C6. CHAPTER 6 – STANDARDS AND CONVENTIONS

C6.1. PURPOSE. The purpose of this chapter is to assist the reader in understanding the basic concepts and semantics of the standards involved in processing logistics transactions: Defense Logistics Standard Systems (DLSS); American National Standards Institute (ANSI) Accredited Standards Committee (ASC) X12 (hereafter referred to as ASC X12) Standards; and Extensible Markup Language (XML) standards.

C6.2. DEFENSE LOGISTICS STANDARD SYSTEMS/MILITARY STANDARD SYSTEMS. DLSS are commonly referred to as Military Standard Systems (MILS) and are legacy 80 record position, fixed-length, DoD-unique standards for DoD logistics transactions.

C6.2.1. Developed in the 1960s, each DoD logistics transaction was based on the 80-record position (fixed-length) punch card. Each record position (column) on the punch card contains a datum as defined in the requirements of that particular transaction. Table C6.T1. is an example of two data items, their record positions and their associated values:

Table C6.T1. MILS Record-Position Example

Record Position	Name	Description
1-3	Document Identifier Code (DIC)	A three-position code that indicates the purpose and use of the document. An example of a DIC is A0A, which stands for domestic shipment/with National Stock Number (NSN)/North Atlantic Treaty Organization (NATO) stock number.
4-6	Routing Identifier Code (RIC)	A three-position code used to represent the recipient of the document. An example of a RIC is SMS, which identifies Defense Logistics Agency (DLA).

C6.2.2. Since their inception, the DLSS legacy formats have provided the backbone of cross-functional interoperability between DoD Components and their commercial trading partners. However, the rigid fixed-length formats are functionally constrained, technologically obsolete, and unable to support current DoD business goals.

C6.3. ASC X12 STANDARDS. In 2000, the DoD issued a directive that mandated the use of Electronic Data Interchange (EDI) Standards for the exchange of DoD logistics business transactions (DoDD 8190.01E "Defense Logistics Management Standards (DLMS)," January 9, 2015). This means that logistics transactions must migrate from DLSS legacy standards to the DLMS. DoD adopted the ASC X12 EDI standards as the basis for the DLMS.

The ASC X12 standards define commonly used business transactions in a formal, structured manner called transaction sets. The structure of the transaction set comprises specific syntax rules for the EDI constructs. The standard defines the data elements, codes, and segments within each transaction set. Most importantly, it also defines specific rules and formats for the content of data within the data elements.

C6.4. STRUCTURE OF EDI TRANSMISSION. To allow different types of transaction sets to be transmitted from one party to another in the same transmission, a hierarchical structure of headers and trailers allows the data to be segregated logically for easy interpretation by the transmitter and receiver. Figure C6.F1. shows an example of the EDI structure.

