XEDI.ORG

**TECHNICAL PAPER**

# XML Representation of X12 EDI

Large companies have invested millions in developing their existing EDI systems, an investment that should be leveraged, not discarded. The following discussion will explain the XEDI.ORG method of representing EDI messages in XML. This technical paper is provided by XMLSolutions Corporation *http://www.xmls.com* and was prepared by Jeffrey Ricker *ricker@xmls.com*, Drew Munro *dmunro@xmls.com* and Doug Hopeman *hopeman@xmls.com*.

# Table of Contents

# Introduction

Extensible markup language (XML) and the Internet have lowered the barriers to e-commerce in both cost and complexity. The advent of XML, however, should not be interpreted as the end of electronic data interchange (EDI). XML does not replace EDI, but rather extends it, bringing e-commerce to small and midsize companies. XML complements EDI and, in so doing, brings the vision of EDI into reality.

EDI has certain advantages over XML. EDI is a well-proven technology with a 20-year history. It has an installation base of 300,000 companies worldwide. Most importantly, hundreds if not thousands of companies have worked to come to a consensus on EDI's semantics.

The primary disadvantage of EDI is its cryptic message format and costs. The creators of EDI were very concerned about the size of their messages. Bandwidth for EDI networks is very expensive even today. EDI messages are thus very compressed and use codes to represent complex values. All the metadata is stripped from the messages, which makes EDI messages very hard to read and debug.

The power of XML is that it combines metadata with data, making messages very easy to read by both humans and computers. XML defines the syntax for building semantic sets such as open financial exchange (OFX) for consumer banking or human resource exchange (HRX). XML's primary disadvantage today is that consensus has not been achieved on these semantics.

The XEDI.ORG working group deduced that if they could combine the well-proven semantics of EDI with the easy syntax of XML, they would have a very solid base for expanding e-commerce. XEDI.ORG has devised a convention for expressing the EDI X12 semantic in XML named XEDI (zee-dee). This paper describes XEDI and explains some of the thought processes behind it.

## Converting EDI and XML

Large companies will be able to leverage existing EDI systems to send and receive XML messages with their small to mid-size suppliers and vendors. The question that arises is how can you do this effectively and efficiently, while still maintaining the business rules and structure for well-formed EDI documents? One simple approach is to have an EDI-XML translator as an intermediate step.

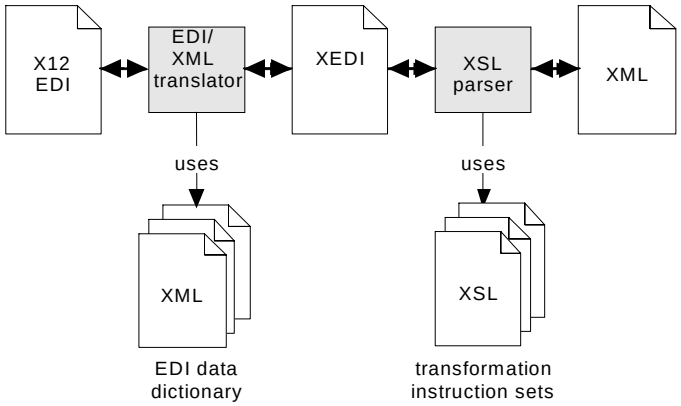


*Figure 1 Transforming EDI and XML*

When converting EDI to XML, the translator uses the X12 data dictionary to transform an EDI message into an XEDI document. Once the EDI message is represented in a well-formed XEDI document an XSL style sheet may be applied to transform the document into HTML for display on the web, or the XEDI document can be passed directly to another application system.

When converting a XEDI document to EDI, the translator will not only function to convert the XML document into EDI, it's X12 data dictionaries will ensure the XEDI document is compliant with a well-formed EDI message. You will be able to apply business rules to in your translation process to ensure the information you pass to your EDI translator is complete and accurate.

Leveraging an EDI-XML translator follows the XEDI.ORG concept of leveraging over 300 well-formed EDI documents for EDI conversions. Keeping the semantics of current EDI document structures allows companies to extend their existing EDI infrastructure without having to change or throw away 20 years of well thought out technology.

# Converting EDI X12 to XML

## Terminology

Before beginning, here is a brief note on terminology. The word *element* has two meanings in this document: X12 element and XML element. The reader hopefully should be able to distinguish the two from the context. To avoid having a third meaning for *element*, the word *piece* will be used to distinguish a portion of data.

## X12 Data Format

To understand the XEDI design decisions for representing X12 in XML, it is important to first understand how a message is formed in the EDI X12 data format. What follows are basic rules for constructing X12 messages, not the complete set of ANSI X12 rules. Readers who are familiar with X12 may wish to skip this section.

