X12.6 APPLICATION CONTROL STRUCTURE

1 Purpose and Scope

This standard defines the structure of business transactions for computer-to-computer interchange. This structure is expressed using a symbolic representation of X12 data independent of the physical representation (e.g., character set encoding). Transformations that affect the physical representation of the data are outside the scope of this standard. The symbolic representation is expressed in terms of both the design and use of X12 structures. This includes the control segments used to bound loops of data segments, transaction sets, and groups of related transaction sets, and the special control segments used to bound transaction sets and groups of transaction sets for security purposes.

This standard does not define any specific transaction set. Data segments are defined in a segment directory; data elements are defined in a data element dictionary; composite data structures are defined in a composite data structure directory; control segments and the binary segment are defined in this standard and fully described in a segment directory.

2 Referenced and Related Standards

This standard is used with the ASC X12 series of standards on electronic data interchange.

3 Definitions and Concepts

3.1 Basic Structure

The X12 standards define commonly used business transactions in a formal, structured manner called transaction sets. A transaction set is composed of a transaction set header control segment, one or more data segments, and a transaction set trailer control segment. Optionally the transaction set may also include a transaction security header control segment and transaction security trailer control segment when it is desirable to send the data within the transaction envelope in an authenticated or encrypted form, or both.

Each segment is composed of: a unique segment ID; one or more logically related simple data elements or composite data structures, or both, each preceded by a data element separator; and a segment terminator. Composite data structures are composed of one or more logically related component data elements, each, except the last, followed by a component element separator. The data segment directory entry referenced by the data segment identifier defines the sequence of simple data elements and composite data structures in the segment and any interdependencies that may exist. The composite data structure directory entry referenced by the composite data structure number defines the sequence of component data elements in the composite data structure.

A data element in the transaction set header identifies the type of transaction set. A functional group contains one or more related transaction sets preceded by a functional group header control segment and terminated by a functional group trailer control segment. Optionally, the functional group may also include a functional security header control segment and functional security trailer control segment when it is desirable to send the data within the functional envelope in an authenticated or encrypted form, or both.

3.2 Syntax Notation

The syntax is defined in a Backus-Naur Form (BNF), which is described in 3.2.1 through 3.2.8. Each definition is accompanied by explanatory text. In some cases actual use (the actual data stream) of an X12 structure may not be the same as the definition. These instances are caused by the optional nature of some structures. Where such structures occur in this document, two representations of the BNF are presented, labeled “definition” and “use” respectively.

3.2.1 Syntactic Entities

Lowercase strings that are enclosed in angle brackets denote syntactic entities. The underscore is included as a valid character for this representation. For example:

<transaction_set>

3.2.2 Defined Constructs

The defined construct is on the left side of a statement and is separated from the defining right side by two colons and an equal sign (::=). For example:

<one_construct> ::= <another_construct>

3.2.3 Other Inclusions

In the statements, spaces between syntactic entities are not significant and may be included for clarity. Meaningful spaces may be included in the statements by enclosing them in quotation marks. Other symbols may be included in quotation marks for clarity. The right side of the statement may be continued onto lower lines and be divided between syntactical entities. Ellipses (...) are used to represent missing items when a logical progression has been established.

3.2.4 Alternative Definitions

When alternative definitions exist, they may be shown separated by a vertical bar (|). For example:

<letter_or_digit> ::= <uppercase_letter> | <digit>

3.2.5 Predefined Labels

Certain character strings (for example, "ST") are predefined labels and are always to be used as they are given. Punctuation symbols appear as they would in the transfer of data; for example, the decimal point in "1.2" shall be specified as

<digit> . <digit>

3.2.6 Optional Items

Square brackets enclose optional items. For example, the optional negative sign is expressed as [-].

3.2.7 Multiple Occurrences

Braces enclose an item which may appear zero or more times. For example:

<unsigned_integer> ::= <digit> {<digit>}

3.2.8 Minimums/Maximums

The minimum and maximum number of characters that may be used for a syntactic entity is specified by the inclusion of two numbers in parentheses appended to the end of the syntactic construct as (minimum/maximum). This is used to represent the minimum and maximum lengths of a data element in the data element dictionary.

<id>(02/03) ::= <letter_or_digit> <letter_or_digit> [<letter_or_digit>]

3.3 Character Set

All data element values, except those of the binary data element, shall be constructed using the character set specified in 3.3.1 through 3.3.3. The character set of this standard is grouped according to common characteristics.

NOTE: The X12 standards are graphic-character oriented, so any common character encoding schemes may be used as long as a common mapping is available. No collating sequence is to be assumed in any definition used in the standard since no single character code is specified and no other means of specifying a sequence is provided.

3.3.1 Basic Character Set

The basic character set of this standard consists of uppercase letters, digits, special characters, and space.

(1) Uppercase letters from A to Z:

```
<uppercase_letter> ::= "A" | ... | "Z"
```

(2) Digits from 0 to 9:

```
<digit> ::= "0" | ... | "9"
```

(3) Special characters:

```
<special_char> ::= "!" | """ | "&" | "'" | "(" | ")" | "*" | "+" | "," |
      "-" | "." | "/" | ":" | ";" | "?" | "="
```

NOTE: Special characters are removed from this category when used as delimiters.

(4) The space character:

```
<space> ::= " "
```

3.3.2 Extended Character Set

An extended character set may be used by agreement between communicating parties and includes the lowercase letters, other special characters, national characters and select language characters.

(1) Lowercase letters from a to z:

```
<lowercase_letter> ::= "a" | ... | "z"
```

(2) Other special characters:

```
<other_special_char> ::= "%" | "@" | "[" | "]" |"_" | "{" | "}" |
      "\" | "|" | "<" | ">" | "~" | "^" | "`"
```

NOTE: Special characters are removed from this category when used as delimiters.

(3) National characters:

```
<national_char> ::= "#" | "$"
```

(4) Select language characters:

```
<select_language_char> ::= “À”|”Á”|”Â”|”Ä”|”à”|”á”|”â”|”ä”|
                           “È”|”É”|”Ê”|”è”|”é”|”ê”|”ë”|
                           ”Ì”|”Í”|”Î”|”ì”|”í”|”î”|”ï”|
                           “Ò”|”Ó”|”Ô”|”Ö”|”ò”|”ó”|”ô”|”ö”|
                           ”Ù”|”Ú”|”Û”|”Ü”|“ù”|”ú”|”û”|”ü”|
                           ”Ç”|”ç”|”Ñ”|”ñ”|”¿”|”¡”
```

Note: See ISO document 8859-1 (Latin-1 Alphabet) for an example of an encoding of select language characters.

3.3.3 Empty Character Set

An empty character set is the absence of data.

```
<Empty> ::= ""
```

3.4 Delimiters

The delimiters consist of three separators and a terminator. The delimiters are an integral part of the transferred data stream. Delimiters are specified in the interchange header and shall not be used in a data element value elsewhere in the interchange with the exception of their possible appearance in the binary data element. The separators and terminator are:

```
<tr> ::= segment terminator

<gs> ::= data element separator

<us> ::= component element separator

<rs> ::= repetition separator
```

3.5 Data Element

The data element is the smallest named unit of information in the standard. Data elements are identified as either simple or component. A data element that occurs as an ordinally positioned member of a composite data structure is identified as a component data element. A data element that occurs in a segment outside the defined boundaries of a composite data structure is identified as a simple data element. The distinction between simple and component data elements is strictly a matter of context since a data element can be used in either capacity. The following definitions apply equally to each class of data element.

```
<simple_data_element> ::= <data_element>

<component_data_element> ::= <data_element>

<data_element> ::= <numeric> | <decimal_number> | <id> | <string> | <date> |
      <time> | <binary> | <fixed_length_string>
```

3.5.1 Data Element Types

The data element types and the length characteristics of each of these types are described in 3.5.1.1 through 3.5.2.3.

3.5.1.1 Numeric

A numeric is represented by one or more digits with an optional leading sign representing a value in the normal base of 10.

```
<numeric> ::= [-] <unsigned_integer>

<unsigned_integer> ::= <digit> {<digit>}
```

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 data element dictionary defines the number of implied decimal positions. The representation 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 n is 0, it need not appear in the specification; N is

equivalent to N0. For negative values, the leading minus sign (-) is used. Absence of a sign indicates a positive value. The plus sign (+) shall not be transmitted. 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 sign.

FOR EXAMPLE:

Value is -123.4
Numeric type is N2 where the "2" indicates an implied decimal placement two positions from the right
The data stream value is -12340
The length is 5 (note padded zero)

3.5.1.2 Decimal Number

A decimal data element contains an explicit decimal point and is used for numeric values that have a varying number of decimal positions. The representation for this data element type is R. The decimal point always appears in the character stream if the decimal point is at any place other than the right end. If the value is an integer (decimal point at the right end) the decimal point should be omitted. For negative values, the leading minus sign (-) is used. Absence of a sign indicates a positive value. The plus sign (+) shall not be transmitted. Leading zeros should be suppressed unless necessary to satisfy a minimum length requirement. Trailing zeros following the decimal point should be suppressed unless necessary to indicate precision. A base 10 exponential may be appended to the end of a decimal number. The absolute value of a decimal number with a base 10 exponential is equal to the value of the <unsigned_decimal_number> multiplied by 10 raised to the power of <exponent>. The use of triad separators (for example, the commas in "1,000,000") is expressly prohibited. The length of a decimal type data element does not include the optional minus signs, decimal point, or trailing exponent indicator "E".

<decimal_number> ::= [-] <unsigned_decimal_number> [<base_10_exponential>]

<base_10_exponential> ::= E <exponent>

<exponent> ::= [-] <unsigned_integer>

<unsigned_decimal_number> ::= <unsigned_integer> | .<unsigned_integer> |
<unsigned_integer> . {<digit>}

EXAMPLE A:

Value is -123.45
Decimal type symbol is R
The data stream value is -123.45
The length is 5

EXAMPLE B:

Value is 12345
Decimal type symbol is R
The data stream value is 12345
The length is 5

EXAMPLE C:

Value is .1250
Decimal type symbol is R
The data stream value is 1250E-4

The length is 5

3.5.1.3 Identifier

An identifier data element always contains a unique value from a single, predefined list of values that is maintained by ASC X12 or some other body recognized by ASC X12 and identified by a reference in Appendix A of X12.3 Data Element Dictionary. Trailing spaces should be suppressed. The representation for this data element type is ID.

```
<id> ::= <letter_or_digit> {<letter_or_digit>} {<space>}
<letter_or_digit> ::= <uppercase_letter> | <digit>
```

3.5.1.4 String

A string data element is a sequence of any characters from the basic or extended character sets and contains at least one non-space character. The significant characters shall be left justified. Leading spaces, when they occur, are presumed to be significant characters. In the actual data stream trailing spaces should be suppressed. The representation for this data element type is AN.

```
<string> ::= {<non_space_char> | <space>} <non_space_char> {<non_space_char> |
      <space>}

<non_space_char> ::= <uppercase_letter> | <digit> | <special_char> |
      <lowercase_letter> | <other_special_char> | <national_char> |
      <select_language_character>
```

3.5.1.5 Date

A date data element is used to express the standard date in either YYMMDD or CCYYMMDD format in which CC is the first two digits of the calendar year, YY is the last two digits of the calendar year, MM is the month (01 to 12), and DD is the day in the month (01 to 31). The representation for this data element type is DT.

```
<date> ::= <year> <month> <day> | <hundred_year> <year> <month> <day>

<hundred_year> ::= <digit> <digit>

<year> ::= <digit> <digit>

<month> ::= "01" | "02" | ... | "12"

<day> ::= "01" | "02" | ... | "31"
```

3.5.1.6 Time

A time data element is used to express the time in HHMMSSd..d 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 d..d is decimal seconds. Seconds and decimal seconds are optional. Trailing zeros in decimal seconds should be suppressed unless necessary to satisfy a minimum length requirement or unless necessary to indicate precision. The representation for this data element type is TM.

```
<time> ::= <hour> <minute> [<seconds>]

<hour> ::= "00" | "01" | "02" | ... | "23"

<minute> ::= "00" | "01" | "02" | ... | "59"

<seconds> ::= <integer_seconds> [<decimal_seconds>]

<integer_seconds> ::= "00" | ... | "59"

<decimal_seconds> ::= <digit> {<digit>}
```

3.5.1.7 Binary

The binary data element is any sequence of octets ranging in value from binary 00000000 to binary 11111111. 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 and Security header segments (see Section 3.11). The representation for this data element type is B.