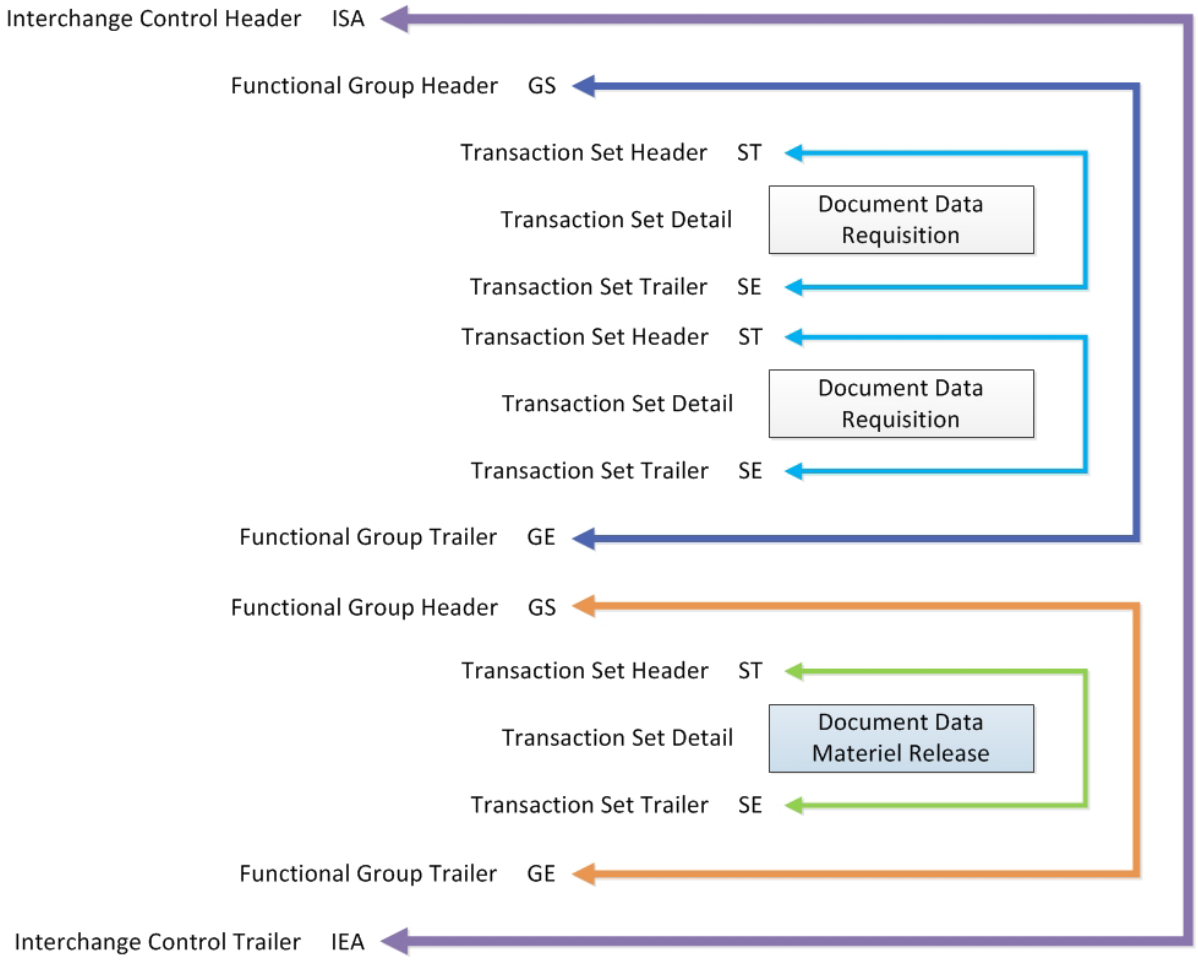

Figure C6.F1. EDI Structure Example[1]

C6.4.1. Interchange Control Header (ISA) and Trailer (IEA) Segments. Interchange Control consists of one or more Functional Groups enclosed in an envelope defined by an ISA Interchange Control Header segment and ending with an IEA Interchange Control Trailer segment. Details of the envelope:

- Contains the structured mailbox address of the sender and the receiver.
- Contains control numbers and counts of the different types of functional groups inside.

[1] Each layer of the EDI enveloping structure and transaction set detail is described below, beginning with the outer layer (Interchange Control Header/Trailer) and moving to the innermost layer (Transaction Set Details).

- Contains a time/date stamp.
- Specifies the format and version of the interchange envelopes.
- Specifies the characters used for data element delimiters (separators) and segment terminators.

C6.4.2. Functional Group Header (GS) and Trailer (GE) Segments. A Functional Group is a group of one or more related Transaction Sets within an EDI transmission. Functional Groups start with a GS Functional Group Header segment and end with a GE Functional Group Trailer segment. The details in the Functional Group GS/GE envelope are often used to route the group's transaction sets to the target environment. Functional Group detail:

- Contains a functional group ID (e.g., RN (511), MD (527)).
- Contains transaction set counts and functional group control numbers.
- Contains a time/date stamp of when the group was generated.
- Provides format, version, and release specifications of the transactions within the group.

C6.4.3. Transaction Set Header (ST) and Trailer (SE) Segments. The Transaction Set Header and Trailer are used to uniquely identify the transaction set. The transaction set begins with an ST Transaction Set Header segment and ends with an SE Transaction Set Trailer segment.

C6.4.3.1. Transaction Set Header. The Transaction Set Identifier Code (ST01) is the first data element of the transaction set header segment. It is used by the translation routine of the interchange partners to select the appropriate transaction set definition (e.g., 511 selects the Requisition transaction set). The Transaction Set Control Number (ST02) uniquely identifies an instance of the transaction set and is assigned by the originator of a transaction set. The control number in ST02 must match the control number in SE02. Some DLMS transactions use the ASC X12 version release 4030 which contains an additional data element in the ST Segment; the Implementation Convention Reference (ST03) uniquely identifies the DLMS IC used in the transaction.