An EDI X12 message has three pieces: the transaction set, EDI segments, and EDI elements. Transaction sets are made up of segments, and segments are made up of elements. Each is described below. Note: Bounded loops are used to combine related segments that can occur more than once in a message.

A transaction set is a collection of segments that form a business document such as a purchase order or an invoice. Many transaction sets are divided into three “areas” which generally relate to the format of a paper document: the header area, the detail area, and the summary area.

A segment is a group of logically related information. Two or three alpha/numeric characters like BEG, FOB, or N1, identify segments. Different transaction sets can use the same segment type. Segments can be mandatory or optional, and may occur one time or multiple times depending on the syntax rules for the specific transaction set. One example of a segment is the N1 “Name” segment. The N1 segment combines four elements to identify a party by type of organization, name, and code.

Segments can also be part of a loop. A loop is a group of logically related segments in a defined sequence. Syntax rules for each transaction set will define the number of times a loop can occur. The N1 segment mentioned above is also part of a loop, called the N1 loop. A set of 4 segments make up a basic N1 loop.

```
N1 segment – Used to identify a party by type of organization, name and
code (i.e. Ship to party and related codes)
N2 segment – Used to identify additional names
N3 segment – Used to identify location of the named party (i.e. street
address)
N4 segment – Used to identify the geographic location of the party (i.e.
city, state, and zip code)
```

*Figure 2 Basic N1 Loop*

Elements are the pieces of logically related data that make up a segment. Each element is defined by a 1-4 digit numerical code. Many different segments may use the same element code types. For instance, element code type 98 is used to identify a party or

organization, and can be used in 26 different segments. Elements can be mandatory, optional, or conditional.

Due to the cryptic nature of X12 data, ANSI has created element identifiers that explain another piece of information. Using the example of the N1 segment, if a company transmitted the name John Doe, without qualifying whom John Doe was, that information would be useless. Element type 98 is a qualifier element used to help define "John Doe". *Figure 3* has a partial list of element type 98.

```
BS  Bill and Ship To
BT  Bill-To-Party
BU  Place of Business
…
MF  Manufacturer of Goods
MG  Government Loan Agency Sponsor or Agent
…
ST  Ship To
SU  Supplier Manufacturer
```

*Figure 3 Partial list of element type 98 code values*

In an X12 transaction set, you have to separate segments and elements apart so you know where one piece ends and another begins. The most common way to separate segments are hard carriage line returns, and the most common way to separate elements is with an asterisk. Furthermore, the first element of every segment is its segment code. For example N1 is the segment code for the N1 segment.

By logically placing segment and element information together you can create a transaction set in X12. The figure below represents a small portion of how a transaction set is displayed in X12. To see a complete representation of a Purchase Order message in X12 see Appendix A.

```
N1 Ship to Loop example:

N1*ST*John Doe
N2*Division 1
N3*1000 Park Avenue
N4*New York City*NY*10610
```

*Figure 4 Basic N1 Loop represented in X12*

## X12 Metadata

The ANSI X12 standard is very compact. When X12 was first released, bandwidth was very expensive. The designers stripped out all the metadata to further compress the data. Wherever possible, they substituted text with abbreviated codes like those shown in *figure 3*. The result is that X12 messages are very cryptic. With XML, metadata is stored with the data, making documents as verbose as necessary.

# XEDI

An X12 message has three major pieces: the transaction set, the EDI segments, and the EDI elements. XEDI follows the same structure. The root XML element is named *transactionSet.* The *transactionSet* element contains child XML elements named *segment*. In turn, the XML element named *segment* contains child XML elements named *element*. These three XML elements, *transactionSet*, *segment,* and *element*, are the major pieces of an XEDI document. By converting the three X12 pieces into XML we can leverage all of the semantics of a well-formed EDI message into the syntax of a well-formed XML document. An explanation of the conversion follows.

## Attributes vs elements

When designing an XML document type, the most recurring dilemma is, "Should this piece be an attribute or an element?" XEDI.ORG applies a couple of rules of thumb to resolve this quandary consistently and effectively. If the answer to any one of these rules of thumb is *yes*, then the piece should probably be an element rather than an attribute.

1. **Is it possible to have more than one of these pieces?** For instance, data has a version that it supports. It seems reasonable to have the version as an attribute, just as the XML processing tag does. However, if it is possible that the data could support more than one version, then the version needs to be an element.
2. **Is the piece readable text that is likely to be displayed?** A good example is currency. The code *CAD* could be displayed, but not likely. The phrase *Canadian dollars*, however, has one primary purpose: to be displayed and read by English-speaking users. Thus, *CAD* is a likely attribute but *Canadian dollars* should be the content of an element.

