNT}}\ \text{IN data-name-7"}$$

$$\text{"}\underline{\text{IDENTIFIED}} \left[\begin{array}{l}\underline{\text{BY}}\left\{\begin{array}{l}\text{data-name-8}\\ \text{literal-1}\end{array}\right\}\\ \underline{\text{USING}}\ \text{data-name-9}\end{array}\right]\ [\ \text{IS}\ \underline{\text{ATTRIBUTE}}\ ]\ \text{"}$$

$$\text{"}\underline{\text{NAMESPACE}} \left\{\begin{array}{l}\underline{\text{IS}}\left[\begin{array}{l}\text{data-name-10}\\ \text{literal-1}\end{array}\right]\\ \underline{\text{USING}}\ \ \text{data-name-11}\end{array}\right\}\ \text{"}$$

[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):**

   $$\left[\ \underline{\text{CODE-SET}}\ \text{IS}\left\{\begin{array}{l}\text{alphabet-name-3}\\ \text{data-name-1}\end{array}\right\}\right]\ .\ \text{"}$$

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 either an alphanumeric or a national coded character set.

   5) 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) When the file is opened in the OUTPUT mode, the encoding referenced by the specified code-set is written as the encoding attribute in the XML declaration.

   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.

   9) When the file is opened in the INPUT or I-O mode, the encoding attribute in the XML declaration in the document specifies the encoding of the document.

   If alphabet-name-3 is specified and the referenced encoding does not match the encoding attribute specified in the document, the EC-XML-CODESET exception is set to exist.

   If data-name-1 is specified, the encoding attribute specified in the document is moved into the data item referenced by data-name-1 during the processing of the OPEN statement.

   10) The specified alphabet 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, the ASCII code-set is used by default."

[e] Add the following new section 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 a tag or attribute 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.

Format

COUNT IN data-name-1

Syntax Rule

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.

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.

2) When an XML-element format REWRITE statement or an XML-element format WRITE statement is executed for the file, content of the data item referenced by data-name-1specifies the number of occurrences of the subject of the entry to write.

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 will contain 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] Add the following new section 13.16.28a, IDENTIFIED clause:

"13.16.28a IDENTIFIED clause

The IDENTIFIED clause maps an XML data name (tag) to a COBOL data name.

General Format

$$\underline{\text{IDENTIFIED}}\left[\begin{array}{l}\underline{\text{BY}}\left\{\begin{array}{l}\text{data-name-1}\\ \text{literal-1}\end{array}\right\}\\ \underline{\text{USING}}\ \ \text{data-name-2}\end{array}\right]\ [\ \text{IS}\ \underline{\text{ATTRIBUTE}}\ ]$$

Syntax Rules

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

2) Literal-1 shall be of class alphanumeric or national.

3) Data-name-1 and data-name-2 shall reference an alphanumeric or national data item.

4) If IDENTIFIED is specified but neither the BY phrase nor the USING phrase is specified, it is as if literal-1 were specified and contained the data-name of the subject of the entry.

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

6) The data-name-2 phrase shall be specified in at most one level 1 data description entry subordinate to a given file description entry.

7) If data-name-1 or data-name-2 is subordinate to a file with organization XML, it must be directly subordinate to the subject of the entry.

8) At any given level in a record description, all literal-1 shall be unique.

General Rule

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.

2) On input, if literal-1 is specified, the subject of the entry is associated only with an XML element or attribute 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 element or attribute 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 tag or attribute, whichever is specified in the IDENTIFIED clause, and the XML tag name or attribute 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 listed in the COBOL record description.

5) On output, the content of data-name-1 or data-name-2 shall be a valid 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 tag name or attribute 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, rather than an element.

8) At any given level within a record description, the data-name-2 phrase is only effective if there are no other data description entries at that level for which literal-1 or data-name-1 matches the tag name. If literal-1 or data-name-1 matches the tag name, that data description is used instead.