C6.4.3.2. Transaction Set Trailer. The purpose of the transaction set trailer is to indicate the end of the transaction set and provide the count of the transmitted segments (including the beginning (ST) and ending (SE) segments). The number of the included segments (SE01) is used to indicate the end of the transaction set and provide the count of the transmitted segments (including the beginning (ST) and ending (SE) segment). The Transaction Control Number (SE02) must match the one in ST02 to ensure that entire transaction set was received. Figure C6.F3. shows an example of the Header and Footer segments.

ST* <...data, separated by *...>
<multiple transaction loops and segments>
SE* <...data, separated by *...>
ST* <...data, separated by *...>
<multiple transaction loops and segments>
SE* <...data, separated by *...>

Figure C6.F2. ST and SE Header/Footer Example

C6.4.4. Transaction Set Detail (Data) Segments. A Transaction Set is a group of data segments, as defined by the X12 Standard, conveyed between trading partners. The information, in the form of a transaction set, is generally patterned after a conventional paper document, such as a requisition or invoice.

C6.4.4.1. A Transaction Set consists of a number and name (e.g., 511 Requisition), purpose, Functional Group ID, table listing the included segments, their position numbers, requirement designation, maximum usage, and loop repeat counts.

C6.4.4.2. The Transaction Set Detail comprises data elements and data segments specific to the business (requisition) transaction. Examples of data in the detail section are identity of ordering activity, item ordered, quantity, order priority, delivery point, and identity of paying activity.

C6.4.4.3. Data Element. The data element is the smallest named unit of information in the standard. A simple data element is equivalent to a field in a data dictionary. It has a name, a data element number, a brief description, a data type, and a minimum and maximum length. When a group of two or more simple data elements are linked together to form a single data element, they are referred to as a composite data structure.

C6.4.4.3.1. Data Element Types. There are seven types of data elements identified in Table C6.T2.

Table C6.T2. Data Element Types

Data Element Type	Data Element Type Description
AN – Alphanumeric String	Sequence of letters, numbers, spaces, and/or special characters. The contents are left-justified and trailing spaces should be suppressed.
B – Binary	Any sequence of octets ranging in value from binary 0000000 to 1111111. This data element type has no defined maximum length. Actual length is specified by the immediately preceding data element. The binary data element type may only exist in the Binary segment and is not used in the DLMS at this time.
DT – Date	Used to express the standard date in (CC)YYMMDD format in which CC is the century, YY is the year, MM is the month (01 to 12), and DD is the day of the month (01 to 31). DLMS require the use of century to satisfy Y2K compliance.
ID – Identifier	Contains a unique value from a predefined list of values maintained by ASC X12, the DoD, or other responsible organization referenced by the data element dictionary. The contents are left-justified and trailing spaces should be suppressed. Identifier type data elements are frequently used as qualifiers to identify by code the type of information contained in an associated data element. For example, the identifier type data element, Product/Service ID Qualifier, may be transmitted with a value of FS to indicate that the value contained in the associated data element Product/Service ID is a national stock number. In this instance, the list of valid identifier codes is maintained by X12. The conventions normally specify which of these values are permissible entries for the specific use under DLMS.
Nn – Numeric	Represented by one or more digits with an optional leading sign representing a value in the normal base of 10. The value of a numeric data element includes an implied decimal point. It is used when the position of the decimal point within the data is permanently fixed and is not to be transmitted with the data. The symbol for this data element type is Nn where "N" indicates that it is numeric and "n" indicates the number of decimal positions to the right of the implied decimal point. If no decimal positions are allowed, the symbol is written as N or N0. A leading minus sign (-) is used to express negative values. Absence of a sign indicates positive value. Leading zeros should be suppressed unless necessary to satisfy a minimum length requirement. The length of a numeric type data element does not include the optional minus sign. For example, where the numeric type is N2 (indicating an implied decimal placement two positions from the right), the value -123.4 would be transmitted as -12340. The length of the value within the data stream is five.
R – Decimal Numeric	Contains an explicit decimal point and is used for numeric values that have a varying number of decimal positions. The decimal point is always carried in the transmission unless it occurs at the right end of the value. A leading minus sign (-) is used to express negative values. Absence of a sign indicates positive value. Leading zeros should be suppressed unless necessary to satisfy a minimum length requirement. Trailing zeros following the decimal point should be suppressed unless used to express precision. Use of commas within the numeric value is prohibited. The length of a numeric type data element does not include the optional minus sign or the decimal point. For example, the numeric value - 123.45 would be transmitted as -123.45. The length of this entry is five.
TM – Time	Used to express the time in HHMMSSdd format in which HH is the hour for a 24-hour clock (00 to 23), MM is the minute (00 to 59), SS is the second (00 to 59) and dd is the decimal seconds. Seconds and decimal second are optional. Trailing zeros in decimal seconds should be suppressed unless necessary to satisfy a minimum length requirement or unless necessary to indicate precision.