## XML Element Names

XEDI.ORG uses only a handful of XML element names such as *transactionSet*, *segment*, *element value* and *name*. All other information is conveyed as the contents or attributes of these elements.

Each of the three major pieces of a XEDI message has an attribute, called *code,* which contains the EDI identifier. The major pieces also have a child element, called *name,* which contains the human readable name for that piece.

An 850 transaction set is a purchase order. The transaction set attribute, called *code*, is **850** and its child element, called *name*, is **Purchase Order**. A fragment of a transaction set would look like this:

```
<transactionSet code="850">
      <name>Purchase Order</name>
      ...
</transactionSet>
```

*Figure 5 XEDI transaction set representation*

Similarly, EDI segment type NTE is a note. The segment attribute, named *code,* is **NTE** and its child element, called *name*, is **Note**.

```
<segment code="NTE">
      <name>Note</name>
      ...
</segment>
```

*Figure 6 XEDI segment representation*

Finally, EDI element code 98 is called the Entity Identifier Code. The element attribute, named *code*, is **98** and its child element, called *name*, is **Entity Identifier Code.**

```
<element code="98">
  <name>Entity Identifier Code</name>
```

*Figure 7 XEDI element representation*

**EDI Element Values**

The contents of an EDI message are contained within its EDI elements. As seen in *figure 4*, when using EDI element code "**98**", the **Entity Identifier Code**, there are many different values for the code. EDI has many elements that have an abbreviated code that defines its human readable meaning. For example, in EDI the "**Ship To**" address is represented as "**ST**". In XML it is important to capture both the human-readable and machine-readable representations of the EDI element. We accomplish this using an XML element named *value.* The attribute of *value,* called *code,* contains the abbreviated EDI code, and the contents of *value* contain the human readable description. An example appears in *Figure 8.*

```
<element code="98">
  <name>Entity Identifier Code</name>
  <value code="ST">Ship To</value>
</element>
```

*Figure 8 EDI element 98 displayed in XEDI format*

When EDI elements are populated with "Free Form Text", XML simply populates the element *value* with its contents. For example "**John Doe**" would be considered Free Form Text and would be represented in XEDI like this:

```
<element code="93">
  <name>Name</name>
  <value>John Doe</value>
</element>
```

*Figure 9 EDI free form text displayed in XEDI Format*

## **850 XEDI Document Example –** Tying the 3 major pieces together

Employing the previously described conventions for the three major XML pieces it is easy to create a well-formed XEDI document. A portion of an 850 X12 message and its representation in XEDI is shown on the following page.

**Portion of X12 850 Message:**

```
ST*850
. . .
N1*ST*John Doe
. . .
SE*3
```

**Portion of XEDI 850 Message:**

```
<transactionSet code="850">
  <name>Purchase Order</name>
  <segment code="ST">
    <name>Transaction Set Header</name>
    <element code="143">
      <name>Transaction Set Identifier Code</name>
      <value code="850">Purchase Order</value>
    </element>
  </segment>

. . .

  <segment code="N1">
    <name>Name</name>
    <element code="98">
      <name>Entity Identifier Code</name>
      <value code="ST">Ship To</value>
    </element>
    <element code="93">
      <name>Name</name>
      <value>John Doe</value>
    </element>
  </segment>

. . .

  <segment code="SE">
    <name>Transaction Set Trailer</name>
    <element code="96">
      <name>Number of Included Segments</name>
      <value>3</value>
    </element>
  </segment>
</transactionSet>
```

There is no denying that X12 representation in XEDI is verbose. However, XML achieves what EDI could not: it is easily readable by both humans and machines. The size of the message is less of an issue than some might think for two important reasons: cheap bandwidth and easy compression. For a complete example of an XEDI 850 document see Appendix B.

# Benefits of XEDI

Creating X12 XML does not have to be a complex problem. XEDI.ORG believes that the approach outlined in this paper not only leverages existing EDI structures, but also follows best practice approach for creating a well-formed XML document. The benefits of using the XEDI approach include:

1. Use only a handful of elements (tags) creates ease of programming
2. Multilingual implementations
3. Leveraging 20 years of technology

## Element (tags)

XEDI.ORG uses only a handful of element names such as *transactionSet*, *segment*, *element, value* and *name*. All other information is conveyed as the attributes or contents of these elements.