<binary> ::= <octet> {<octet>}

<octet> ::= "00000000_2" | ... | "11111111_2"

3.5.2 Data Element Dictionary

The data elements in the dictionary are identified through a reference number. For each data element, the dictionary specifies the name, description, type, minimum length, and maximum length. For ID data elements, the dictionary lists all code values and their descriptions or a reference where the valid code list can be obtained.

3.5.2.1 Data Element Reference Number

Data elements that appear in X12.3 Data Element Dictionary are assigned a unique reference identifier from one to four characters.

<data_element_reference_number>(01/04) ::= <unsigned_integer>

3.5.2.2 Data Element Type

The following types of data elements, as defined in 3.5.1, appear in the dictionary.

Type	Symbol
Numeric	Nn (n indicates decimal positions)
Decimal Number	R
Identifier	ID
String	AN
Date	DT
Time	TM
Binary	B

3.5.2.3 Data Element Length

Each data element is assigned a minimum and maximum length. The length of the data element value is the number of character positions used except as noted for numeric, decimal, and binary elements.

3.5.2.4 Data Element Name

Data element names shall be unique throughout the X12.3 Data Element Dictionary. A simple data element name shall not duplicate a composite data structure name in the X12.3 Data Element Dictionary or a segment name appearing in the X12.22 Segment Directory.

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>
```

3.6.1 Composite Data Structure Reference Identifier

Each composite data structure has a unique four-character reference identifier. The reference identifier serves as a means of locating the composite data structure in the composite data structure directory. The first character is an "S" when the composite data structure is used in a control segment and is a "C" when the composite data structure is used in a data segment. The remaining character positions will contain digits.

```
<composite_data_structure_identifier> ::=    <composite_identifier_prefix>
      <digit> <digit> <digit>

<composite_identifier_prefix> ::= "S" | "C"
```

3.6.2 Composite Data Structure Name

Composite data structure names shall be unique throughout the X12.3 Data Element Dictionary. A composite data structure name shall not duplicate a simple data element name appearing in the X12.3 Data Element Dictionary or a segment name appearing in the X12.22 Segment Directory.

3.6.3 Component Data Elements in a Composite Data Structure

In defining a composite data structure, each component data element within the composite data structure is further characterized by a condition designator and a data element reference number.

3.6.2.1 Component Data Element Condition Designator

Component data elements shall have a designator that defines their requirement in a composite data structure. These condition designators are represented by a single-character code on the composite data structure diagrams. Refer to Section 3.7.3.2 for the definition of condition designators.

3.6.2.2 Absence of Data

Any component data element that is indicated as mandatory must be included in the composite data structure if the composite data structure is used. Optional component data elements may be omitted if they are not needed. The absence of these component data elements is noted by the occurrence of the component element separators which would have followed them, or, in the case of the last component data element in the composite data structure, by the occurrence of the data element separator or segment terminator. If the optional component data elements being omitted occur at the end of a composite data structure, the structure must be terminated by the occurrence of the next data element separator or the segment terminator following the last used data element value or 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>
```

3.7.1 Data Segment Identifier

Each data segment has a unique two- or three-character identifier. This identifier serves as a label for the data segment.

```
<seg_id> ::= <letter_or_digit> <letter_or_digit> [<letter_or_digit>]
```

3.7.2 Data Segment Name

Segment names shall be unique throughout the X12.22 Segment Directory. A segment name shall not duplicate a simple data element name or a composite data structure name appearing in the X12.3 Data Element Dictionary.

3.7.3 Data Elements in a Segment

In defining a segment, each simple data element or composite data structure within the data segment is further characterized by a reference designator and a data element reference number or composite data structure reference identifier. Simple data elements and composite data structures have additional attributes. They shall have a reference designator, a condition designator and a repetition designator. They may also have a semantic note designator.

3.7.3.1 Reference Designator

Each simple data element or composite data structure in a segment is provided a structured code that indicates the segment in which it is used and the sequential position within the segment. The code is composed of the segment identifier followed by a two-digit number that defines the position of the simple data element or composite data structure in that segment. For purposes of creating reference designators, the composite data structure is viewed as the hierarchical equal of the simple data element.

Each component data element in a composite data structure is identified by a suffix appended to the reference designator for the composite data structure of which it is a member. This suffix is a two-digit number, prefixed with a hyphen, that defines the position of the component data element in the composite data structure. For example, the third simple element of the BEG segment would be identified as BEG03 because the position count does not include the segment identifier, which is a label. If the fourth position in the BEG segment were occupied by a composite data structure that contained three component data elements, the reference designator for the second component data element would be BEG04-02.

definition:

<ref_desig> ::= <seg_id> <digit> <digit> ["-"<digit> <digit>]

use:

In use the reference designator does not appear.

3.7.3.2 Condition Designator

Data segment unit or component data element conditions are of three types: mandatory, optional, and relational, and define the circumstances under which a simple data element, composite data structure, or component data element may be required to be present or absent in a particular segment or composite data structure.

3.7.3.2.1 Mandatory Condition

The designation of mandatory is absolute in the sense that there is no dependency on other data segment units or component data elements and may apply to either simple data elements, composite data structures, or component data elements. Mandatory conditions are specified by condition code "M".

Condition	*Requirement*
(M) Mandatory	The designated simple data element or composite data structure, whether allowed to repeat or not, must be present in the segment, or the designated component data element must be present in the composite data structure (presence means a data element, composite data structure, or component data element shall not be empty; see Section 3.3.3). If the designation applies to a composite data structure, then at least one value of a component data element in that composite data structure shall be included in the data segment. If the designation applies to a repeating simple data element or a repeating composite data structure then at least one data element value shall be present.

3.7.3.2.2 Optional Condition

The designation of optional means that there is no syntactic requirement for a simple data element or composite data structure to be present in the segment or for a component data element to be present in the composite data structure. Optional conditions are specified by condition code "O".

Condition	*Requirement*
(O) Optional	The presence of a value for a simple data element or the presence of a value for any of the component data elements of a composite data structure is at the option of the sender.

3.7.3.2.3 Relational Conditions

Relational conditions may exist among two or more data segment units within the same data segment, or among two or more component data elements within the same composite data structure, based on the presence or absence of one of those data segment units or component data elements (presence means a data element must not be empty). Relational conditions are specified by a condition code and the

identity of the subject elements or structures. Relational conditions may not exist between a component data element and anything outside its parent composite data structure.

```
<relational_cond> ::= <relation_code> <subj_elem> <subj_elem>  {<subj_elem>}

<relation_code> ::= "P" | "R" | "E" | "C" | "L"

<subj_elem> ::= <segment_unit_position_in_segment> |
      <component_element_position_in_composite>

<segment_unit_position_in_segment> ::= <digit> <digit>

<component_element_position_in_composite> ::=
      <segment_unit_position_in_segment> "-" <digit> <digit>
```

A data element may be subject to more than one relational condition.

The definitions for each of the <relation_code> values are:

Code	***Requirement***
(P) Paired or Multiple	If any <subj_elem> specified in the <relational_cond> is present, then all of the <subj_elem> specified must be present.
(R) Required	At least one of the <subj_elem> specified in the <relational_cond> must be present.
(E) Exclusion	Not more than one of the <subj_elem> specified in the <relational_cond> may be present.
(C) Conditional	If the first <subj_elem> specified in the <relational_cond> is present, then all other <subj_elem> must be present. However, any or all of the <subj_elem> NOT specified as the FIRST <subj_elem> in the <relational_cond> may appear when the first <subj_elem> is not present. The order of the <subj_elem> in the <relational_cond> does not have to be the same as the order of the data elements in the data segment.
(L) List Conditional	If the first <subj_elem> specified in the <relational_cond> is present, then at least one of the remaining <subj_elem> must be present. However, any or all of the <subj_elem> NOT specified as the FIRST <subj_elem> in the <relational_cond> may appear when the first <subj_elem> is not present. The order of the <subj_elem> in the <relational_cond> does not have to be the same as the order of the data elements in the data segment.

3.7.3.3 Repetition Designator

The repetition designator may be one, indicating the simple data element or composite data structure may occur no more than once, or contain a specific value greater than one, indicating the maximum number of occurrences of the simple data element or composite data structure.

3.7.3.4 Semantic Notes

A semantic note provides important additional information regarding the intended use of a standard or portion thereof. These usage notes, while not part of the formal syntax, must be followed in order to be in compliance with the standard. Semantic notes, therefore, are considered to be part of the relevant standard and are appropriately designated to distinguish them from syntax notes (which are part of the standard) and comments (which are not part of the standard).

Semantic notes for transaction sets provide information with respect to the intended use of a segment within the context of a particular transaction set. Semantic notes for segments provide information regarding the intended meaning of a designated data element, particularly a generic type, in the context of its use within a specified data segment. Semantic notes may also define relational conditions among

data elements in a segment based on the presence of a specific value (or one of a set of values) in one of the data elements.

3.7.4 Absence of Data

Absence of data is represented by the value <empty>. Any value other than <empty> is an indication that data are present. 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; or, if they do not occur at the end of the segment, the simple data element values or composite data structure values may be omitted, and their absence is indicated by the occurrence of their preceding data element separators, in order to maintain the element's or structure's position as defined in the data segment.

3.8 Transaction Set

The transaction set is a semantically meaningful unit of information exchanged between trading partners. The transaction set shall consist of a transaction set header segment, optionally one or more transaction security header segments, optionally one or more transaction assurance header segments, one or more data segments and loop control segments (if bounded loops exist) in a specified order, one transaction security value segment for each assurance header present, one transaction security trailer segment for each security header segment present, and a transaction set trailer segment.

```
<transaction_set>::= <trans_set_header> <trans_block> <trans_set_trailer>

<trans_block> ::= <secured_trans_block> | <unsecured_trans_block>
<secured_trans_block> ::= <trans_security_header> <trans_block>
      <trans_security_trailer>

<unsecured_trans_block> ::= <assured_trans_block> | <trans_body>

<assured_trans_block> ::= <trans_assurance_header> <unsecured_trans_block>
      <security_value>

<trans_body> ::= <data_segment_group> {<data_segment_group>}
```

3.8.1 Transaction Set Header and Trailer

The transaction set header and trailer segments are constructed as follows:

```
<trans_set_header>  ::= ST <gs> <trans_set_id>
                           <gs> <trans_set_control_number>
                           {<gs> <ic_id_string>} <tr>

<trans_set_trailer> ::= SE <gs> <number_of_included_segments>
                           <gs> <trans_set_control_number>
                           <tr>

<trans_set_id> ::= <id>

<trans_set_control_number> ::= <string>

<ic_id_string>  ::= <string>

<number_of_included_segments> ::= <unsigned_integer>
```

The transaction set identifier, <trans_set_id>, uniquely identifies the transaction set. This identifier is the first data element of the transaction set header segment.