9) When the subject of the entry is referenced in an OPEN or READ statement, the value of literal-1 or the contents of the data item referenced by data-name-1 shall be unique from the values referenced in IDENTIFIED clauses in all the other data items defined at the same level in the hierarchy as the subject of the entry; otherwise, the I-O status value in the associated file connector is set to '4C' and the execution of the statement is unsuccessful."

[g] Add the following new section 13.16.34a, NAMESPACE clause:

"13.16.34a NAMESPACE clause

The NAMESPACE clause sets or identifies the namespaces that are contained in an XML document.

13.16.34a.1 General Format

$$\underline{\text{NAMESPACE}} \left\{ \begin{array}{l} \underline{\text{IS}} \left[\begin{array}{l} \text{data-name-1} \\ \text{literal-1} \end{array} \right] \\ \underline{\text{USING}}\ \text{data-name-2} \end{array} \right\}$$

13.16.34a.2 Syntax rules

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

2) Data-name-1 and data-name-2 shall reference an alphanumeric or national data item.

3) Literal-1 shall be alphanumeric or national.

13.16.34a.3 General rules

1) When the file is open in the output mode, execution of an XML-document format CLOSE statement writes the contents of the data item referenced by data-name-1 or data-name-2 or the value of literal-1 as the namespace attribute in the XML.

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

   If literal-1 or data-name-1 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-2 is specified, the namespace attribute specified in the document is moved into the data item referenced by data-name-2."

[h] In 13.16.42.2, syntax rules for the REDEFINES clause, syntax rule 3, add to the end "or has organization XML".

[i] 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 with organization XML."

[j] 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 clause 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-XML		XML error
EC-XML-CODESET	NF	The encoding attribute in the XML declaration in the XML document does not match the encoding specified in the CODE-SET clause
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-INVALID	NF	The XML document does not conform to the DTD or schema
EC-XML-NAMESPACE	NF	The namespace attribute in the XML declaration in the XML document does not match the encoding specified in the NAMESPACE clause
EC-XML-UPDATES	NF	The updates made to the internal representation of an XML document are about to be discarded

[d] CLOSE statement, 14.8.6:

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

"The CLOSE statement completes the processing of an XML document.

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

   "Format 2 (XML-document)

   <u>CLOSE</u> <u>DOCUMENT</u> file-name-1"

3. Add the heading "Format 1 (files): 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 XML files.

   FORMAT 2

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

4. General rules 1 and 4 apply to both formats. The other 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 changes have been made to the internal representation of the XML document that have not been written to the file referenced by file-name-1, the EC-XML-UPDATES exception condition is set to exist. If this exception condition results in the execution of a declarative procedure that executes a RESUME statement with the NEXT STATEMENT phrase, processing resumes with continuation of the execution of this close operation.

6. Add the following general rules for format 2:

12) The file connector referenced by file-name-1 shall have an open mode of extend, I-O, or output. If the open mode is some other value or the file is not open, the I-O status in the file connector referenced by file-name-1 is set to '4B' and the execution of the CLOSE statement is unsuccessful.

13) If the OUTPUT phrase is specified in the TYPE clause of the file control entry for fle-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 in the file connector referenced by file-name-1 is set to '3B' and the execution of the CLOSE statement is unsuccessful.

14) The successful execution of a CLOSE statement creates an XML document from the internal representation and releases that document to the operating environment. If the TYPE clause in the file control entry for file-name-1 has the DTD phrase without an EXTERNAL phrase, a DTD that describes that internal representation is included in the file. .

    NOTE 1 The content of the XML document may be impacted by the CODE-SET and NAMESPACE clauses in the file description.

    NOTE 2 Updates to data items subordinate to a file description entry that are not written with XML-element format WRITE or XML-element format REWRITE statements are discarded.

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

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

17) If the execution of the CLOSE statement is unsuccessful, no update of the file takes place and the content of the internal representation is unaffected.

18) 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

    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.