This approach differs strongly from other proposals. For instance, one proposal for XML representation of X12 uses a different XML element name for each X12 segment and element. The result is hundreds of XML element names such as:

- AdvertisingDemographicInformation
- ServiceAllowanceChangeLOOP2
- CarrierDetailsSpecialHandlingOrHazardous-MaterialsOrBoth

Using element names like these creates an environment where style sheets and searching scripts become very large because there is no way to handle elements in a generic sense. It forces the programmer to create a specific instance for every single element type. Such long element names are also hard to remember and are easily mistyped. Other approaches also stymie the ability to quickly modify XEDI messages to a language other than English (see: multiple language support)

XEDI.ORG uses EDI X12 codes as attributes and human readable text as the content of elements, allowing computers to read the codes and display the text for humans. By using only a handful of elements, DTDs are small and style sheets are easy to write. Furthermore, this approach makes the meaning and purpose of data in the XML document obvious without having to refer to an X12 standards manual. By giving programmers both existing EDI semantics and a new XML syntax, this approach allows EDI programmers to leverage what they have used for the last 20 years, and gives XML programmers a human readable version of X12.

## Multilingual Approach

To be able to support the Global 2000, it is important that -XML design is flexibly enough to support multiple languages. XEDI was designed with language support in mind, and with little effort can be transformed from English into any other language. The design

allows for any display value to have multiple instances, and can support as many languages as needed. For example:

```
<element code="363">
      <value code="DEL" lang="EN">delivery</value>
      <value code="DEL" lang="FR">livraison</value>
      </element>
```

*Note: The language abbreviations are the ISO standard two-letter abbreviations.*

It is also conceivable that users might wish to have the actual element tags in their own language. By designing XEDI with only a handful of element tags this too can be accomplished simply. Such a requirement came arose with the Polish Ministry of Finance. In such cases, a simple one-to-one transformation instruction set written in XSL will suffice. Take the following example.

```
<element type="363">
      <value code="DEL" lang="EN">delivery</value>
      <value code="DEL" lang="FR">livraison</value>
      </element>
```

Perhaps we wish to transform the tag names into French as follows.

```
<élément genre="363">
      <valeur code="DEL" langue="FR">livraison</valeur>
      <valeur code="DEL" langue="EN">delivery</valeur>
      </élément>
```

The following XSL transformation would accomplish this feat.

```
<?xml version="1.0"?>
<style-sheet>
      <template match="element">
            <element name="élément">
                  <attribute name="genre"><value-of
select="@type"/></attribute>
                  <apply-templates/>
                  </element>
            </template>
      <template match="value">
            <element name="valeur">
                  <attribute name="code"><value-of
select="@code"/></attribute>
                  <attribute name="langue"><value-of
select="@lang"/></attribute>
                  <value-of/>
                  </element>
            </template>
      </xsl:style-sheet>
```

These examples illustrate the capabilities of XML. They are not necessarily examples of good design. However, the XEDI approach will accommodate the expanding global market.

**Expanding EDI –** Leveraging 20 years of technology

XEDI.ORG's approach to XEDI allows companies to customize their XML documents to match the idiosyncrasies of their EDI approach. The approach requires an EDI-XML translator that can match the EDI X12 message against an X12 XML based data dictionary.

All of the EDI semantics for transaction sets, segments and elements are stored in a data dictionary. XEDI.ORG has created a complete set of X12 dictionaries that are simply a collection of XML documents. XML programmers can modify and customize the data dictionary to meet their company or industry specific trading requirements.

XEDI data dictionaries are automatically generated from the existing EDI ANSI X12 data dictionaries. Every EDI transaction set in every version release of X12 is already available in XML data dictionary form.

Other approaches to X12 XML hard code the particulars of each EDI message in DTD's. If a user makes any slight changes to the format of an EDI message, they have to create a new DTD. If a company does business with thousands of suppliers and customers, having to create a new DTD for the nuances of different trading partners could become very cumbersome. The XEDI.ORG approach allows modification of XML documents using the same DTD, making it easy to integrate with trading partners who have different requirements.

The X12 standards have a wealth of definitions, or semantics, for different types of business information. The advent of XML should not be interpreted as the end of EDI. The goal was not to replace traditional X12 syntax, but to enable XML as an alternate syntax for X12 transaction sets. XML does not replace EDI, but rather extends it by bringing e-commerce to small and midsize companies. XEDI complements EDI and, in so doing, realizes the vision of EDI.