The value for the transaction set control number, <trans_set_control_number>, in the header and trailer control segments must be identical for any given transaction. The value for the number of included

segments, <number_of_included_segments>, is the total number of segments in the transaction set including the ST and SE segments.

The implementation convention identification string, <ic_id_string>, identifies which implementation convention is in use for the transaction set. Its value may be a reference to a repository, or is available for trading partner definition. This identifier is the optional third data element of the transaction set header segment.

In order to provide sufficient discrimination for the acknowledgment process to operate reliably and to ensure that audit trails are unambiguous, the Transaction Set Control Number (ST02, SE02) shall be unique within a given functional group envelope (GS/GE).

3.8.2 Transaction Security Header and Trailer

The X12.58 Security Structures Standard defines the data formats for authentication, encryption, and assurances in order to provide integrity, confidentiality, and verification and non-repudiation of origin for two levels of exchange of Electronic Data Interchange (EDI)-formatted data defined by Accredited Standards Committee (ASC) X12. These security services can be applied at either the functional group level or the transaction set level or both. Please refer to the X12.58 standard for the construction and usage of the security related segments.

3.8.3 Data Segment Groups

The data segments in a transaction set may be repeated as individual data segments or as unbounded or bounded loops.

```
<data_segment_group> ::= <data_segment> | <data_segment_repeat> |
      <data_segment_loop> | <bounded_loop>
```

3.8.3.1 Repeated Occurrences of Single Data Segments

When a single data segment is allowed to be repeated, it may have a specified maximum number of occurrences defined at each specified position within a given transaction set standard. Alternatively, a segment may be allowed to repeat an unlimited number of times. The notation for an unlimited number of occurrences is ">1". The ">1" construct is used only if a specific maximum number of occurrences cannot be determined or is unknown. By definition a data segment repeat consists of at least two occurrences. In use, in the actual data stream, it may appear fewer than two times.
definition:

```
<data_segment_repeat> ::= <data_segment> <data_segment> {<data_segment>}
```

use:

```
<data_segment_repeat> ::= {<data_segment>}
```

3.8.3.2 Loops of Data Segments

By definition, loops are groups of two or more semantically related segments. In use, in the actual data stream, a loop may appear as only the loop beginning segment. Data segment loops may be unbounded or bounded.

3.8.3.2.1 Unbounded Loops

In order to establish the start of the loop, the first data segment in the loop shall appear once and only once in each occurrence. The beginning segment of a loop shall not appear elsewhere in the loop. Loops may have a specified maximum number of occurrences. Alternatively, the loop may be specified as having an unlimited number of occurrences. The notation for an unlimited number of repeats is ">1". There is a specified sequence of segments 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 an 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.

definition:

<data_segment_loop> ::= <loop_beg_seg> <loop_body> {<loop_body>}

<loop_beg_seg> ::= <data_segment>

<loop_body> ::= <data_segment> | <data_segment_group>

use:

<data_segment_loop> ::= <loop_beg_seg> {<loop_body>}

If unbounded loops are nested within loops, the inner loop shall not start at the same ordinal position as any outer loop. The inner loop shall not start with the same segment ID as any outer loop, nor may the nested loop contain a segment that is also the beginning segment of any outer loop in the same nesting structure. The inner loop must end before or on the same segment as its immediate outer loop.

3.8.3.2.2 Bounded Loops

The characteristics of unbounded loops described in 3.8.3.2.1 also apply to bounded loops except that bounded loops have no restriction with respect to the beginning segment ID. In addition, bounded loops require a loop start (LS) segment to appear before the first occurrence and a loop end (LE) segment to appear after the last occurrence of the loop. If the loop does not occur within an actual transmission, the LS and LE segments shall be suppressed. The requirement designator on the LS and LE segments must match the requirement designator of the beginning segment of the loop. A bounded loop may contain only one loop structure (a single definition of the included segments) at the level bracketed by the LS and LE segments. Subordinate loops are permissible.

<bounded_loop> ::= <loop_header> <data_segment_loop> {<data_segment_loop>}
<loop_trailer>

<loop_header> ::= LS <gs> <loop_id> <tr>

<loop_trailer> ::= LE <gs> <loop_id> <tr>

<loop_id>(01/04) ::= <uppercase_letter_or_digit> {<uppercase_letter_or_digit>}

If bounded loops are nested within loops, the inner loop shall not start at the same ordinal position as any outer loop. The inner bounded loop must end before the immediate outer loop.

Each bounded loop within a transaction set shall have a uniquely defined <loop_id> value of one to four uppercase letters or numeric digits. The corresponding LS and LE segments shall contain the same unique <loop_id> value. The specific <loop_id> values shall be defined in the transaction set standard. In use, the actual LS and LE segments shall contain the defined <loop_id> values. In usage, the <loop_id> will in some transaction sets be necessary to convey segment position or loop hierarchy, or both, within the transaction set.

3.8.4 Data Segments in a Transaction Set

When data segments and control segments are combined to form a transaction set, three characteristics shall be applied to each segment in that usage: a requirement designator, a position in the transaction set definition, and a maximum occurrence.

3.8.4.1 Data Segment Requirement Designator

A data segment shall have one of the following requirement designators indicating its appearance in the data stream of a transmission. These requirement designators are represented by a single character code.

Designator	*Requirement*
(M) Mandatory	This data segment shall be included in the transaction set. (Note that a data segment may be mandatory in a loop of data segments, but the loop itself is optional if the beginning segment of the loop is designated as optional.)
(O) Optional	The presence of this data segment is at the option of the sending party.

3.8.4.2 Data Segment Position

The ordinal positions of the segments in a transaction set definition are explicitly specified for that transaction set. This positioning shall be maintained during transmission.

3.8.4.3 Data Segment Occurrence

A data segment may have a maximum occurrence of one, a finite number greater than one, or an unlimited number. (See also Section 3.8.3.1)

3.8.4.4 Data Segment Order

A transaction set's segment order shall be defined such that sequential processing of any legitimate instance of the transaction set will result in positive identification of each segment in terms of its ordinal position in the standard. Positive identification shall accordingly be made on the basis of the segment identifiers alone, with the exception of the LS and LE segments, where the identification is made using the segment identifier in conjunction with the <loop_id>. Positive identification shall not depend on a segment's requirement designator or maximum occurrences.

3.9 Functional Group

A functional group shall consist of a functional group header segment, optionally one or more functional security header segments, optionally one or more functional assurance header segments, one or similar transaction sets, one functional security value segment for each assurance header present, one functional security trailer segment for each security header segment present, and a functional group trailer segment.The functional identifier (<function_id>) defines the collection of transaction sets that may be included within the functional group.

```
<functional_group>::= <function_header> <function_block> <function_trailer>

<function_block> ::= <secured_function_block> | <unsecured_function_block>

<secured_function_block> ::= <function_security_header> <function_block>
      <function_security_trailer>

<unsecured_function_block> ::= <assured_function_block> | <function_body>

<assured_function_block> ::= <function_assurance_header>
      <unsecured_function_block> <security_value>

<function_body> ::= <transaction_set> {<transaction_set>}
```

3.9.1 Functional Group Header and Trailer

The functional group header and trailer segments are constructed as follows:

```
<function_header>  ::= GS <gs> <function_id>
                       <gs> <app_sender_id>
                       <gs> <app_receiver_id>
                       <gs> <fg_date>
                       <gs> <fg_time>
                       <gs> <function_control_number>
```

```
                        <gs> <standard>
                        <gs> <version> <tr>

<function_id> ::= <id>

<app_receiver_id> ::= <string>

<app_sender_id> ::= <string>

<fg_date> ::= <date>

<fg_time> ::= <time>

<function_control_number> ::= <numeric>

<standard> ::= <id>

<version> ::= <string>

<function_trailer> ::= GE <gs> <no_included_trans_sets>
                        <gs> <function_control_number> <tr>

<no_included_trans_sets> ::= <numeric>
```

The functional group control number, <function_control_number>, in the header and trailer control segments shall be the same for any given group. The number of included transaction sets, <no_included_trans_sets>, is the total number of transaction sets in the group.

In order to provide sufficient discrimination for the acknowledgment process to operate reliably and to ensure that audit trails are unambiguous, the combination of Functional ID Code (GS01), Application Sender's ID (GS02), Application Receiver's ID (GS03), and Functional Group Control Numbers (GS06, GE02) shall by themselves be unique within a reasonably extended time frame whose boundaries shall be defined by trading partner agreement. Because at some point it may be necessary to reuse a sequence of control numbers, the Functional Group Date and Time may serve as an additional discriminant only to differentiate functional group identity over the longest possible time frame.

3.9.2 Functional Security Header and Trailer

The X12.58 Security Structures Standard defines the data formats for authentication, encryption, and assurances in order to provide integrity, confidentiality, and verification and non-repudiation of origin for two levels of exchange of Electronic Data Interchange (EDI) formatted data defined by Accredited Standards Committee (ASC) X12. These security services can be applied at either the functional group level or the transaction set level or both. Please refer to the X12.58 standard for the construction and usage of the security-related segments.

3.10 Control Segment

A control segment has the same structure as a data segment but is used for transferring control information rather than application information. Control segments referred to in this standard are fully described in X12.22 Segment Directory.

3.10.1 Loop Control Segments

Loop control segments are used only to delineate bounded loops. Delineation of the loop shall consist of the loop header (LS segment) and the loop trailer (LE segment). The loop header defines the start of a structure that must contain one or more iterations of a loop of data segments and provides the loop

identifier for this loop. The loop trailer defines the end of the structure. The LS segment appears only before the first occurrence of the loop, and the LE segment appears only after the last occurrence of the loop.

3.10.2 Transaction Set Control Segments

The transaction set is delineated by the transaction set header (ST segment) and the transaction set trailer (SE segment). The transaction set header starts and identifies the transaction set. The transaction set trailer defines the end of the transaction set and provides a count of the data segments, which includes the ST and SE segments.

3.10.3 Functional Group Control Segments

The functional group is delineated by the functional group header (GS segment) and the functional group trailer (GE segment). The functional group header starts and identifies one or more related transaction sets and provides a control number and application identification information. The functional group trailer defines the end of the functional group of related transaction sets and provides a count of contained transaction sets.

3.10.4 Security Control Segments

The security control segments are used to provide data integrity, confidentiality, verification of origin and non-repudiation of origin by delineating information for authentication, encryption, and compression. The X12.58 Security Structures Standard defines the data formats and business usage for these security services. Please refer to this standard for the construction and usage of the security-related segments.

3.10.5 Relations among Control Segments

The control segments of this standard shall have a nested relationship as is shown and annotated in this subsection. The letters preceding the control segment name are the segment identifier for that control segment. The indentation of segment identifiers shown below indicates the subordination among control segments. Security segments are optional.

GS Functional Group Header starts a group of related transaction sets.

Functional Group Security and Assurance header segments, if used, shall be inserted here. Please refer to the X12.58 Security Structures Standard for the construction and usage of the security-related segments.

ST Transaction Set Header, starts a transaction set.

Transaction Set Security and Assurance header segments, if used, shall be inserted here. Please refer to the X12.58 Security Structures Standard for the construction and usage of the security-related segments.

LS Loop Header, starts a bounded loop of data segments but is not part of the loop.