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

The DELETE statement deletes an XML element and all sub-elements from the in-memory representation of an XML document.

[f] In the general format of the DELETE statement, 14.8.9.1, 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):

```
DELETE file-name-1 RECORD

    ELEMENT IS data-name-1

    ⎡| INVALID KEY imperative-statement-1         |⎤
    ⎣| NOT INVALID KEY imperative-statement-2     |⎦

    [ END-DELETE ]
```

[g] 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 a record description of file-name-1. "

[h] 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 open mode of the file connector referenced by file-name-1 shall be I-O.

13) After the successful execution of a DELETE statement, the element identified by data-name-1 and all its child elements are removed from the internal representation of the document and can no longer be accessed. All other elements represented in the document remain unchanged.

14) The value of the element position indicator at the start of the execution of the DELETE statement is used to determine what element is to be deleted in accordance the following rules:

   a. If the element position indicator indicates that no valid position has been established, execution of the DELETE statement is unsuccessful.

   b. If the element position indicator 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 in the internal representation that is indicated by the element position indicator and that is within the occurrences of all current directly and indirectly containing groups described with an IDENTIFIED clause that are indicated by the element position indicator.

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

   If no element or attribute is found that satisfies this general rule, the I-O status value associated with file-name-1 is set to '25' and the invalid key condition exists.

[h1] 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 …"

[h2] Append the following sentence to the introductory paragraph of the OPEN statement:

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

[h3] Label the existing format of the OPEN statement as "Format 1 (file):" and add the following new format:

"FORMAT 2 (XML-document):

```
OPEN  NEXT  DOCUMENT  file-name-1

        ┌| AT END imperative-statement-1      |┐
        └| NOT AT END imperative-statement-2  |┘

       [ END-OPEN ]
```

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

1. 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."

2. Add the following syntax rules for format 2 only:

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

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

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

"memory location or "

2. 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	
	CLOSE DOCUMENT		X	X	X

"

3. 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."

4. 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."

5. Add the following new general rules to the OPEN statement:

"26) If the INPUT, I-O, or EXTEND phrase is specified on the OPEN statement and data-name-12 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 declaration in the document into the data item referenced by data-name-12. If literal-3 is specified in the

TYPE clause for file-name-1, execution of the OPEN statement checks that literal-3 matches the schema or DTD specified in the XML declaration in the 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 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.

29) If the INPUT or I-O phrase is specified and the file referenced by file-name-1 has organization XML, the actions for the file format OPEN statement are executed and then the actions for an XML-document format OPEN statement are executed.

FORMAT 2

30) The open mode of the file connector referenced by file-name-1 shall be INPUT or I-O. If it is any other value, the execution of the OPEN statement is unsuccessful and the I-O status value for file-name-1 is set to '47'.

31) The OPEN statement creates an internal representation of the next XML document in file-name-1.

The element position indicator is set such that a subsequent XML-element format READ statement that references any data item in the hierarchy reads the first XML element whose tag matches that specified by the associated IDENTIFIED clause.

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.

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.

32) The value of the file position indicator at the start of the execution of the OPEN statement is used to determine what is to be 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 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 36.

33) If changes have been made to the internal representation created by the previous XML-document format OPEN statement that have not been written to the file referenced by file-name-1, the EC-XML-UPDATES exception condition is set to exist. 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.

34) Any tags in the XML document with no associated data items in the COBOL record are ignored. If a tag is ignored, the data associated with that tag is ignored. If tags are ignored and there is no error that causes the execution of the OPEN statement to be unsuccessful, the execution of the OPEN statement is successful and the I-O status value associated with file-name-1 is set to '08'.

35) 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 filename-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.

36) 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.

37) 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.

38) 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 indicator 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."

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

For XML access, the READ statement makes available a specified XML element and populates the associated COBOL record area.

[l] In the general format of the READ statement, 14.8.29.1, General format:

2. Add the following after the current general format:

"FORMAT 3 (XML-element):

```
READ file-name-1

    ELEMENT data-name-2

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

    [ END-READ ] "
```

[m] 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 a record description of file-name-1."