# APPENDIX A
## EDI X12 850 DOCUMENT EXAMPLE

```
ST*850*0001
BEG*00*SA*XX-1234**19980301*AE123
N1*ST*John Doe
N2*Division 1
N3*1000 Park Avenue
N4*New York*NY*10610
PO1*1*25*EA*9.5*CT*MG*XYZ-1234
PID*F****HAMMER-CLAW
PO1*2*75*EA*6.95*CT*MG*L505-123
PID*F****PLIERS 8” – NEEDLE NOSE
CTT*2
AMT*TT*758.75*C
SE*13*0001
```

# APPENDIX B
## XEDI 850 DOCUMENT EXAMPLE

```
<?xml version="1.0" ?>
<transactionSet code="850" version="004010">
   <name>Purchase Order</name>
   <segment code="ST">
   <name>Transaction Set Header</name>
       <element code="143">
      <name>Transaction Set Identifier Code</name>
      <value code="850">Purchase Order</value>
    </element>
       <element code="329">
      <name>Transaction Set Control Number</name>
      <value>0001</value>
    </element>
  </segment>
   <segment code="BEG">
   <name>Beginning Segment for Purchase Order</name>
       <element code="353">
      <name>Transaction Set Purpose Code</name>
      <value code="00">Original</value>
    </element>
       <element code="92">
      <name>Purchase Order Type Code</name>
      <value code="SA">Stand-alone Order</value>
    </element>
       <element code="324">
      <name>Purchase Order Number</name>
      <value>XX-1234</value>
    </element>
       <element code="328">
      <name>Release Number</name>
      <value />
    </element>
       <element code="373">
      <name>Date</name>
      <value>19980301</value>
    </element>
       <element code="367">
      <name>Contract Number</name>
      <value>AE123</value>
    </element>
   </segment>
<loop code="N1">
     <segment code="N1">
     <name>Name</name>
     <element code="98">
     <name>Entity Identifier Code</name>
     <value code="ST">Ship To</value>
   </element>
     <element code="93">
     <name>Name</name>
     <value>john Doe</value>
   </element>
  </segment>
  <segment code="N2">
   <name>Additional Name Information</name>
     <element code="93">
     <name>Name</name>
     <value>DIVISION 1</value>
```

```
   </element>
 </segment>
  <segment code="N3">
  <name>Address Information</name>
    <element code="166">
    <name>Address Information</name>
    <value>1000 Park Ave.</value>
  </element>
 </segment>
  <segment code="N4">
  <name>Geographic Location</name>
    <element code="19">
    <name>City Name</name>
    <value>New York</value>
  </element>
   <element code="156">
    <name>State or Province Code</name>
    <value code="NY" />
  </element>
   <element code="116">
    <name>Postal Code</name>
    <value code="10610" />
  </element>
  </segment>
</loop>
<loop code="PO1">
 <segment code="PO1">
  <name>Baseline Item Data</name>
   <element code="350">
    <name>Assigned Identification</name>
    <value>1</value>
  </element>
   <element code="330">
    <name>Quantity Ordered</name>
    <value>25</value>
    </element>
       <element code="355">
      <name>Unit or Basis for Measurement Code</name>
      <value code="EA">Each</value>
    </element>
       <element code="212">
      <name>Unit Price</name>
      <value>9.5</value>
    </element>
        <element code="639">
      <name>Basis of Unit Price Code</name>
      <value code="CT">Contract</value>
    </element>
       <element code="235">
      <name>Product/Service ID Qualifier</name>
      <value code="MG">Manufacturer's Part Number</value>
    </element>
       <element code="234">
      <name>Product/Service ID</name>
      <value>XYZ-1234</value>
    </element>
  </segment>
 <loop code="PID">
    <segment code="PID">
    <name>Product/Item Description</name>
       <element code="349">
      <name>Item Description Type</name>
      <value code="F">Free-form</value>
    </element>
```

```
      <element code="750">
     <name>Product/Process Characteristic Code</name>
     <value />
   </element>
      <element code="559">
     <name>Agency Qualifier Code</name>
     <value />
   </element>
      <element code="751">
     <name>Product Description Code</name>
     <value />
   </element>
      <element code="352">
      <name>Description</name>
      <value>HAMMER-CLAW</value>
   </element>
 </segment>
</loop>
</loop>
<loop code="P01">
  <segment code="PO1">
  <name>Baseline Item Data</name>
     <element code="350">
    <name>Assigned Identification</name>
    <value>2</value>
  </element>
    <element code="330">
    <name>Quantity Ordered</name>
    <value>75</value>
  </element>
    <element code="355">
    <name>Unit or Basis for Measurement Code</name>
    <value code="EA">Each</value>
  </element>
    <element code="212">
    <name>Unit Price</name>
    <value>6.95</value>
  </element>
    <element code="639">
    <name>Basis of Unit Price Code</name>
    <value code="CT">Contract</value>
  </element>
    <element code="235">
    <name>Product/Service ID Qualifier</name>
    <value code="MG">Manufacturer's Part Number</value>
  </element>
    <element code="234">
    <name>Product/Service ID</name>
    <value>L505-123</value>
  </element>
 </segment>
<loop code="PID">
  <segment code="PID">
  <name>Product/Item Description</name>
     <element code="349">
     <name>Item Description Type</name>
     <value code="F">Free-form</value>
   </element>
      <element code="750">
     <name>Product/Process Characteristic Code</name>
     <value />
   </element>
      <element code="559">
     <name>Agency Qualifier Code</name>
```

```
      <value />
    </element>
       <element code="751">
      <name>Product Description Code</name>
      <value />
    </element>
       <element code="352">
       <name>Description</name>
       <value>PLIERS 8' - NEEDLE NOSE</value>
    </element>
  </segment>
 </loop>
 </loop>
 <loop code="CTT">
   <segment code="CTT">
   <name>Transaction Totals</name>
       <element code="354">
       <name>Number of Line Items</name>
       <value>3</value>
    </element>
  </segment>
   <segment code="AMT">
   <name>Monetary Amount</name>
      <element code="522">
      <name>Amount Qualifier Code</name>
      <value code="TT">Total Transaction Amount</value>
   </element>
     <element code="782">
     <name>Monetary Amount</name>
     <value>758.75</value>
   </element>
     <element code="478">
     <name>Credit/Debit Flag Code</name>
     <value code="C">Credit</value>
   </element>
  </segment>
 </loop>
   <segment code="SE">
   <name>Transaction Set Trailer</name>
       <element code="96">
      <name>Number of Included Segments</name>
      <value>13</value>
    </element>
      <element code="329">
      <name>Transaction Set Control Number</name>
      <value>0001</value>
   </element>
  </segment>
</transactionSet>
```