LS Loop Header, starts an inner, nested, bounded loop.

LE Loop Trailer, ends an inner, nested, bounded loop.

LE Loop Trailer, ends a bounded loop of data segments but is not part of the loop.

Transaction Set Security and Assurance trailer segments, if used, shall be inserted here. Please refer to the X12.58 Security Structures Standard for the construction and usage of the security-related segments.

SE Transaction Set Trailer, ends a transaction set.

Functional Group Security and Assurance trailer segments, if used, shall be inserted here. Please refer to the X12.58 Security Structures Standard for the construction and usage of the security-related segments.

GE Functional Group Trailer, ends a group of related transaction sets.

More than one ST/SE pair, each representing a transaction set, may be used within one functional group. Also more than one LS/LE pair, each representing a bounded loop, may be used within one transaction set.

3.11 Binary Segment

A binary segment, BIN or BDS, has the same structure as a data segment but is used for transferring binary data with an accompanying length parameter. The BDS segment also contains a filter parameter. The filter parameter specifies the encoding algorithm, if any. Bit patterns normally reserved for other functions may appear in the binary data element. These bit patterns could be misinterpreted. The length of the binary data element is provided in the preceding data element in order to locate the end of the binary data and prevent such misinterpretation. The binary segment may not occur outside the boundaries of a transaction set.

```
<binary_seg> ::= BIN <gs> <number_of_octets> <gs> <binary> <tr> |
                 BDS <gs> <filter_id> <gs> <number_of_octets> <gs> <binary> <tr>

<number_of_octets> ::= <unsigned_integer>

<filter_id> ::= <id>
```

The data element that references the number of octets provides a count of all octets contained in the binary data element. This count does not include the preceding data element separator or trailing data segment terminator. The count is provided to enable finding the data segment terminator.

X12.5 INTERCHANGE CONTROL STRUCTURES

1 Purpose and Scope

The purpose of this standard is to define the control structures for the electronic interchange of one or more encoded business transactions including the EDI (electronic data interchange) encoded transactions of Accredited Standards Committee (ASC) X12. This standard provides the interchange envelope of a header and trailer for the electronic interchange through a data transmission, a structure to acknowledge the receipt and processing of this envelope, and optional, interchange-level service request structures.

These service request segments are treated as a logical extension of the basic interchange control header. An interchange delivery notice segment provides a method to report on interchange services requested and indicates that status of delivery and retrieval of an interchange.

2 Referenced American National Standards

This standard is intended to be used with transaction set standards developed by ASC X12, as well as X12.6 Application Control Structure.

3 Description of Use

There are six areas of the interchange which appear in the following order.

- Interchange control header
- Interchange service requests
- Interchange delivery notice
- Interchange acknowledgment
- Functional groups
- Interchange control trailer

The interchange control header and the interchange control trailer form the basic interchange. An interchange may contain additional control segments (e.g., service requests, delivery notices, and acknowledgments) and one or more functional groups.

3.1 Definitions and Concepts

The following definitions apply to this description of the interchange control structure. Italicized words indicate these terms are defined elsewhere in this section.

DELIVER: The action whereby the original interchange is made available to the *interchange receiver*. An interchange is considered "delivered" when it has been successfully conveyed to the *mailbox*, or, if a mailbox is not in use, to the *interchange receiver.*

INTERCHANGE RECEIVER: The interchange receiver is the entity that is the intended recipient of the interchange as specified by the *interchange sender* in the interchange header segment. The interchange receiver performs syntactical analysis of the interchange header, trailer, and acknowledgment segments and generates interchange acknowledgments to be returned to the *interchange sender*, if requested.

INTERCHANGE SENDER: The interchange sender is the entity that constructs the interchange header, interchange trailer, and interchange service requests.

INTERCHANGE SERVICE REQUEST: Segment used to specify additional services within the interchange.

MAILBOX: A mailbox is an entity capable of providing protected storage under the control of an *interchange receiver* for an interchange in transit between the *interchange sender* and *interchange receiver.*

RECEIVING SERVICE REQUEST HANDLER: The receiving service request handler is the entity that receives an interchange transferred by other service request handlers and delivers the interchange as specified by the *interchange sender*. The receiving service request handler also generates interchange delivery notices to be returned to the sending *service request handler.*

REFUSE: The action taken on an interchange *delivered* to a *mailbox* that cannot, or will not, be *retrieved.*

REJECT: The action taken on an interchange that cannot be *delivered* or *transferred* successfully.

RETRIEVE: The successful conveyance of an interchange from a *mailbox* to the *interchange receiver*.

SENDING SERVICE REQUEST HANDLER: The sending service request handler is the entity that acts upon interchange service requests as specified by the *interchange sender*, *transferring* the interchange to the intended *service request handler* of the *interchange receiver*. The sending service request handler also processes interchange delivery notices returned by other *service request handlers.*

SERVICE REQUEST HANDLER: A service request handler is an entity, intermediate to the *interchange sender* and *interchange receiver*, capable of acting upon one or more optional service requests or delivery notifications.

TRANSFER: The action whereby a *sending service request handler* sends an interchange to a *receiving service request handler.*

3.2 Basic Interchange Service

The basic interchange control structure is designed to satisfy the basic requirements for enveloping and routing electronic business data.

3.2.1 Basic Interchange Service Request

The Interchange Control Header (ISA) and Interchange Control Trailer (IEA) segments form the basic interchange service request. This is a basic delivery service request for one or more functional groups or interchange-related control segments, or a combination of functional groups and interchange-related control segments. The basic interchange service request performs the following functions.

- Defines the data element separator, component element separator, and segment terminator
- Identifies the sender and receiver
- Provides control information for the interchange
- Allows for authorization and security information

The basic interchange service is required for delivery of the interchange. The addressing information is located within the interchange header.

3.2.2 Basic Interchange Acknowledgment

An Interchange Acknowledgment segment (TA1) is used to report the receipt of the contents of one interchange control header and trailer envelope where that envelope surrounds one or more functional groups. This acknowledgment is transferred between the interchange receiver and sender as addressed

in the interchange header. The TA1 reports the results of the syntactical analysis of the interchange control header and trailer. There is no acknowledgment for the TA1, thereby preventing an endless loop of acknowledgments. The flow of the original interchange and the corresponding acknowledgment are diagrammed in Figure 1.

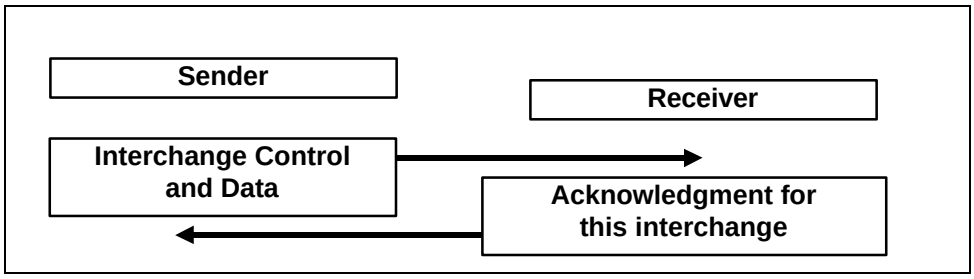

Figure 1: Flow of Original Interchange and Corresponding Acknowledgment

The interchange control number value in TA1 is the same as for the ISA for which the acknowledgment was prepared. This control number serves as a link between the interchange header and the acknowledgment of that header.

The TA1 reports the status of the interchange envelope only. It is not used to report the status of the functional groups and transaction sets in the interchange.

The preparer of the interchange header and trailer indicates the level of acknowledgment requested in the acknowledgment requested data element. If the value in this data element position indicates no acknowledgment, the interchange receiver shall not return that acknowledgment. A TA1 control segment shall not be generated for an interchange that does not contain any transaction sets, thus preventing an endless loop of TA1 or TA3 acknowledgments.

3.3 Optional Interchange Service Requests

The basic interchange service capability provided by the ISA can be expanded by optional interchange service requests and responses.

The optional interchange service requests accompanying a basic interchange request (ISA) provide the interchange sender with a means to request ancillary services that may be provided by service requests handlers.

The optional interchange service request segment and the service functions they are designed to support are:

ISB - Grade of Service Request and
ISE - Deferred Delivery Request.

It is the responsibility of the interchange sender to know whether or not their service request handler (SRH) supports the service request.

When the SRH receives a service request from the interchange sender or sending SRH, the SRH will, by default, remove the service request. To override the default, the SRH must establish an agreement with the interchange receiver or receiving SRH to retain the service request.

Although the segments for these services are separate segments, they immediately follow the basic interchange header and must be treated as a logical extension to the ISA. This means that the service request handlers must examine the fully extended header to decide if there are any optional services to perform.

3.4 Interchange Delivery Notice (TA3)

The Interchange Delivery Notice segment (TA3) is exchanged between service request handlers to inform the sending service request handlers of actions taken on the interchange by the receiving service request handler. The TA3 reports the delivery and the retrieval of the interchange. The TA3 also reports the unsuccessful delivery or retrieval of the interchange and identifies the error condition. Data extracted from the reported interchange for identification purposes and timestamps that identify the time an action has occurred are included. Other optional service request handler actions are also reported.

The flow of the original interchange and the corresponding TA3 segments are diagrammed in Figure 2.

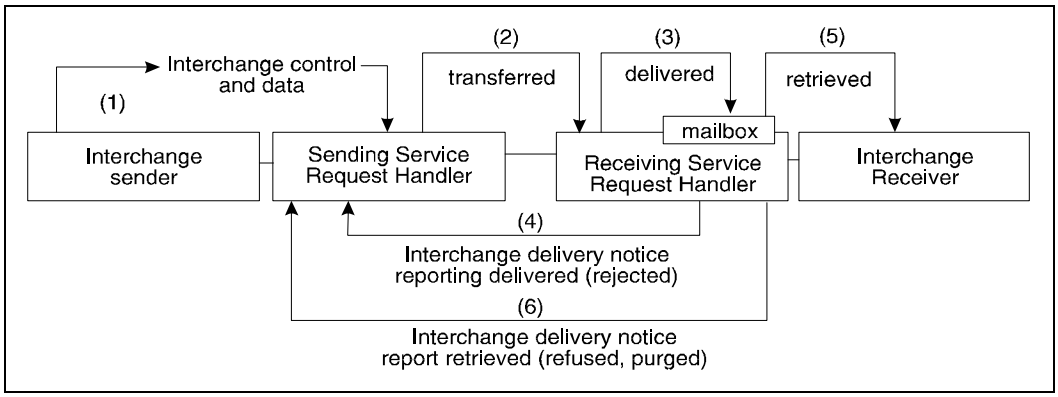

Figure 2: Flow of Original Interchange and Corresponding Interchange Delivery Notice (event sequence illustrated as numbered)

The use of TA3 is optional. Its usage should be based on mutual agreement. A TA3 control segment shall not be generated for an interchange that does not contain any transaction sets, thus preventing an endless loop of TA1 or TA3 acknowledgments.

TA3 segments are conveyed in an interchange addressed from the receiving service request handler to the sending service request handler.

3.5 Order of Control Segments

The interchange prepared by the interchange sender shall occur in the order as listed in Figure 3. (See explanations of field heading that follow figure 4.)

