TRAILING DELIMITERS AND 999 RESPONSE

RFI 2010-14, JANUARY 2011

REQUEST

We have noticed new wording in version 5010 which is causing disagreement between our users. Below are the related changes (referenced from the HIPAA IGs/TR3's and the X12 CHM):

Old wording:
HIPAA 4010 IG A.1.3.9 Absence of Data "Optional simple data elements and/or composite data structures and their preceding data element separators that are not needed should be omitted if they occur at the end of a segment."

New wording:
HIPAA 5010 TR3 B.1.1.3.10 Absence of Data "Optional simple data elements and/or composite data structures and their preceding data element separators that are not needed must be omitted if they occur at the end of a segment."

New wording:
X12 5010 CHM 3.7.4 Absence of Data "Optional simple data elements and composite data structures and their preceding data element separators that are not needed shall be omitted if they occur at the end of a segment; ..."

For example: Assuming that * is the data element separator and ~ is the segment terminator and UM05 and UM06 are both optional data elements in the segment (sic).

UM*SC*I*3*11:B**~

If we are interpreting the above rule correctly then this situation must result in an X12 EDI syntax error. However, we are still seeing data being traded with this excess trailing data element separators (and with excess trailing composite separators). If this is to be trapped and reported, what IK304/IK403 would be the clearest reporting on the 999?

REFERENCED ASC X12 STANDARDS

An RFI applies to a specific version of the ASC X12 Standards. The author failed to provide a specific version of the standard in the request. We have chosen to base this response on Version 5 Release 1 of the ASC X12 Standard. As the areas of the ASC X12 Standards applicable to this interpretation have been relatively stable over time, it is likely that the same interpretation would be provided for earlier versions of the ASC X12 Standards.

X12.6 – Application Control Structures
3.6 Composite Data Structure
The composite data structure is an intermediate unit of information in a segment. By definition, a composite data structure consists of two or more component data elements preceded by a data element separator. In use, in the actual data stream, a composite data structure may appear as only one component data element. Each component data element within the composite data structure, except the last, is followed by a component element separator. The final component data element is followed by the next data element separator or the segment terminator. Trailing component data element separators <us> shall be suppressed. Composite data structures are defined in a composite data structure directory. The directory defines each composite data structure by its name, purpose, reference identifier, and included component data elements in a specified sequence.

A composite data structure is constructed in the following manner:

definition:
<composite_data_structure> ::= <component_data_element> <us>
<component_data_element> {<us>
<component_data_element>}

use:
<composite_data_structure> ::= {[<component_data_element>]
<us>} <component_data_element>

3.7 Data Segment

The data segment is an intermediate unit of information in a transaction set. A data segment consists of a segment identifier; one or more composite data structures or simple data elements, each of which may be permitted to repeat, when so indicated in the segment specification. Adjacent non-repeating simple data elements and composite data structures shall be separated by a data element separator. Adjacent occurrences of the same repeating simple data element or composite data structure in a segment shall be separated by a repetition separator. The data segment shall end with a segment terminator. Trailing data element separators <gs> and trailing repetition separators <rs> shall be suppressed. Data segments are defined in a data segment directory. The directory defines each segment including the segment's name, purpose, and identifier. The directory also defines composite data structures and data elements that a segment contains in their specified order. A data segment is constructed in the following manner:

definition:
<data_segment> ::= <seg_id> <gs> <data_segment_unit> {<gs>
<data_segment_unit>} <tr>
<data_segment_unit> ::= <repeating_simple_data_element> |
<repeating_composite_data_structure>
<repeating_simple_data_element> ::= <simple_data_element>
{<rs> <simple_data_element>}

<repeating_composite_data_structure> ::=
<composite_data_structure> {<rs>
<composite_data_structure>}

use:
<data_segment> ::= <seg_id> {<gs>
[<data_segment_unit>]} <gs> <data_segment_unit>
<tr>
<repeating_simple_data_element> ::=
{[<simple_data_element>] <rs>}
<simple_data_element>
<repeating_composite_data_structure> ::=
{[<composite_data_structure>] <rs>}
<composite_data_structure>

999 005010X231A1 – Implementation Acknowledgement for Health Care

IK304
REQUIRED IK304 620 Implementation Segment Syntax Error Code O 1 ID 1/3
Code indicating implementation error found based on the syntax editing of a segment
CODE DEFINITION
1 Unrecognized segment ID

2 Unexpected segment
3 Required Segment Missing
4 Loop Occurs Over Maximum Times
5 Segment Exceeds Maximum Use
6 Segment Not in Defined Transaction Set
7 Segment Not in Proper Sequence
8 Segment Has Data Element Errors
I4 Implementation "Not Used" Segment Present
I6 Implementation Dependent Segment Missing
I7 Implementation Loop Occurs Under Minimum Times
I8 Implementation Segment Below Minimum Use
I9 Implementation Dependent "Not Used" Segment

IK403
REQUIRED IK403 621 Implementation Data Element Syntax Error Code M 1 ID 1/3
Code indicating the implementation error found after syntax edits of a data element
CODE DEFINITION
1 Required Data Element Missing
2 Conditional Required Data Element Missing
3 Too Many Data Elements
4 Data Element Too Short
5 Data Element Too Long
6 Invalid Character In Data Element
7 Invalid Code Value
8 Invalid Date
9 Invalid Time
10 Exclusion Condition Violated
12 Too Many Repetitions
13 Too Many Components
I10 Implementation "Not Used" Data Element Present
I11 Implementation Too Few Repetitions
I12 Implementation Pattern Match Failure
I13 Implementation Dependent "Not Used" Data Element Present
I6 Code Value Not Used in Implementation
I9 Implementation Dependent Data Element Missing

FORMAL INTERPRETATION

Your interpretation and examples concerning the use of three words, should, shall and must, is correct. Usage of the words must, shall and should are as follows. Must and shall convey an absolute requirement. Should conveys a recommendation that can be ignored in circumstances that may be restricted. Your example demonstrates the incorrect usage of the data element separator.

Your second question, concerning reporting the errors in the example would be to use code 8 – "Segment has data element errors" in the IK304 and code value 3 – "Too many data elements" in the IK403. If the example referenced a composite data element, the appropriate value for IK403 is 13 – "Too many components".

FURTHER DISCUSSION

During the creation and discussion of this RFI, X12C subcommittee members agreed more descriptive codes would be helpful in reporting the error within the example. Additional codes will be added to a future version of the 999 transaction set.