# APPENDIX C
## X12 DOCUMENT TYPE DEFINITION (DTD)

```
<!-- Copyright (c) 1999 XYZ Corporation.
     Version 1.0  1999-06-22
     file location: http://www.xyzc.com/dtd/edi.dtd
     -->

<!-- Import ISO language codes -->
<!ENTITY % defineLanguageCodes SYSTEM “Languages.mod”>
%defineLanguageCodes;

<!ELEMENT transactionSet (name,(loop?|segment?)*>
<!ATTLIST transactionSet
  code CDATA #REQUIRED
  version CDATA #REQUIRED
  >
<!ELEMENT loop (name,(loop?|segment?)*)>
<!ELEMENT loop
  code CDATA #REQUIRED
  >
<!ELEMENT segment (name,element*)>
<!ATTLIST segment
  code CDATA #REQUIRED
  >
<!ELEMENT element (name,value)>
<!ATTLIST element
  code CDATA #REQUIRED
  >
<!ELEMENT name (#PCDATA)>
<!ATTLIST name
  lang %languageCodes; “EN”
  >
<!ELEMENT value (#PCDATA)>
<!ATTLIST value
  lang %languageCodes; “EN”
  code CDATA #IMPLIED
  >
```

# APPENDIX D

## X12 XML DATA DICTIONARIES

XEDI.ORG has created a data dictionary in XML that describes the X12 data format. An EDI/XML translator uses these data dictionaries to transform X12 EDI into XML. The simplicity of these data dictionaries allows users to extend the dictionaries to handle obscure, modified and even non-standard transaction sets, segments and elements.

### DTD

The data type definition (DTD) for the segment data dictionary is as follows.

```
<!-- Copyright (c) 1999 XYZ Corporation
     Version 1.0  1999-06-22
     file location: http://www.xyzc.com/dtd/x12dd.dtd
     -->

<!ELEMENT transactionSet (name,desc?,version*,(loop|segment)*)>
<!ATTLIST transactionSet
  code CDATA #REQUIRED
  >
<!ELEMENT loop ((loop|segment)*)>
<!ATTLIST loop
  code CDATA #REQUIRED
  >
<!ELEMENT segment (name,desc?,version*,element*)>
<!ATTLIST segment
  code CDATA #REQUIRED
  >
<!ELEMENT element (name,desc?,version*,value*)>
<!ATTLIST element
  code CDATA #REQUIRED
  req (M,O) “O”
  width CDATA #IMPLIED
  ref CDATA #IMPLIED
  >
<!ELEMENT name #PCDATA>
<!ELEMENT desc #PCDATA>
<!ELEMENT version #PCDATA>
<!ELEMENT value #PCDATA>
<!ATTLIST value
  code CDATA #REQUIRED>
```

A description of some of these elements is as follows.