[n] 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 this new format.

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 moves the next occurrence(s) of that associated element to the specified data item and all subordinate items. No file I/O is performed. It moves as much data as possible into the associated data item, as dictated by the elements in the document and 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, a value that indicates the at end condition is placed into the I-O status associated with file-name-1.

Note When there are more elements with a given tag name in the document than fit into the COBOL record, successive reads that specify the associated COBOL data-name can be done until the at end condition exists. This can be repeated at each level where multiple elements with the same name are possible.

31) The value of the element position indicator at the start of the execution of the READ statement is used to determine what is to be made available in accordance with the following rules:

c) If the element position indicator indicates that no valid position has been established, execution of the READ statement is unsuccessful.

d) If the element position indicator was established by a prior OPEN, READ or START statement, the element that is selected is the first existing element associated with the data item referenced by data-name-2 in the internal representation that is after the location indicated by the element position indicator and that is within the occurrences of all current directly or indirectly containing groups described with an IDENTIFIED clause that are indicated by the element position indicator.

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

If no element or attribute is found that satisfies this general rule, the element position indicator 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 read is the maximum number allowed by the OCCURS clause or the number of unread XML elements, whichever is less.

33) Any tags in the XML document with no associated data items in the COBOL record are ignored. If a tag is ignored, the data associated with that tag is ignored. If tags 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 an item has an IDENTIFIED clause, any directly subordinate data items without an IDENTIFIED clause in their description that are not referenced by an IDENTIFIED or NAMESPACE clause on the group item are associated with the identified XML element.

If there is only one associated subordinate item and that subordinate item is numeric, the data is transferred as though the following statement were executed:

```
    COMPUTE receiving-field = FUNCTION NUMVAL-C (XML-data, ANYCASE)
```

For other single associated subordinate items, the data is transferred with a MOVE statement. If the execution of any of these COMPUTE or MOVE statements causes the EC-DATA-INCOMPATIBLE exception condition to exist and that results in the execution of a RESUME statement with the NEXT STATEMENT phrase, processing resumes with the next COMPUTE or MOVE statement.

If there are multiple associated subordinate items, these subordinate items are receiving items that are filled as though they were receiving items in an UNSTRING statement without any optional clauses.

If there are no associated subordinate items, the data associated with the tag referenced by the IDENTIFIED clause is ignored."

[o] Add the following new paragraphs to 14.8.33, REWRITE statement:

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

[p] 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

ELEMENT IS [ALL] data-name-1

```
⎡| INVALID KEY imperative-statement-1         |⎤
⎣| NOT INVALID KEY imperative-statement-2 |⎦

[ END-REWRITE ] "
```

[q] 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 subordinate to a record description of the rewrite file. "

[r] 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 this new format.
2. Add the following after the current general rules:

"FORMAT 2:

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

If ALL is specified, the contained elements and attributes are also individually updated in the internal representation of the XML document from their associated data items in the COBOL record; otherwise, only the specific element and its attributes and data are updated.

Any data items directly subordinate to data-name-1 that have no IDENTIFIED clause in their description and that are not referenced by an IDENTIFIED or NAMESPACE clause in the description of data-name-1 are associated with the XML element or attribute. These subordinate item or items are sending items and act as though they were sending items delimited by size in a STRING statement. If the execution of this STRING statement causes the EC-DATA-INCOMPATIBLE exception condition to exist and that results in the execution of a RESUME statement with the NEXT STATEMENT phrase, processing resumes with the next STRING statement.

NOTE There are multiple STRING statements executed when there is an ALL phrase on the REWRITE statement.

When the ALL phrase is 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 paragraph.

If the resultant XML is not well-formed, the I-O status value of the rewrite file connector is set to '4A' and the execution of the REWRITE statement is unsuccessful.