C6.4.4.3.2. Data Element Length. Each data element is assigned a minimum and maximum length, which may be the same. The length of the data element value is the number of character positions used except as noted for numeric, decimal, and binary elements. A data element is of variable length unless the minimum and maximum lengths are equal, in which case it is of fixed length. The length attribute of a data element is expressed as minimum length / maximum length, (e.g., 2/30).

C6.4.4.4. Data Segment. The data segment comprises simple data elements and/or composite data structure(s) and separators, as an intermediate unit of information in a transaction set. Each data segment has a unique segment ID and is used to convey a grouping of functionally related user information.

C6.4.4.4.1. Condition Designator. The condition designator (or requirement designator) is used to define the circumstances under which a data element is required to be present or absent in a particular usage. These conditions are of three basic types: mandatory, optional, and conditional. Under DLMS, optional and conditional designations can be further defined as either recommended or required. Condition designators shown in Table C6.T3. are identified by the symbol as specified in parentheses.

Table C6.T3. Condition Designators

Condition Designator	Condition Designator Definition
Mandatory (M)	The designation of mandatory is absolute in the sense that there is no dependency on other data elements within the segment or composite data structure. A mandatory data element must appear in the segment.
Optional (O)	The designation of optional means that there is no syntactic requirement for the presence of the data element within the segment or composite data structure. Optional data elements may be included or omitted based upon instructions provided in the DLMS ICs or at the discretion of the transmitting activity (as applicable).
Conditional (X)	A designation of conditional defines a special relationship between two or more data elements within a segment or composite data structure. Relational conditions are based upon the presence of one of those data elements. The specific relationship is defined in a syntax note. The first character of the syntax note identifies one of the following conditions: (1) Paired (P). If any specified data element is present, then all of the specified data elements must be present. (2) Required (R). At least one of the specified data elements must be present. (3) Exclusion (E). Not more than one of the specified data elements may be used. (4) Conditional (C). If the specified data element is present, then all other specified data elements must be present. However, any or all of the data elements not specified as the first in the condition may appear when the first is not present. (5) List Conditional (L). If the first specified data element is present, then at least one of the remaining specified data elements must be present. However, any or all of the data elements not specified as the first may appear when the first is not present.

C6.4.4.4.2. Data Segment Loops. Data Segment Loops are groups of two or more data segments that represent a block of related information in a Transaction Set. Different loops may be nested within each other, and loops may repeat up to the maximum loop occurrences specified within the Transaction Set. In some cases, it may be specified as having an unlimited number of occurrences (noted as ">1"). Loops can be Unbounded or Bounded as defined in the X12 Standard.

C6.4.4.4.2.1. Unbounded. An Unbounded loop starts with a specific segment, and all of the other segments in the loop may be considered children of that segment. To establish the iteration of a loop, the first data segment in the loop must appear once and only once in each iteration. Loops may have a specified maximum number of repetitions. A specified sequence of segments is in the loop. Loops themselves are optional or mandatory. The requirement designator of the beginning segment of a loop indicates whether at least one occurrence of the loop is required. Each appearance of the beginning segment defines a new occurrence of the loop. The requirement designator of any segment within the loop after the beginning segment applies to that segment for each occurrence of the loop. If there is a mandatory requirement designator for any data segment within the loop after the beginning segment, that data segment is mandatory for each occurrence of the loop. If the loop is optional, the mandatory segment only occurs if the loop occurs.

C6.4.4.4.2.2. Bounded. The characteristics of unbounded loops described previously also apply to bounded loops. In addition, bounded loops require a Loop Start Segment (LS) to appear before the first occurrence of the loop and a Loop End Segment (LE) to appear after the last occurrence of the loop. If the loop does not occur, the LS and LE segments are suppressed.