- The segment *code* is the two or three letter code at the beginning of a line that identifies segment type.
- A segment definition can be applicable to several *versions* of X12.
- The element code is the three-digit number used to identify the element.
- Element contains a child *name* in case more children are necessary for different languages.
- X12 elements have two requirements (*req*) settings, manual or optional.
- Some X12 elements have a fixed *width*.
- For elements that are references or qualifiers, the data dictionary provides a URL (*ref*) to an XML document that describes the codes used.

## Transaction Set

```
<?xml version="1.0"?>
<!DOCTYPE segment SYSTEM "http://www.xyzc.com/dtd/x12dd.dtd">
<transactionSet code="840" lang="EN">
      <segment code="ST">Transaction Set Header</segment>
      <segment code="BQT">Beginning Segment for Request For
Quotation</segment>
      <segment code="NTE">Note/Special Instruction</segment>
      <segment code="CUR">Currency</segment>
      <segment code="REF">Reference Numbers</segment>
      <segment code="PER">Administrative Communications
Contact</segment>
      <segment code="TAX">Tax Reference</segment>
      <segment code="FOB">F.O.B. Related Instructions</segment>
      <segment code="CTP">Pricing Information</segment>
      <segment code="CSH">Header Sale Condition</segment>
      <segment code="SAC">Service, Promotion, Allowance, or Charge
Information</segment>
      <segment code="ITD">Terms of Sale/Deferred Terms of Sale</segment>
      <segment code="DIS">Discount Detail</segment>
      <segment code="DTM">Date/Time Reference</segment>
      <segment code="LDT">Lead Time</segment>
      <segment code="LIN">Item Identification</segment>
      <segment code="PID">Product/Item Description</segment>
      <segment code="MEA">Measurements</segment>
      <segment code="PWK">Paperwork</segment>
      <segment code="PKG">Marking, Packaging, Loading</segment>
      <segment code="TD1">Carrier Details (Quantity and
Weight)</segment>
      <segment code="TD5">Carrier Details (Routing Sequence/Transit
Time)</segment>
      <segment code="TD3">Carrier Details (Equipment)</segment>
      <segment code="TD4">Carrier Details (Special Handling or Hazardous
Mat</segment>
      <segment code="MAN">Marks and Numbers</segment>
      <segment code="RRA">Required Response</segment>
      <loop code="N9">
            <segment code="N9">Reference Number</segment>
            <segment code="MSG">Message Text</segment>
            </loop>
      <loop code="N1">
            <segment code="N1">Name</segment>
            <segment code="N2">Additional Name Information</segment>
            <segment code="N3">Address Information</segment>
            <segment code="N4">Geographic Location</segment>
            <segment code="REF">Reference Numbers</segment>
            <segment code="PER">Administrative Communications
Contact</segment>
            <segment code="FOB">F.O.B. Related Instructions</segment>
            <segment code="TD1">Carrier Details (Quantity and
Weight)</segment>
            <segment code="TD5">Carrier Details (Routing
Sequence/Transit Time)</segment>
            <segment code="TD3">Carrier Details (Equipment)</segment>
            <segment code="TD4">Carrier Details (Special Handling or Haz
Mat)</segment>
            <segment code="PKG">Marking, Packaging, Loading</segment>
            <segment code="RRA">Required Response</segment>
            </loop>
      <loop code="PO1">
            <segment code="PO1">Baseline Item Data</segment>
            <segment code="CUR">Currency</segment>
```

```
            <segment code="PO3">Additional Item Detail</segment>
            <segment code="CTP">Pricing Information</segment>
            <segment code="MEA">Measurements</segment>
            <loop code="PID">
                  <segment code="PID">Product/Item Description</segment>
                  <segment code="MEA">Measurements</segment>
                  </loop>
            <segment code="PWK">Paperwork</segment>
            <segment code="PKG">Marking, Packaging, Loading</segment>
            <segment code="PO4">Item Physical Details</segment>
            <segment code="REF">Reference Numbers</segment>
            <segment code="PER">Administrative Communications
Contact</segment>
            <segment code="SAC">Service, Promotion, Allowance, or Charge
Informati</segment>
            <segment code="IT8">Conditions of Sale</segment>
            <segment code="ITD">Terms of Sale/Deferred Terms of
Sale</segment>
            <segment code="DIS">Discount Detail</segment>
            <segment code="TAX">Tax Reference</segment>
            <segment code="FOB">F.O.B. Related Instructions</segment>
            <segment code="SDQ">Destination Quantity</segment>
            <segment code="DTM">Date/Time Reference</segment>
            <segment code="SCH">Line Item Schedule</segment>
            <segment code="FST">Forecast Schedule</segment>
            <segment code="TD1">Carrier Details (Quantity and
Weight)</segment>
            <segment code="TD5">Carrier Details (Routing
Sequence/Transit Time)</segment>
            <segment code="TD3">Carrier Details (Equipment)</segment>
            <segment code="TD4">Carrier Details (Special Handling or Haz
Mat)</segment>
            <segment code="MAN">Marks and Numbers</segment>
            <segment code="RRA">Required Response</segment>
            <segment code="MSG">Message Text</segment>
            <loop code="LDT">
                  <segment code="LDT">Lead Time</segment>
                  <segment code="QTY">Quantity</segment>
                  </loop>
            <loop code="SLN">
                  <segment code="SLN">Subline Item Detail</segment>
                  <segment code="PID">Product/Item Description</segment>
                  </loop>
            <loop code="N9">
                  <segment code="N9">Reference Number</segment>
                  <segment code="MSG">Message Text</segment>
                  </loop>
            <loop code="N1">
                  <segment code="N1">Name</segment>
                  <segment code="N2">Additional Name
Information</segment>
                  <segment code="N3">Address Information</segment>
                  <segment code="N4">Geographic Location</segment>
                  <segment code="REF">Reference Numbers</segment>
                  <segment code="PER">Administrative Communications
Contact</segment>
                  <segment code="FOB">F.O.B. Related
Instructions</segment>
                  <segment code="SCH">Line Item Schedule</segment>
                  <segment code="TD1">Carrier Details (Quantity and
Weight)</segment>
                  <segment code="TD5">Carrier Details (Routing
Sequence/Transit Time)</segment>
```

```
                  <segment code="TD3">Carrier Details
(Equipment)</segment>
                  <segment code="TD4">Carrier Details (Special Handling
or Haz Mat)</segment>
                  <segment code="PKG">Marking, Packaging,
Loading</segment>
                  <segment code="RRA">Required Response</segment>
                  <segment code="CTP">Pricing Information</segment>
                  <segment code="LDT">Lead Time</segment>
                  <segment code="MAN">Marks and Numbers</segment>
                  <segment code="QTY">Quantity</segment>
                  </loop>
            <segment code="CTT">Transaction Totals</segment>
            <segment code="SE">Transaction Set Trailer</segment>
            </loop>
      </transactionSet>
```