26) If the preceding I-O statement executed for the rewrite file was a START statement, the invalid key condition exists, the execution of the REWRITE statement is unsuccessful, and the I-O status in the rewrite file connector is set to the invalid key condition '23'.

27) 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 element with the new tag-name replaces the previous XML element."

[r1] 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."

[s] 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

                          ┌ INDEX IS { data-name-3 }                                                  ┐
ELEMENT IS data-name-2    │          { 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 ]"
```

[t] 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 a record 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."

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

2. Add the following after the current general rules:

"XML FILES

22) The element position indicator is set to indicate an element with the tag name specified in the IDENTIFIED clause of the description of data-name-2. The element indicated by the element position indicator becomes the element of reference. ELEMENT phrases in subsequent statements that reference a data item subordinate to the element of reference do so within the element of reference.

NOTE The element position indicator therefore maintains position within all levels of the hierarchy specified with IDENTIFIED clauses in the description of the document.

If the specified occurrence of the element does not exist, the I-O status value in the file connector referenced by file-name-1 is set to '23', the invalid key condition exists, the element position indicator 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, the element position indicator is set to indicate the first occurrence of an element with the tag name specified in the IDENTIFIED clause of the description of data-name-2 that is within the current element of reference.

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 indicator 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 value in the file connector referenced by 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 element whose tag name 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. This comparison is made according to the collating sequence of the XML document. 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 indicator is set to indicate the first element that satisfies the comparison, or

2. If the comparison is not satisfied by any element, the invalid key condition exists and the execution of the START statement is unsuccessful.

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

27) 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 To insert an element after the element specified in the START statement, instead of before it, a READ statement that specifyies the same element as specified on the START must first be executed.

[v] 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.

[w] 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
        ELEMENT IS [ ALL ] data-name-1
       ┌| INVALID KEY imperative-statement-1      |┐
       └| NOT INVALID KEY imperative-statement-2  |┘
[ END-WRITE ]"
```

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

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

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

[y] 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 document and does not modify the file. The element or attribute specified by data-name-1 is inserted immediately after the position indicated by the element position indicator. If any containing elements are required for the element specified, these are also inserted into the internal representation. If ALL is specified, the contained elements and attributes are also individually added to the internal representation of the XML document from their associated data items in the COBOL record; otherwise, only the specific element and its attributes and data are added.

Any data items directly subordinate to data-name-1 that have no IDENTIFIED clause in their description and that are not referenced by an IDENTIFIED or NAMESPACE clause in the description of data-name-1 are associated with the XML element or attribute. These subordinate item or items are sending items and act as though they were sending items delimited by size in a STRING statement. If the execution of this STRING statement causes the EC-DATA-INCOMPATIBLE exception condition to exist and that results in the execution of a RESUME statement with the NEXT STATEMENT phrase, processing resumes with the next STRING statement.

NOTE There are multiple STRING statements executed when there is an ALL phrase on the WRITE statement.

When the ALL phrase is 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 paragraph.

If the resultant XML is not well-formed, the I-O status value of the write file connector is set to '4A' and the execution of the WRITE statement is unsuccessful."

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.

1) White space (where white space is inserted). This item is optional. This item, if provided by the implementor, does not have to be documented in the implementor's user documentation. (9.1.7.4, XML)

2) 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) The content of the associated internal representation is undefined after unsuccessful execution of an XML-document format OPEN statement. (14.8.26.1, OPEN statement, general rule 38)

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.

NOTE Resolved issues are documented in J4/05-0028, Issues for the XML TR - Resolved and Open.

1. Early direction was to use the keyword INITIATE for opening an XML document and the keyword TERMINATE for closing an XML document. Users objected to the use of INITIATE and TERMINATE. Particularly troublesome was the use of TERMINATE, because it carries the connotation of killing the run unit for those who are not familiar with the Report Writer facility. The following syntax is used in this draft:

- OPEN NEXT DOCUMENT (instead of INITIATE)
- CLOSE DOCUMENT (instead of TERMINATE)

2. In this preliminary draft Technical Report, the IDENTIFIED clause maps an XML data structure to a COBOL record structure. XML can contain data of two different types: either all characters except whitespace characters are returned (space, tab, etc.) OR all characters including whitespace are returned (the XML datatypes are #CDATA and #PCDATA). In neither case is there a way to determine the character count of the XML data. Is there a need to know this length and whether the data was truncated?