Seg. ID	Name	Req. Des.	Max Use
ISA	Interchange Control Header	M	1
ISB	Grade of Service	O	1
ISE	Deferred Delivery Request	O	1
TA1	Interchange Acknowledgment	O	>1
-	Functional Groups (see note)	O	99,999
IEA	Interchange Control Trailer	M	1

Figure 3: Order of the Interchange Segments for Interchange Sender

NOTE: In the diagram above, the functional group is not an interchange component of this standard but appears in this figure to establish positioning for the functional groups. Zero, one, or more than one interchange acknowledgments may appear in one interchange. Zero, one, or more than one functional groups may appear in one interchange. However, at least one interchange acknowledgment or functional group must appear in an interchange for an interchange sender.

The interchange prepared by the service request handler shall occur in the order as listed in Figure 4.

Seg. ID	Name	Req. Des.	Max Use
ISA	Interchange Control Header	M	1
TA3	Interchange Delivery Notice	M	>1
IEA	Interchange Control Trailer	M	1

Figure 4: Order of the Interchange Segments for Service Request Handler

3.6 Date and Time in the Interchange Segments

3.6.1 Basic Interchange Service - Dates and Times

The dates and times used in the ISA and TA1 segments represent local date and time of the interchange sender.

3.6.2 Optional Interchange Service Requests - Dates and Times

For optional interchange service request segments, a relationship to universal time coordinate (UTC) is provided to allow calculation of the time and date independent of the location of the interchange sender and the service request handler. This is particularly important if the interchange sender and the service request handler do not have the same local time.

3.6.3 Interchange Delivery Notice - Dates and Times

For the TA3, the date and time used to identify an interchange is copied from the interchange header and therefore represent local date and time of the interchange sender. The dates and times used as action dates and times shall be UTC.

4 Syntax

4.1 Basic Structure

A data element corresponds to a data field in data processing terminology. It is the smallest named item in the standard.

A control segment has the same structure as a data segment; the distinction is in the usage. The data segment is used primarily to convey user information, while the control segment is used primarily to convey control information and to group data segments. A data segment corresponds to a record in data processing terminology. The data segment begins with a segment ID and contains related data elements.

Other definitions, such as data element types, may be found in X12.6 Application Control Structure.

4.2 Syntax Notation

The syntax is defined in Backus Naur Form (BNF), which is described below. Other syntax elements may be found in X12.6 Application Control Structure.

- Lower case words denote syntactic constructs which are enclosed in angle brackets. The underscore is included as a valid character, e.g.,

  <transaction_set>

- The defined construct is on the left side of a statement and is separated from the defining right side by the symbol "::=", e.g.,

  <one_construct> ::= <another_construct>

- Uppercase words are predefined labels, e.g.,

  TRM

- Lowercase words not enclosed in angle brackets denote a general class of items which are not further defined in this section of the standard, e.g.,

```
data
```

- Brackets denote optional items, e.g.,

```
[<one_construct>]
```

- When alternative definitions exist, they may be shown separated by a vertical bar "|", e.g.,

```
<letter_or_digit> ::= <uppercase_letter> | <digit>
```

- Braces enclose an item which may appear zero or more times, e.g.,

```
<unsigned_integer> ::= <digit> {<digit>}
```

4.3 Delimiter Specifications

The delimiters consist of three separators and a terminator. The delimiters are devised for inclusion within the data stream of the transfer. The delimiters are:

- <tr> segment terminator
- <gs> data element separator
- <us> component element separator
- <rs> repetition separator

The delimiters are assigned by the interchange sender. These characters are disjoint from those of the data elements; if a character is selected for the data element separator, the component element separator, the repetition separator or the segment terminator from those available for the data elements, that character is no longer available during this interchange for use in a data element. The instance of the terminator <tr> must be different from the instance of the data element separator <gs>, the component element separator <us> and the repetition separator <rs>. The data element separator, component element separator and repetition separator must not have the same character assignment.

4.4 Interchange Structure Syntax

The BNF for the syntax of this standard is listed below. Other syntax elements not defined here may be found in X12.6 Application Control Structure.

```
<interchange_structure> ::= <inter_header>
                            <inter_request> | <inter_response>
                            <inter_trailer>

<inter_request> ::= [<grade_of_service>]
                    [<time_delay>]
                    {<inter_ack>}
                    {<functional_group>}

<inter_response> ::= <inter_status> {<inter_status>}
```

4.4.1 Basic Interchange Segments

The basic interchange segments consist of the ISA, the IEA, and the TA1 segments.

4.4.1.1 Interchange Control Header Segment (ISA)

```
<inter_header>     ::= ISA <gs> <authorization>
                       <gs> <security>
                       <gs> <sender>
                       <gs> <receiver>
                       <gs> <inter_date_time>
                       <gs> <rs>
```

```
                        <gs> <version_id>
                        <gs> <inter_control>
                        <gs> <ack_requested>
                        <gs> <test_indicator>
                        <gs> <us>
                        <tr>

<authorization>    ::= (<authorization_qualifer>
                        <gs> <authorization_info>) | ("00" <gs>
                        <no_information>)
<authorization_qualifier> ::= <id>
<authorization_info> ::= <string>
<security>         ::= (<security_qualifier> <gs> <security_info>) | ("00"
                   <gs> <no_information>)
<security_qualifier> ::= <id>
<security_info>    ::= <string>
<sender>           ::= <inter_id_qualifier> <gs> <sender_id>
<receiver>         ::= <inter_id_qualifier> <gs> <receiver_id>
<inter_id_qualifier> ::= <id>
<sender_id>        ::= <string>
<receiver_id>      ::= <string>
<inter_date_time>  ::= <date> <gs> <time>
<version_id>       ::= <id>
<inter_control>    ::= <numeric>
<ack_requested>    ::= <id>
<test_indicator>   ::= <id>
<no_information>   ::= <space> <space> <space> <space> <space> <space> <space>
                        <space> <space> <space>
```

4.4.1.2 Interchange Trailer Segment (IEA)

```
<inter_trailer>    ::= IEA <gs> <number_groups>
                        <gs> <inter_control>
                        <tr>
<number_groups>    ::= <numeric>
```

4.4.1.3 Interchange Acknowledgment Segment (TA1)

```
<inter_ack>        ::= TA1 <gs> <inter_control>
                        <gs> <date>
                        <gs> <time>
                        <gs> <ack_code>
                        <gs> <note_code>
                        <tr>
      <ack_code>   ::= <id>
      <note_code>  ::= <id>
```

4.4.2 Interchange Service Request Segments

The interchange service request segments are the Grade of Service segment (ISB) and the Deferred Delivery Request (ISE) segments.

4.4.2.1 Grade of Service Request Segment (ISB)

```
<priority>          ::= ISB <gs> <pri_code> <tr>
      <pri_code>    ::= <id>
```

4.4.2.2 Deferred Delivery Request Segment (ISE)

```
<time_delay>        ::= ISE <gs> <delay_date>
                        <gs> <delay_time>
                        [<gs> <delivery_time_code>]
                        <tr>
      <delay_date> ::= <date>
      <delay_time> ::= <time>
      <delivery_time_code> ::= <numeric>
```

4.4.3 Interchange Delivery Notice

4.4.3.1 Interchange Delivery Notice Segment (TA3)

```
<inter_status>      ::= TA3 <gs> <service_handler_identification>
                        <gs> <error_reason_code>
                        <gs> <interchange_information>
                        <gs> <interchange_action>
                        <gs> [<interchange_action>]
                        <gs> <reference>
                        <gs> <reference>
                        <gs> <reference>
                        <tr>
<service_handle_identification> ::= <service_handler_qualifier>
                        <gs> <service_handler_id>
<service_handler_qualifier> ::= <id>
<service_handler_id> ::= <string>
<interchange_action> ::= <interchange_action_code>
                        <gs> <interchange_action_date>
                        <gs> <interchange_action_time>
<interchange_action_code> ::= <id>
<interchange_action_date> ::= <date>
<interchange_action_time> ::= <time>
<error_reason_code> ::= <id>
<interchange_information> ::= <interchange_segment_id>
                        <gs> <interchange_control_number>
                        <gs> <interchange_date>
                        <gs> <interchange_time>
                        <gs> <interchange_sender_qualifier>
                        <gs> <interchange_sender>
                        <gs> <interchange_receiver_qualifier>
                        <gs> <interchange_receiver>
                        <gs> <first_reference>
                        <gs> <second_reference>
<interchange_segment_id> ::= <id>
<interchange_control_number> ::= <string>
<interchange_date> ::= <string>
<interchange_time> ::= <string>
<interchange_sender_qualifier> ::= <string>
```

```
<interchange_sender> ::= <string>
<interchange_receiver_qualifier> ::= <string>
<interchange_receiver> ::= <string>
<first_reference>  ::= [<string>] <gs> [<string>]
<second_reference> ::= [<string>] <gs> [<string>]
<reference>        ::= [<reference_code_qualifier>]
                       <gs> [<reference_code>]
<reference_code_qualifier> ::= <id>
<reference_code>   ::= <string>
```

5 Interchange Segment Specifications

5.1 Basic Interchange Segments

5.1.1 Segment Specifications

Specifications for the interchange segments are provided in the segment directory.

5.1.2 Interchange Control Header Segment (ISA)

Purpose:
To start and identify an interchange of zero or more functional groups and interchange-related control segments.

The actual values of the data element separator, component element separator, repetition separator, and segment terminator for this interchange are set by the interchange control header. For a particular interchange, the value at the fourth character position of the interchange control header is the data element separator, and the value of the last character position is the segment terminator. The extent of this particular usage of the data element separator, component element separator, and the segment terminator is from this header to, and including, the next interchange trailer. The interchange control number value in this header must match the value in the same data element in the IEA segment.

In order to provide sufficient discrimination for the acknowledgment process to operate reliably and to ensure that audit trails are unambiguous, the combination of interchange sender's qualifier and ID (ISA05, ISA06), interchange receiver's qualifier and ID (ISA07, ISA08) and the interchange control number value (ISA13) shall by themselves be unique within a reasonably extended time frame whose boundaries shall be defined by trading partner agreement. Because at some point it may be necessary to reuse a sequence of control numbers, the Interchange Date and Time may serve as an additional discriminant only to differentiate interchange identity over the longest possible time frame.

5.1.3 Interchange Control Trailer Segment (IEA)

Purpose:
To define the end of an interchange of zero or more functional groups and interchange-related control segments.

The interchange control number in this trailer must match the value in the same data element in the corresponding ISA segment.

5.1.4 Interchange Acknowledgment Segment (TA1)

Purpose:
To report the status of the processing of an interchange header and trailer by the addressed receiver. TA1 fields 1, 2, and 3 are used to identify the original interchange whose status is being reported.

5.2 Optional Interchange Segments

5.2.1 Grade of Service Request Segment (ISB)

Purpose:
To request a delivery priority for this interchange higher or lower than normally provided.
The priority request is performed by each service request handler, using each handler's criteria for providing priority delivery service.

5.2.2 Deferred Delivery Request Segment (ISE)

Purpose:
To specify the earliest time when the interchange can be delivered.

The time and date in the ISE segment indicates the earliest time the interchange can be delivered to the interchange receiver.

5.3 Interchange Delivery Notice Segment (TA3)

Purpose:
To provide a notice from the receiving service request handler to the sending service request handler that an interchange was delivered or not delivered to the interchange receiver's mailbox, or some other ancillary service was performed, and that the interchange receiver retrieved, refused, or purged the interchange; TA3 is exchanged only between service request handlers; use of the TA3 segment is optional.

6 Data Element Specifications

6.1 Data Element Definitions

Specifications for the data elements of this standard are provided in the Data Element Dictionary. Code Sources for code lists not maintained by ASC X12 appear in the Index to Code Sources at the end of the Data Element Dictionary. Reference to Code Source Identifiers follow Data Element descriptions or Data Element definitions.