### Segment

The following is a sample data dictionary entry for segment type *BEG*.

```
<?xml version="1.0"?>
<!DOCTYPE segment SYSTEM “http://www.xyzc.com/dtd/x12dd.dtd”>
<segment code="BEG" lang=”EN”>
      <name>Beginning Segment for Purchase Order</name>
      <desc>To indicate the beginning of the purchase order transaction
set and transmit identifying numbers and dates.</desc>
      <version>003040</version>
      <element code="353" req="M" ref="e353.xml"><name>Transaction Set
Purpose Code</name></element>
      <element code="92" req="M" ref="e92.xml"><name>Purchase Order Type
Code</name></element>
      <element code="324" req="M"><name>Purchase Order
Number</name></element>
      <element code="328" req="O"><name>Release Number</name></element>
      <element code="323" req="M"><name>Purchase Order
Date</name></element>
      <element code="367" req="O"><name>Contract Number</name></element>
      <element code="587" req="O" ref="e587.xml"><name>Acknowledgment
Type</name></element>
      <element code="1019" req="O" ref="e1019.xml"><name>Invoice Type
Code</name></element>
      </segment>
```

## Element

The following is a sample data dictionary entry for element type 587.

```
<?xml version="1.0"?>
<!DOCTYPE element SYSTEM “http://www.xyzc.com/dtd/x12dd.dtd”>
<element code="587" lang=”EN”>
      <name>Acknowledgment Type</name>
      <desc>Code specifying the type of acknowledgment.</desc>
      <version>003040</version>
      <value code="AC">Acknowledge - With Detail and Change</value>
      <value code="AD">Acknowledge - With Detail</value>
      <value code="AE">Acknowledge - With Exception Detail Only</value>
      <value code="AH">Acknowledge - Hold Status</value>
      <value code="AK">Acknowledge - No Detail or Change</value>
      <value code="AP">Acknowledge - Product Replenishment</value>
      <value code="AT">Accepted</value>
      <value code="NA">No Acknowledgment Needed</value>
      <value code="RD">Reject with Detail</value>
      <value code="RF">Reject with Exception Detail Only</value>
      <value code="RJ">Rejected - No Detail</value>
      <value code="RO">Rejected With Counter Offer</value>
      <value code="ZZ">Mutually Defined</value>
      </element>
```