C6.4.4.5. EDI fields and records are separated by delimiter characters. The delimiter for a field and the delimiter for a record are set externally by the Interchange Control Header (ISA) segment. This means, the EDI parser may not know what the delimiters will be until it has begun to parse the file. EDI handles this problem by making the first segment, ISA, fixed length and defining the delimiters in the ISA segment of the EDI interchange. In an actual interchange, ASCII Hexadecimal characters are used, a graphic representation is used for print examples.

C6.4.4.5.1. Delimiters. In ASC X12 EDI interchanges (Releases 4010 and 4030), there are three delimiters. The delimiters cannot appear as a value in the business transaction; otherwise, the syntax rule will fail.

C6.4.4.5.1.1. Data Element Separator. The first delimiter is the data element separator. This defines the delimiter between each field within the record. This character will likely be the most common character used for any given EDI file.

C6.4.4.5.1.2. Component Element Separator. The second, and least commonly used, is the component element separator. ASC X12 supports the use of sub-elements in transactions employing a Composite data element such as in the Unit of Measure (MEA) and Reference (REF) segments. The component element separator delimits the sub-elements.

C6.4.4.5.1.3. Segment Terminator. Lastly, the segment terminator defines the end of each segment within the transaction.

C6.4.4.5.2. EDI Interchange and Delimiter Example. Figure C6.F3. shows an example of the EDI data in an interchange that includes the delimiters.

ISA*00* *00* *01*1515151515 *01*5151515151 *041201*1217*U*00403*000032123*0*P**~ GS*CT*9988776655*1122334455*20041201*1217*128*X*004030~ ST*831*00128001~ BGN*00*88200001*20041201~ N9*BT*88200001~ TRN*1*88200001~ RCD*1*20*EA\2\1~ AMT*2*100000.00~ QTY*46*1~ SE*8*00128001~ GE*1*128~ IEA*1*000032123~
Data Element Separator = * (Asterisk). Defined in the fourth position of the ISA Segment Component Element Separator = \ (Back slash). Defined in the 3rd to last position of ISA segment Segment Terminator = ~ (Tilde). First occurrence defines the segment termination

Figure C6.F3. ASC X12 Delimiters

C6.4.4.6. Special Character Use in DLMS Transaction.

C6.4.4.6.1. XML Reserved Characters. DLMS develops and publishes XML schemata (paragraph C6.5.) that are equivalent to the X12-based DLMS ICs. Because the W3C XML standard defines a number of reserved characters that senders may not convey in the XML data element values, senders of X12-based transactions also may not convey these characters in DLMS X12-based transactions as data element values, because they will result in errors if the X12-based transactions are translated to XML.

C6.4.4.6.2. DoD Allowed Special Characters. DoD has identified a number of special characters as valid for use in specific DoD data values, (e.g., the dash (-) and the slash (/) characters are valid in a unique item identifier). Because trading partners may legitimately convey these characters in their EDI data content, senders must not use these special characters as delimiters in DLMS transaction that may require the use of these characters in the transaction data. To avoid any possibility of this type of data collision, DLMS procedures do not allow the use of these DoD allowed special characters as X12 EDI delimiters in any DLMS transactions. Table C6.T4., Special Characters Allowed as Delimiters in X12-based DLMS Transactions, lists the characters that are valid for use as X12 delimiters in DLMS transactions. Senders may choose delimiters from among this list and encode those delimiters in the ISA segment of the DLMS transactions they send. Table C6.T5., Preferred Special Characters as Delimiters for use in X12-based DLMS Transactions, lists the special characters that are preferred for use by type of delimiter.

C6.4.4.6.3. In addition to the above, see Volume 2, Chapter 17 for SDR special character exceptions/inclusions.

Table C6.T4. Special Characters Allowed as Delimiters in X12-Based DLMS Transactions

Authorized Characters	Meaning
!	Exclamation Mark
“	Double Quote
&	Ampersand
‘	Single Quote
*	Asterisk
:	Colon
%	Percent Sign
_	Underscore
{	Open Bracket
}	Close Bracket
|	Pipe (Vertical Bar)
<	Less Than
>	Greater Than
~	Tilde
^	Caret
1D (hex value)	Group Separator[2]
1F (hex value)	Unit Separator[3]
1C (hex value)	File Separator[4]
0D 0A (hex value)	Newline[5] (Line Feed/Carriage Return)

[2] Group Separator is an unprintable character; senders may use it only as a data element separator in X12 transactions.
[3] Unit Separator is an unprintable character; senders may use it only as a component element separator.
[4] File Separator is an unprintable character; senders may use it only as a segment terminator.
[5] Newline is an unprintable character; senders may use it only as a segment terminator.