Annex C
(informative)

Concepts

C XML processing concepts

This section presents an overview of XML and examples of working with XML using the COBOL syntax extensions specified by this preliminary draft 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 preliminary draft Technical Report allows validation of XML documents described by either the XML Schema Description Language or a Document Type Definition (DTD).

Valid XML must conform 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:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <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 namespace with the URI http://www.w3.org/2001/XMLSchema-instance. (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.

C.2 Describing XML files

Use the file control entry to describe files that contains XML. The SELECT clause specifies the use of XML; gives the location of the XML data and optionally of the schema or DTD that describes the XML data; and optionally identifies the data item to receive the file status for XML operations. For example:

```
select library-file
        assign to data mybuf
        organization is xml
        type is external schema "library.xsd"
        file status is library-status.
```

In this example, the file that contains the XML is named library-file. The data item mybuf contains the XML data at runtime. ORGANIZATION IS XML identifies the file as an XML file and is required. The schema that describes the data is found at library.xsd. The data item library-status contains information about the success or failure of all operations on library-file.

The File Description (FD) entry in the file section describes the "record layout" of the XML data with new descriptive clauses, specifies the mapping between a COBOL data-name and an XML tag name, and identifies namespaces. For example:

```
FD myxml.
01  myxml-rec    IDENTIFIED BY "myxml_rec".
    05  myxml-data  pic x(40).
    05  myxml-data-nest IDENTIFIED USING myxml-data-nest-tag
          IS ATTRIBUTE
          COUNT IN myxml-data-nest-count.
        10  myxml-data-nest-tag     pic x(40).
        10  myxml-data-nest-value   pic x(40).
```

In this example, the IDENTIFIED clause at the 01 level indicates that the COBOL record myxml-rec corresponds to the XML document with root tag "myxml_rec". The COBOL data item myxml-data-nest represents an attribute of "myxml_rec" and, at runtime, the XML name of that attribute will be stored in the data item myxml-data-nest-tag since IDENTIFIED USING was specified. Since the attribute may or may not exist in the XML document read, the count data item myxml-data-nest-count is used to indicate its presence or absence. At runtime, the data associated with the root tag is contained in myxml-data and the value of the attribute named by myxml-data-nest-tag will appear in myxml-data-nest-value.

The NAMESPACE clause can be included in the record description to identify the XML namespace to which the XML-name belongs. This is analogous to a qualified name in COBOL. A namespace applies to all items subordinate to the one where it is specified. For example:

```
01  root-tag            identified by "root"
               namespace is "yourname".
    10  root-tag-val         pic x(80).
```

In this example, the NAMESPACE clause specifies that the name root belongs to the namespace "yourname".

Consider a slightly more complicated example:

```
01  root-tag            identified using root-tag-name.
    10  root-tag-name   pic x(80).
    10  root-tag-val    pic x(80).
    10  root-tag-attr   identified using root-tag-attr-name
                           is attribute
                           namespace using yourname.
     15  yourname pic x(80).
     15  root-tag-attr-name pic x(80).
             15  root-tag-attr-val  pic x(80).
```

Here both the tag-names and the namespace are variable and derived from the XML document processed.

Consider an example to illustrate the COUNT clause:

```
01  root-tag identified by "library".
    10 root-data pic x(80).
    10  book identified by "book" occurs 5 count in kount.
       15  book-data pic x(80).
```

If there are 7 occurrences of the tag "book" in the document with the root tag "library", after the file is initially read, kount will contain the value 5, to indicate that 5 occurrences were read into the table book. Subsequently, after the following statement:

```
READ file-name ELEMENT book
```

is executed, kount will contain the value 2, to indicate that 2 additional occurrences were read.