Appendix A — Implementation Considerations

This Appendix covers implementation considerations regarding the character sets used in the interchange of the transaction sets with particular emphasis on the delimiters. The basic and extended character sets are defined in X12.6; reference should be made to that standard for a definition of those character sets. Portions of those definitions are repeated here as required for understanding. The BNF used below is defined in X12.6.

1 Basic Character Set

The selection that follows is designed to have representation in the common character code schemes of EBCDIC, ASCII, and CCITT International Alphabet 5. The ASC X12 standards are graphic-character-oriented; therefore, common character encoding schemes other than those specified herein may be used as long as a common mapping is available. Since the graphic characters have an implied mapping across character code schemes, those bit patterns are not provided here.

The basic character set of this standard includes those selected from the uppercase letters, digits, space, and special characters as specified below.

<uppercase_letter> ::=	A	...	Z			
<digit> ::=	0	...	9			
<special_char> ::=	"!"	"""	"&"	"'"	"("	")"
	"*"	"+"	","	"-"	"."	"/"
	":"	";"	"?"	"="		
<space> ::=	" "					

2 Extended Character Set

An extended character set may be used by negotiation between the two parties and includes the lowercase letters and other special characters as specified below.

<lower_case_letter> ::=	"a"	...	"z"			
<other_special_char> ::=	"%"	"@"	"["	"]"	"_"	"{"
	"}"	"\"	"|"	"<"	">"	"~"
	"^"	"`"				
<national_char> ::=	"#"	"$"				

It should be noted that

<special_char>
<other_special_char>

and

<national_char>

include several character codes which have multiple graphical representations for a specific bit pattern. The complete list appears in other standards such as CCITT S.5. Use of the USA graphics for these codes presents no problem unless data is exchanged with an international partner. Other problems, such as the translation of item descriptions from English to French, arise when exchanging data with an international partner, but minimizing the use of codes with multiple graphics eliminates one of the more obvious problems.

3 Control Characters

Two control character groups are specified which have only restricted usage. The common notation for these is also provided, together with the character coding in three common alphabets represented by their hexadecimal values. In the following table IA5 represents CCITT V.3 International Alphabet 5.

3.1 Terminal Control Set

The terminal control set includes characters used to control terminal display characteristics. These characters usually will not have an effect on a transmission system. These are presented by the syntactic entity of <term_control_char>.

SYM	Notation	EBCDIC	ASCII	IA5
BEL	bell	2F	07	07
HT	horizontal tab	05	09	09
LF	line feed	25	0A	0A
VT	vertical tab	0B	0B	0B
FF	form feed	0C	0C	0C
CR	carriage return	0D	0D	0D
NL	new line	15	(See Note)	
FS	file separator	1C	1C	1C
GS	group separator	1D	1D	1D
RS	record separator	1E	1E	1E
US	unit separator	1F	1F	1F

Note: The equivalent representation for the EBCDIC “new line” character is the character pair “carriage return” “line feed” in ASCII or IA5. If mapped in this manner, the superfluous “line feed” is ignored, and the “carriage return” is treated as the delimiter. For historical reasons, many systems support this mapping for use as the

<segment_terminator>

3.2 Communication Control Set

The communication control set includes those that may have an effect on a transmission system. These are represented by the syntactic entity of <comm_control_char>.

SYM	Notation	EBCDIC	ASCII	IA5
GS	group separator	1D	1D	1D
RS	record separator	1E	1E	1E
US	unit separator	1F	1F	1F
SOH	start of header	01	01	01
STX	start of text	02	02	02
ETX	end of text	03	03	03
EOT	end of transmission	37	04	04
ENQ	enquiry	2D	05	05
ACK	acknowledge	2E	06	06
DC1	device control 1	11	11	11
DC2	device control 2	12	12	12
DC3	device control 3	13	13	13
DC4	device control 4	3C	14	14
NAK	negative acknowledge	3D	15	15
SYN	synchronous idle	32	16	16
ETB	end of block	26	17	17

4 Specification of Delimiters

The delimiters are defined within this standard. This section of this Appendix covers the recommended maximum range of the characters for the delimiters. A series of specific recommendations are provided in Section 5.

4.1 Terminator

<tr> ::= segment_terminator

4.2 Data Element Separator

<gs> ::= data_element_separator

4.3 Subelement Separator

<us> ::= sub_element_separator

4.4 Repetition Separator

<rs> ::= repetition_separator

5 Recommendations for the Delimiters

Because of the potential conflicts with either the data elements or with special uses in transmission and device control, the following recommendations are provided for the delimiter character selection.

The following characters usually occur in data. They should not be used as delimiters:

<uppercase_letter>
<lowercase_letter>
<digit>
<space>
“-” (minus sign)

The following characters often appear in data. Use as delimiter characters with caution.

<special_char>

The following characters sometimes appear in data. Use as delimiter characters with caution.

<other_special_char>

The following characters are used for terminal control. Use as delimiter characters with caution.

<term_control_char>

The following characters may disrupt communications. Use as delimiter characters with caution.

<comm_control_char>

Delimiter characters must be chosen with care, after consideration of data content, limitations of the transmission protocol(s) used, and applicable industry conventions. In the absence of other guidelines, the following recommendations are offered:

<tr> terminator				
<file separator>	<new line>	<carriage return>	<line feed>	"~"
Note: the "~" was chosen for its infrequency of use in textual data.				

<gs> data element separator		
<horizontal tab>	"*"	
Note: the "asterisk" has a potential for conflict with the data stream; however, it has been the preferred character historically, and so is explicitly listed in the recommendation.		

<us> component element separator		
<vertical tab>	"\"	
Note: the "\" was chosen for its infrequency of use in textual data.		

<rs> repetition separator		
<record separator>	"^"	
Note: the "^" was chosen for its infrequency of use in textual data.		

X12.56 INTERCONNECT MAILBAG CONTROL STRUCTURES

1 Purpose and Scope

This Standard contains the format and establishes the data contents of the Interconnect Mailbag Control Structures for use within the context of an Electronic Data Interchange (EDI) environment. This standard defines the control segments used to start and end a mailbag containing EDI data to be exchanged between two interconnecting entities. The mailbag includes zero or more interconnect acknowledgment segments and zero or more EDI data interchanges. The basic interconnect mailbag consists of an interconnect mailbag header segment (IH), the EDI data to be transmitted from one interconnect entity to another, and the interconnect mailbag trailer segment (IT). In addition, an interconnect mailbag acknowledgment segment (IA) is provided to report complete receipt and safe storage of an interconnect mailbag.

The purpose of this standard is to provide control structures and an audit mechanism to facilitate the exchange and receipt acknowledgment of EDI data between interconnecting entities. The original sender and the ultimate receiver of the data contained in the mailbag have no responsibility for creating, managing, or removing the interconnect mailbag segments. This standard is solely for use between sites acting as interconnect entities.

Delivery of data from the original sender to the ultimate receiver may require several interconnect links. It is recognized that other point-to-point data tracking mechanisms exist. The interconnect mailbag control structures are designed to stand alone in addressing a given interconnect link. The structures are in no way dependent on the types of data tracking mechanisms that may be used in links prior to or following a link that employs the interconnect mailbag control structures.

2 Referenced and Related Standards

This standard is intended to be used with standards developed by ASC X12, including X12.5 Interchange Control Structures and X12.6 Application Control Structure. This standard does not affect any ASC X12 standards that define the syntax of the EDI data interchanges contained within an interconnect mailbag.

3 Description of Use

The interconnect mailbag header and trailer segments envelope zero or more interconnect mailbag-related control segments or interchanges and perform the following functions:

- Identify the interconnect mailbag sender and receiver
- Provide control information for the interconnect mailbag
- Allow for logon and password information

The interconnect mailbag header segment is transmitted first, followed by any optional interconnect mailbag acknowledgments, followed by any interchanges, with the interconnect mailbag trailer segment following as the last segment in the mailbag.

The interconnect mailbag acknowledgment segment is used to acknowledge one interconnect mailbag header and trailer envelope. The interconnect mailbag acknowledgment segment may be used to report the success or failure of the syntactical analysis of the interconnect mailbag segments. There is no acknowledgment for a syntactically correct interconnect mailbag containing only interconnect mailbag acknowledgments, thereby preventing an endless loop of acknowledgments. The flow of the original interconnect mailbag and the corresponding interconnect mailbag acknowledgment are diagrammed in Figure 1 with increasing time down the page.

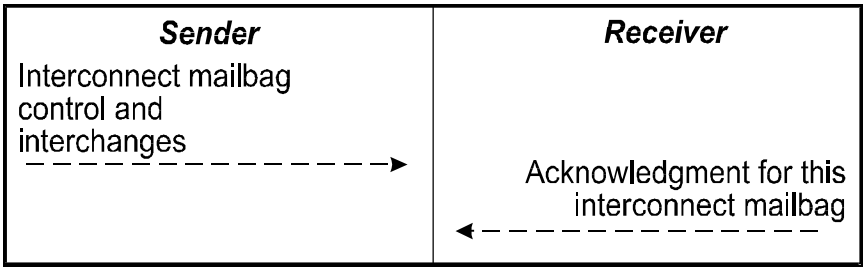

Figure 1: Flow of Original Interconnect Mailbag and Corresponding Acknowledgment

The interconnect mailbag control number value in the interconnect mailbag acknowledgment segment (IA) is the same as for the interconnect mailbag header segment (IH) for which the interconnect mailbag acknowledgment was prepared. This control number serves as a link between the interconnect mailbag header and trailer and the acknowledgment of that header and trailer.

The interconnect mailbag acknowledgment does not report on the processing or delivery status of the interchange(s) in the interconnect mailbag. The interconnect mailbag acknowledgment segment serves as receipt status notification (positive or negative) from an interconnect mailbag receiver to an interconnect mailbag sender.

The interconnect mailbag acknowledgment control segments (IA) are placed after the interconnect mailbag header segment (IH) and before the first interchange or before the trailer if there are no interchanges. More than one interconnect mailbag acknowledgment segment (IA) may be placed after the interconnect mailbag header providing the interconnect mailbag sender and receiver ID values in the interconnect mailbag header segment (IH) are appropriate for the proper delivery of the acknowledgment.

Interconnect mailbags and interconnect mailbag acknowledgments are used only between interconnecting entities.

4 Terms and Definitions

X12.6 Application Control Structure contains the formal definitions of all terms related to electronic data interchange. Additional terms used in this standard are defined below.

INTERCONNECT ENTITY
Any organization that communicates EDI data with another organization where neither communicating party is responsible for processing and acting upon the semantic data (business transactions which may be contained in the mailbag).

SERVICE PROVIDER
Any process or set of processes that accepts an interchange from a sender and conveys it to a mailbox that represents the addressed receiver. Direct connections between senders and receivers involve no service providers.

RECEIPT AND SAFE STORAGE
An interconnect mailbag is received and safely stored when: (a) the control numbers in the interconnect mailbag header and trailer match; (b) the control totals listed in the interconnect mailbag trailer are consistent with the number of interconnect mailbag acknowledgments and interchanges in the interconnect mailbag; and (c) the entire interconnect mailbag has been successfully stored such that its contents can be obtained in their entirety at some later time for processing or delivery or both.

DELIVERY

An interchange is delivered when the interchange is placed in the mailbox of the receiver addressed in the interchange header segment.

MAILBOX
A protected receiving location with a unique address such that, once an interchange is placed in the mailbox of the receiver, it may only be retrieved by the receiver.