Table C6.T5. Preferred Special Characters as Delimiters in X12-Based DLMS Transactions

Delimiter Type		Preferred Character	Meaning
Data Element Separator	<gs>	1D (hex value)	Group Separator
		*	Asterisk
		~	Tilde
Component/Composite Element Separator	<us>	1F (hex value)	Unit Separator
		:	Colon
Segment Terminator	<tr>	1C (hex value)	File Separator
		0D 0A (hex value)	Newline (Line Feed/Carriage Return)

C6.5. XML STANDARDS. DLMS use XML as an alternative to EDI for exchanging data between logistics trading partners. XML offers a flexible way to describe and tag content (data, word, phase, etc.) in a structured way. The XML standard emphasizes simplicity and usability over the Internet. It is a textual data format with worldwide support. Though originally designed to focus on documents, it is widely used to represent data structures (e.g., DLMS) and is the foundation of web services. XML only refers to the data; the XML Schema (e.g., XSD file) is used to express the set of business rules to which the XML must conform to be considered valid. The schema is an abstract collection of metadata components. The XML instance document is validated against the schema (a process known as the assessment) prior to sending the transaction for processing. This validation ensures required fields are present, the elements are in the correct format and valid codes are used (when defined in the schema).

C6.5.1. Well-Formed. The XML specification defines an XML document as text that is well-formed; for example, it satisfies a list of syntax rules provided in the specification. Some of the key criteria are:

C6.5.1.1. It contains only properly encoded legal Unicode characters.

C6.5.1.2. None of the special syntax characters such as “<” and “&” appear except when performing their markup-delineation roles.

C6.5.1.3. The beginning, ending, and empty-element tags that delimit the elements are correctly nested, with none missing and none overlapping.

C6.5.1.4. The element tags are case-sensitive; the beginning and end tags must match exactly. Tag names cannot contain any of the characters !"#$%&'()*+,/;<=>?@[\]^`{|}~, nor a space character, and cannot start with -, ., or a numeric digit.

C6.5.1.5. There is a single "root" element that contains all the other elements. The XML instance document must adhere to all the rules of a well-formed file, or it is not XML. An XML processor that encounters violation of the well-formed rules is required to report such errors and to cease normal processing.

C6.5.2. In addition to being well-formed, DLMS XML must be valid. This means that it contains a reference to a schema (XSD file) and that its elements and attributes are declared in that schema and follows the grammatical rules for them that the schema specifies. Additional usage information is further described in Chapter 8.

C6.5.3. XML Tags. XML and EDI tag names are similar, but XML fields and records are handled differently than in EDI. In EDI, data is separated by delimiters. In XML, documents are comprised of markup code to delimit content. Markup and content are distinguished by syntactic rules. All strings that constitute markup begin with the character < and end with a >. These bracketed strings are called XML tags. Strings of characters that are not XML tags are content.

C6.5.3.1. XML tags define the beginning and end of each section of the XML transaction. The start tag contains the field or record name. The end tag will use the same name but will be preceded by a forward slash. Anything in between the two tags is content. For example, to define the value 1000 in the quantity field the XML might appear as <quantity>1000</quantity>. Figure C6.F6. shows the hierarchy.

```
<segment>
<code>ISA</code>
<element>00</element>
<element>          </element>
<element>00</element>
<element>          </element>
<element>01</element>
<element>1515151515     </element>
.
.
.
</segment>
```

Figure C6.F6. XML Hierarchy

C6.5.3.2. XML is self-validating. Each DLMS XML transaction has an XSD (XML Schema Definition) file. The XSD defines the data types (e.g., string, numeric, binary) and detailed constraints (e.g., size, optional/required, enumeration value (lookup table), and format). The process of checking to see if an XML transaction conforms to a schema is called validation, which is separate from XML's core concept of being syntactically well formed. All XML transactions must be well formed or they cannot be parsed. The schema ensures the transaction conforms to the process rules. Validation of an instance transaction against a schema can be regarded as a conceptually separate operation from XML parsing. In practice, the schema validation is integrated within the XML parser.