C.3 Accessing XML files in the procedure division

The OPEN, CLOSE, READ, WRITE, DELETE, START, and REWRITE statements can be used to access XML.

An XML-document format OPEN statement creates an internal representation of that document. In addition, it populates the appropriate record definition with as much data as possible, based on the document elements and occurrence limitations of the associated XML tags. The populated elements are associated with their in-memory counterparts to allow DELETE, REWRITE, and WRITE statements to modify the internal representation.

The START statement sets the position that will be returned by a READ statement for the specified element or attribute.

A READ statement is used to access the next element or attribute of this internal representation with the specified name.

The DELETE statement removes an element of an XML document from its in-memory representation.

The REWRITE statement changes the in-memory representation of an element of an XML document.

The WRITE statement adds elements to the internal representation at the location established by previous I/O statements.

The XML-document format CLOSE statement outputs the internal representation to the file.

The file format OPEN and CLOSE statements operate as they do for other types of files.

C.4 An example of transforming an existing application to use XML

In this example, an application is modified to use XML for communicating information, instead of passing the information as parameters. This is a powerful technique because it facilities processing on differing platforms which communicate in a loosely-coupled manner.

C.4.1 Overview of existing application

This sample application gets information about a prospective customer and calculates a life insurance premium based on age and policy value. The application consists of one program that communicates with the customer and another that calculates the premium. The user-interface program calls the calculation program and passes parameters.

Your application is:

```
identification division.
program-id.  get-quote.
*> basic flow of this application - in original environment
*> - (prospective) customer enters data for price quotation
*> - calls another program to calculate price of policy
*> - price returned to customer
data division.
working-storage section.
01 screen-status pic x(50).
copy "mycust.cpy".
screen section.
copy "scrn-desc.cpy".
procedure division .
lets-begin.
    perform get-ready
    perform format-request
    perform send-request
    perform display-result
    goback.
get-ready.
   initialize customer
   move "Enter the following information:"  to screen-status
   display user-request
   accept user-request.
format-request. exit.
send-request.
    call "calc-quote" using customer.
display-result.
    move "The premium for this policy is shown below:"
                 to screen-status
    display user-request.
```

A simple program that performs this totally fictitious method of payment calculation could be:

```
identification division.
program-id.  calc-quote.
data division.
working-storage section.
01 nr-years-2-pay pic s999.
linkage section.
copy "mycust.cpy".
procedure division using customer.
start-here.
  compute nr-years-2-pay = 75 - cust-age
  compute cust-policy-cost = cust-policy-value / nr-years-2-pay
  goback.
```

As you can see, the 01 level customer data record is passed as an argument to communicate information between these two programs. It was put in the library text mycust.cpy to insure that the description matched between the two programs. It looks as follows:

```
01 customer.
    03 cust-name pic x(30).
    03 cust-age pic 9(3).
    03 cust-policy-value pic 9(12).
    03 cust-policy-cost pic 9(12).
```

The record, except for cust-policy-cost, is filled in from the information that the user enters during the ACCEPT statement. The cost is calculated by the called program.

C.4.2 Migrating to XML

A possbile first step in the migration to XML is to create a schema. This example puts the schema at http://www.schemalocation.com/my_schema, where our example program will reference it. The schema for this example is

```
<?xml version="1.0" encoding="utf-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
   <element name="customer">
      <complexType>
         <sequence>
            <element name="cust-name" type="string"/>
            <element name="cust-age" type="string"/>
            <element name="cust-policy-value" type="string"/>
            <element name="cust-policy-cost" type="string"/>
         </sequence>
      </complexType>
   </element>
</schema>
```

Then a Select statement is written that describes the file. The file control entry in our example the following:

```
        select quote-info assign to "myfile", file status is filestat,
            organization is xml, access mode is xml,
            type is external schema
                          "http://www.schemalocation.com/my_schema".
```