5 Interconnect Mailbag Concepts

The primary elements of this standard are three control segments intended to be used by interconnect entities for EDI management functions.

5.1 Interconnect Entities

An interconnect entity may be charged to perform some syntactic processing of the data, such as translation or conversion, on behalf of the original sender or ultimate receiver of the data. However, a key element of an entity being considered an interconnect entity is that the entity is not a participating party in any business transaction represented in the EDI data being transferred, and the entity is not transferring the EDI data to or from the original sender or the ultimate receiver. The transaction set originator and ultimate receiver will never see nor use the interconnect mailbag control segments.

An original sender or ultimate receiver whose data center acts as a sending or receiving representative (as opposed to the end processing site) for the end user division or department may elect to have their data center use the interconnect mailbag structure when exchanging EDI data with an entity that is likewise not the original sender or ultimate receiver. For such cases, the user's data center would only use the interconnect mailbag structure when exchanging data with a service provider or the data center of another user whose data center was acting as a sending or receiving representative for an end user. An internal corporate routing or network mechanism is an example of this type of end user representative. In any event, the end user would not see the interconnect mailbag.

An entity may be considered an interconnect entity in some communication exchanges and not an interconnect entity in others, depending on the nature of the party with which the data is being exchanged. For example, a service provider receiving an EDI data transmission from the original sender of the data is not considered to be an interconnect entity and would not receive interconnect mailbag segments from that sender. If, however, that same service provider transmits that EDI data to a second service provider for delivery to the ultimate receiver, both service providers would be considered to be acting as interconnect entities for that transfer.

5.2 Interconnect Mailbag Defined

The interconnect mailbag header and the interconnect mailbag trailer form the basic interconnect mailbag. The interconnect mailbag contains data to be transmitted from one interconnect entity to another interconnect entity. EDI data interchanges contained in a mailbag may be from one or more original senders and to one or more ultimate receivers. The interconnect mailbag acknowledgment provides an interconnect entity with an audit mechanism to acknowledge the complete receipt and safe storage of an interconnect mailbag.

5.2.1 The Interconnect Mailbag Header Segment (IH)

The IH segment indicates the beginning of an interconnect mailbag, establishes the version of the interconnect mailbag control structures, identifies the interconnect mailbag sender and receiver, provides control information for the interconnect mailbag, and allows for the use of a logon ID and password for the interconnect mailbag.

5.2.2 The Interconnect Mailbag Trailer Segment (IT)

The IT segment indicates the end of an interconnect mailbag, provides control information, and reports control totals for the interconnect mailbag.

5.2.3 The Interconnect Mailbag Acknowledgment Segment (IA)

The IA segment is sent by the interconnect mailbag receiver to the interconnect mailbag sender. The interconnect mailbag acknowledgment is used to perform the following functions:

- Acknowledge the complete receipt and safe storage of an interconnect mailbag
- Report the receipt of an interconnect mailbag with incomplete contents received during a communications session that ended normally at a data link protocol level
- Report the receipt of a syntactically invalid interconnect mailbag received during a communications session that ended normally at a data link protocol level

When a complete and syntactically valid interconnect mailbag is received, the following interconnect mailbag content conditions dictate whether an interconnect mailbag acknowledgment is to be returned:

- If the interconnect mailbag contains one or more interchanges, an interconnect mailbag acknowledgment must be returned.
- If the interconnect mailbag contains no interconnect mailbag acknowledgments and no interchanges, an interconnect mailbag acknowledgment may be returned.

NOTE: There is no provision to acknowledge or respond to a complete and syntactically valid interconnect mailbag containing only interconnect mailbag acknowledgments (and no interchanges). This prevents the possibility of an endless cycle of acknowledgments and responses.

When an incomplete or invalid interconnect mailbag is received during a communications session that ended normally at a data link protocol level, a negative interconnect mailbag acknowledgment must be returned regardless of the contents of the incomplete or invalid mailbag.

If a communications session terminates abnormally at a data link protocol level, an interconnect mailbag acknowledgment (IA) is not returned in response to the data received during that failed session. Furthermore, the partial file received during that failed communications session should not be processed.

The interconnect mailbag acknowledgment (IA) must be transmitted in an interconnect mailbag whose sender and receiver IDs are equal to the receiver and sender IDs, respectively, from the mailbag being acknowledged. For example, a mailbag from sender A to receiver B is acknowledged with an interconnect mailbag acknowledgment (IA) in an interconnect mailbag from sender B to receiver A.

5.2.3.1 Timing The Interconnect Mailbag Acknowledgment

When the use of an interconnect mailbag acknowledgment is appropriate, the interconnect entity issuing the interconnect mailbag acknowledgment shall transmit the acknowledgment within an agreed upon period from the time the interconnect mailbag being acknowledged was received. The definition of what this response period should be shall be determined between each pair of interconnect entities. The response time agreed upon between interconnect entities A and B shall have no bearing on the response time established for interconnect entities B and C or A and C. In any event, it is likely that the maximum response time for the return of an interconnect acknowledgment will ultimately be suggested by the original EDI sender and the ultimate EDI receiver whom the interconnecting entities are representing.

If an interconnect entity transmits an interconnect mailbag containing one or more interchanges to a receiving interconnect entity, and the communications session terminates normally at a data link protocol level, the sending interconnect entity shall expect to receive an interconnect mailbag acknowledgment from the receiving interconnect entity within the agreed upon response period established for their interconnect relationship. Failure to receive an interconnect mailbag acknowledgment for the transmitted mailbag within the agreed upon response period shall precipitate human intervention on the part of the initial sending interconnect entity to determine the status of the unacknowledged interconnect mailbag.

5.2.3.2 Interconnect Mailbag Acknowledgment Action Codes

The IA segment provides the acknowledging interconnect entity with a mechanism to indicate the action to be taken under various situations. The following defines the intent of each of these action codes:

Action Code A (Accept)
An action code equal to "A" indicates that the interconnect mailbag is received in full, safely stored, and accepted. An accept response means only that the interconnect mailbag header and trailer control numbers match, the interconnect mailbag is syntactically valid, and the interconnect mailbag trailer control totals have been verified as accurate for the received interconnect mailbag. The accept response makes no claims that the receiving interconnect entity can process the mailbag contents, recognize the addressees indicated in any interchanges in the mailbag, or is able to deliver the interchange. The accept response simply indicates the complete receipt and safe storage of the mailbag and its contents (if any).

Action Code R (Reject/Retransmit)
An action code equal to "R" indicates that one or more problems have been detected with the interconnect mailbag. The data contents (if any) of the rejected interconnect mailbag will not be processed. The interconnect entity sender of the rejected interconnect mailbag should retransmit the mailbag.

NOTE: The reject/retransmit response is used only to indicate receipt of an *incomplete* interconnect mailbag received via a communication session that ended normally. This type of response should not be used to indicate receipt of a partial interconnect mailbag received during a communication session that ended abnormally because the interconnect mailbag sender will be able to detect such failures during the communications session. This response is not intended to indicate the inability or unwillingness of the receiving interconnect entity to process the mailbag contents of a complete mailbag, recognize the addressees, or deliver any interchanges in the mailbag.

Action Code C (Reject/Contact)
An action code equal to "C" indicates that one or more problems have been detected with the interconnect mailbag. The nature of the problem is such that retransmission of the mailbag is unlikely to correct the problem. The data contents (if any) of the rejected interconnect mailbag will not be processed as is. The interconnect administrator from the interconnect entity issuing the reject notice will contact the interconnect administrator from the interconnect entity receiving the reject notice to resolve the problems found in the interconnect mailbag.

NOTE: This response should only be used to indicate the receipt of an *invalid* interconnect mailbag received via a communication session that ended normally. This response should not be used to indicate receipt of a partial interconnect mailbag received during a communication session that ended abnormally, because the interconnect mailbag sender will be able to detect such failures during the communications session. The reject response is not intended to indicate the inability or unwillingness of the receiving interconnect entity to process the mailbag contents of a complete mailbag, recognize the addressees, or deliver any interchanges in the mailbag.

The interconnect entity that receives a reject/contact response should not retransmit the mailbag in question without first resolving the appropriate issues with the interconnect entity issuing the reject. This code should be used for cases involving multiple errors for a given interconnect mailbag.

6 Interconnect Mailbag Syntax Notation

In this section the syntax for the interconnect mailbag segments is given in Backus Naur Form (BNF), described below. Other syntax elements may be found in X12.6 Application Control Structure.

6.1 Backus Naur Form (BNF) Elements Explained

Lowercase words denote syntactic constructs, which are enclosed in angle brackets. The underscore is included as a valid character. For example:

```
<transaction_set>
```

The defined construct is on the left side of a statement and is separated from the defining right side by the symbol (::=). For example:

```
<one_construct> ::= <another_construct>
```

Uppercase words are predefined labels. For example:

```
TRM
```

Lowercase words not enclosed in angle brackets denote a general class of items that are not further defined in this section of the standard. For example:

```
data
```

Brackets denote optional items. For example:

```
[<one_construct>]
```

A repeat count range for a repeatable construct that is appended to that particular construct is added to the BNF syntax of X12.6 Application Control Structure. The range of the repeat count is /minimum:maximum/. The repeatable construct is in braces. For example, a numeric field with one to six digits could be specified as:

```
<a_number> ::= {<digit>}/1:6/
```

6.2 Interconnect Mailbag Structures

The BNF for the syntax of this standard is given below. Other syntax elements not defined here may be found in X12.6 Application Control Structure.

```
<interconnect_mailbag_structures> ::= <interconnect_mailbag_header>
      {<interconnect_mailbag_acknowledgment>}{<interchange_structure>}
      <interconnect_mailbag_trailer>
```

NOTE: <interchange_structure> is defined in X12.5 Interchange Control Structures.

6.3 Interconnect Mailbag Segments

The basic interconnect mailbag segments consist of the IH header segment and the IT trailer segment.

6.3.1 Interconnect Mailbag Header Segment

```
<interconnect_mailbag_header> ::= IH <gs> <interconnect_mailbag_version_#>
      <gs> <interconnect_logon_id> <gs> <interconnect_password>
      <gs> <interconnect_sender> <gs> <interconnect_receiver>
      <gs> <interconnect_date_time> <gs> <interconnect_time_code>
      <gs> <interconnect_control> <gs> <interconnect_test_indicator>
      <tr>
<interconnect_mailbag_version_#> ::= <id>
<interconnect_logon_id> ::= <string>
<interconnect_password> ::= <string>
<interconnect_sender> ::= <interconnect_id_qualifier>
      <gs> <interconnect_sender_id>
<interconnect_receiver> ::= <interconnect_id_qualifier>
      <gs> <interconnect_receiver_id>
<interconnect_id_qualifier> ::= <id>
```

```
<interconnect_sender_id> ::= <string>
<interconnect_receiver_id> ::= <string>
<interconnect_date_time> ::= <date> <gs> <time>
<interconnect_time_code> ::= <id>
<interconnect_control> ::= <numeric>
<interconnect_test_indicator> ::= <id>
```

6.3.2 Interconnect Mailbag Trailer Segment

```
<interconnect_mailbag_trailer> ::= IT <gs> <interconnect_control>
      <gs> <interconnect_ack_count> <gs> <interconnect_intchg_cnt>
      <tr>
<interconnect_control> ::= <numeric>
<interconnect_ack_count> ::= <numeric>
<interconnect_intchg_cnt> ::= <numeric>
```

6.3.3 Interconnect Mailbag Acknowledgment Segment

The interconnect mailbag acknowledgment segment consists of the IA segment.

```
<interconnect_mailbag_ack> ::= IA <gs> <interconnect_control>
      <gs> <interconnect_ack_action_code>
      {<gs> <interconnect_error_code>}/0:5/
      <tr>
<interconnect_control> ::= <numeric>
<interconnect_ack_action_code> ::= <id>
<interconnect_error_code> ::= <id>
```

7 Specifications for Interconnect Mailbag Control Structures

7.1 Sequence of Interconnect Mailbag Segments

The interconnect mailbag segments shall occur in the sequence shown in Figure 2.