Once the record description in the file section is modified to use the XML extensions, it may read as follows:

```
               01 x-customer identified by "customer".
                02 cust-name pic X(30) identified.
                02 cust-age pic 9(3) identified.
                02 cust-policy-value pic 9(12) identified.
                02 cust-policy-cost pic 9(12) identified.
```

The IDENTIFIED clause is used to associate the XML tag name with a COBOL data item. The COBOL data-name x-customer is associated with the XML tag name customer at the 01 level. For the other data items the COBOL data-name and the XML tag name are the same, so an abbreviated form of the IDENTIFIED clause may be used to specify this association.

One XML document that conforms to the schema shown above is as follows:

```
<customer xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance>
  <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>
```

C.4.3 Transformed main program

Once these environment and data division changes have been made, the procedure division can be modified to write the customer information to an XML file instead of passing it as a parameter.

The format-request paragraph is modified to create an XML file with the information about the customer who wants a price quotation for the insurance policy. The send-request paragraph is modified to remove the parameters from the CALL statement, since the information is communicated by means of the XML file. The display-results paragraph is modified to read the information from the XML file.

The converted program reads as follows:

```
identification division.
program-id.  get-quote.
environment division.
input-output section.
file-control.
    select quote-info assign to "myfile", file status is filestat,
        organization is xml, access mode is xml,
        type is external schema
                      "http://www.schemalocation.com/my_schema".
data division.
file section.
fd quote-info.
copy "mycustxml.cpy".
working-storage section.
01 filestat pic s9(9).
01 screen-status pic x(50).
copy "mycust.cpy".
screen section.
copy "scrn-desc.cpy".
procedure division .
lets-begin.
    perform get-ready
    perform format-request
    perform send-request
    perform display-result
    goback.
get-ready.
    initialize customer
    move "Enter the following information:" to screen-status
    display user-request
    accept user-request.
format-request.
    open output quote-info
    move corresponding customer to x-customer
    write x-customer
    close document quote-info
    close quote-info.
send-request.
    call "calc-quote-xml". *>no argument, use XML file
display-result.
    open input quote-info
    open document quote-info
    read quote-info
    move "The premium for this policy is shown below:"
                             to screen-status
    move cust-policy-cost of x-customer to cust-policy-cost of customer
    display user-request
    close quote-info.
```

Note how few statements are required to create the XML file in the format-request paragraph.

C.4.4 Transformed called program

The program that does the calculation was also easily transformed to use XML and now reads as follows:

```
identification division.
program-id.  calc-quote.
environment division.
input-output section.
file-control.
    select quote-info
        assign to "myfile"
        organization is xml
        document-type is external schema
                   "http://www.schemalocation.com/my_schema"
        file status is filestat.
data division.
file section.
fd quote-info.
copy "mycustxml.cpy".
working-storage section.
01 nr-years-2-pay pic s999.
01 filestat pic s9(9).
procedure division.
start-here.
open-xml-file.
  open i-o quote-info
  open document quote-info
  read quote-info.
calc-as-before.
  compute nr-years-2-pay = 75 - cust-age
  compute cust-policy-cost = cust-policy-value / nr-years-2-pay
update-XML-file.
  rewrite x-customer element is cust-policy-cost
  close document quote-info
  close quote-info
  goback.
```

This example has shown how to create an XML description of data and how to use that description in a program with little impact on the application logic.

Bibliography

[1] ISO/IEC Directives, Part 2, *Rules for the structure and drafting of International Standards*, 2001, 4th edition.

[2] JTC 1 Directives, *Procedures for the technical work of ISO/IEC JTC 1,* 2003.

[3] Merriam-Webster's Collegiate® Dictionary, Tenth Edition; Merriam-Webster, Incorporated, 2001, ISBN 0-87779-709-9.

[4] Extensible Markup Language (XML) 1.1, W3C Recommendation 04 February 2004.