NOTE	POS NO.	SEG. ID	NAME	REQ. DES.	MAX. USE	LOOP REPEAT
	010	IH	Interconnect Mailbag Header	M	1	
	020	IA	Interconnect Mailbag Acknowledgement	O	>1	
N	030	----	Interchanges (see note)	O	>1	
	040	IT	Interconnect Mailbag Trailer	M	1	

NOTES	
030	An interchange is not an interconnect mailbag component of this standard, but appears in this figure to establish positioning for the interchange(s). Zero, one, or more than one interchange may appear in one interconnect mailbag.

Figure 2: Segment Sequence Diagram

7.2 Date and Time in the IH Segment

The dates and times used in the interconnect mailbag header segment represent the point at which the interconnect mailbag was created for initial transmission. An interconnect mailbag time code is used to identify the basis of the time that is given in the interconnect mailbag time element.

In cases where it is necessary to retransmit an interconnect mailbag, the interconnect mailbag time should not be changed from its original value when the mailbag was created. This aids the receiver in the identification of the mailbag and the determination of the age of the mailbag.

7.3 Delimiter Specifications

The delimiters for the mailbag segments are a data element separator and a segment terminator. The delimiters that shall be used in the segments of the interconnect mailbag structure (IH, IT, and IA) are as follows:

Description	Delimiter	Name	EBCDIC (Hex)	ASCII (Hex)
Segment Terminator	~	Tilde	A1	7E
Data Element Separator	*	Asterisk	5C	2A

Although X12.5 Interchange Control Structures discourages the use of these characters because of the likelihood of their inclusion in data element contents, these characters are not needed in the interconnect mailbag data elements. Neither the tilde nor the asterisk conflicts with the reserve characters used by commonly implemented interconnect protocols (2780, 3780, SNA, etc.).

These delimiters apply only to the interconnect mailbag segments, not to any interchanges being transferred in the mailbag, and in no way restrict the values that may be used in the interchanges placed in the mailbag.

7.4 Interconnect Mailbag Segment Specifications

7.4.1 Data Segment Diagram Key

The format described in Figure 3 shows, in the general case, how interconnect mailbag segments are diagrammed and how reference information is provided in the diagram. It must be noted that the actual transfer of information does not include all of the reference information; only the data segment identifier characters, the delimiters, and the values for each included data element are actually transferred in the interconnect mailbag segments.

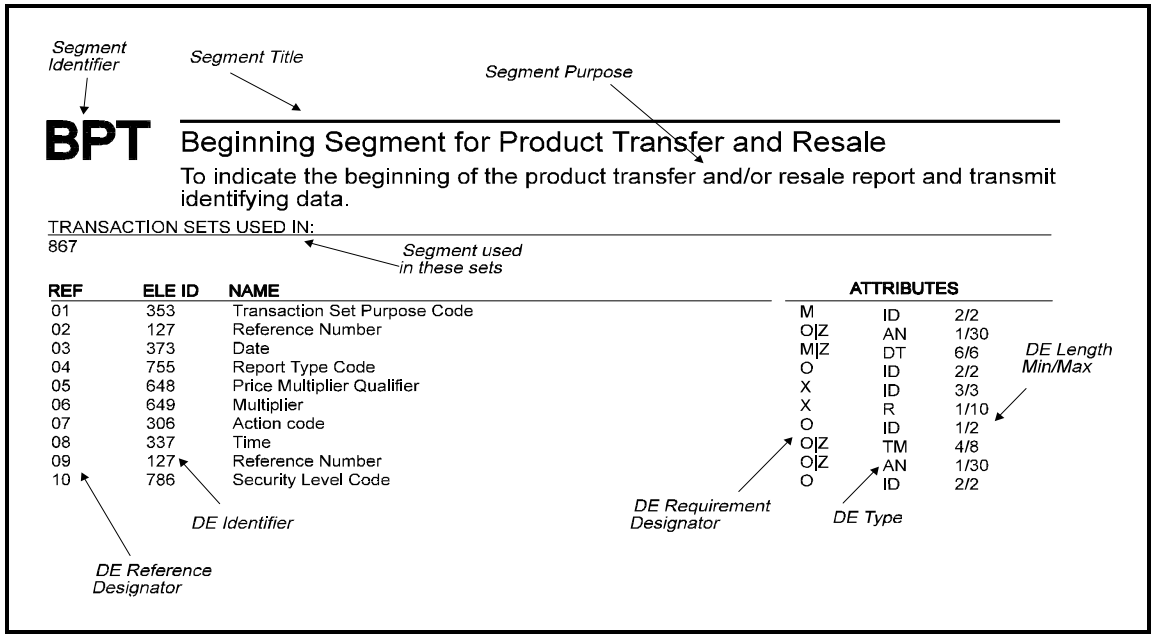

Figure 3: Segment Key Diagram

7.4.2 Interconnect Control Segments

The specifications for the control segments of this standard are shown in the Segment Directory.

7.5 Interconnect Mailbag Data Element Specifications

7.5.1 Data Element Diagram Key

The format described in Figure 4 shows, in the general case, how a data element is diagrammed and how the reference information is provided in the diagram. It must be noted that actual transfer of information does not include all of the reference information; only the values for each included data element are actually transferred in the interconnect mailbag control segments.

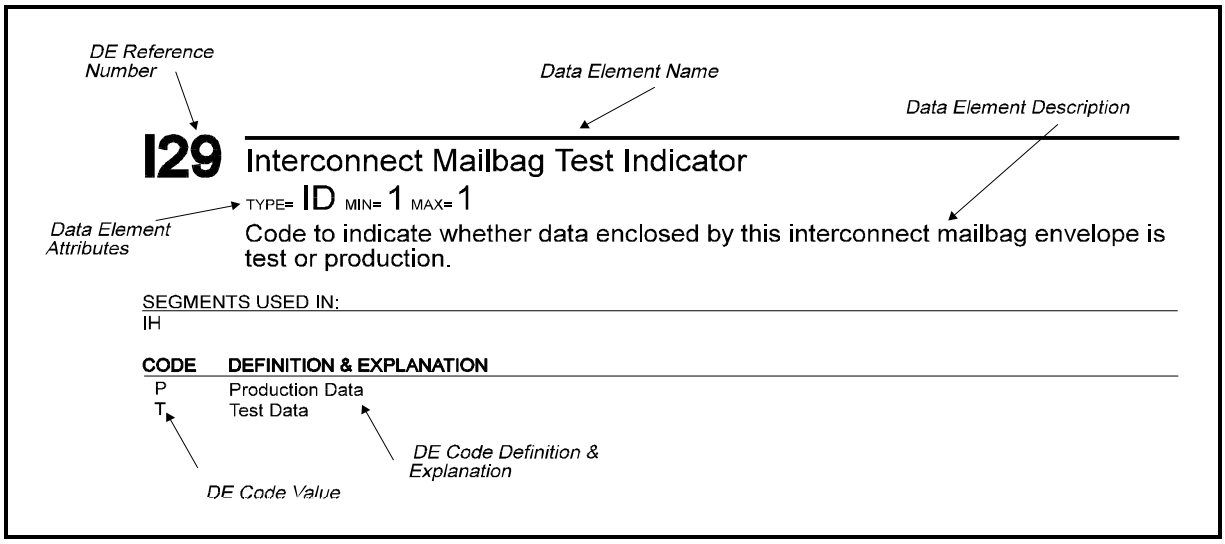

Figure 4: Data Element Key Diagram

7.5.2 Data Element Definitions

The specifications for the data elements of this standard are shown in the Data Element Dictionary.

X12.58 SECURITY STRUCTURES

1 Purpose and Scope

The purpose of this standard is to define the data formats for authentication, encryption, and assurances in order to provide integrity, confidentiality, and verification and non-repudiation of origin for two levels of exchange of Electronic Data Interchange (EDI) formatted data defined by Accredited Standards Committee (ASC) X12. These security services can be applied at either the functional group level or the transaction set level or both.

This standard defines data formats for the following optional mechanisms:

- Authentication of EDI encoded data
- Compression of EDI encoded data
- Encryption of EDI encoded data
- Assurances applied to EDI encoded data
- Assurances applied to EDI encoded data and passed independent of EDI encoded data
- Non-repudiation of Origin
- Non-repudiation of Receipt
- Security protocol error reporting

The business requirements addressed in this standard for the security of EDI-formatted data encompass:

- Verification of the security originator to the security recipient
- Verification of assurance originator to any interested party ("non repudiation of sender")
- Electronic signatures, including expression of business purpose for applying each of several possible multiple signatures and date-time stamp
- Verification of data integrity ("error detection," "hash totals," "control totals")
- Confidentiality of business data
- Detection of replay (e.g., duplication), insertion/modification/deletion, or impersonation

2 Referenced and Related Standards

- American National Standard X3.4-1977, *Code for Information Interchange*
- American National Standard X3.92-1981, *Data Encryption Algorithm*
- American National Standard X3.106-1983, *Modes of Operation of the DEA*
- American National Standard X9.9-1986 (revised), *Financial Institution Message Authentication (Wholesale)*
- American National Standard X9.17-1985, *Financial Institution Key Management (Wholesale)*
- American National Standard X9.23-1988, *Encryption of Wholesale Financial Messages*
- American National Standard X9.32-1992, *Data Compression in Wholesale Financial Telecommunications*

When the American National Standards referenced above are superseded by a revision approved by ANSI, the revisions shall apply to this document.

The ASC X12 series of standards on electronic data interchange are interdependent. The following standards are required to interpret, understand, and use this standard:

- X12.6, Application Control Structure
- X12.5, Interchange Control Structures
- X12.30, Functional Acknowledgment (997)
- Secured Receipt or Acknowledgment (993)
- X12.22, Segment Directory
- X12.3, Data Element Dictionary

The following standard is a companion document that provides a mechanism for the exchange of keying materials in an ASC X12 environment:

Cryptographic Service Message Transaction Set (815)

For information on how to obtain copies of American National Standards, please contact:

American National Standards Institute
Publications Department
11 W. 42nd Street 13th Floor
New York, NY 10036
(212) 642-4900

For information on how to obtain copies of ASC X12 Standards, contact the ASC X12 Secretariat:

Data Interchange Standards Association, Inc.
Publications Department
333 John Carlyle Street • Suite 600
Alexandria, VA 22314-2852
Phone: (703) 548-7005 FAX (703) 548-5738
e-mail: *publications@disa.org*

3 Definitions and Common Abbreviations

X12.6 Application Control Structure provides formal definition of terminology related to the electronic data interchange standards developed by ASC X12.

3.1 Definitions

Algorithm
A clearly specified mathematical process for computation; a set of rules which, if followed, will give a prescribed result.

Assurance Originator
The entity responsible for generating an assurance.

Assurance Recipient
The entity responsible for verifying an assurance.

Assurances
A generalization of the concept of "electronic signature" which includes similar concepts such as notarization or a secured time stamp.

Authentication
The verification of the source, uniqueness, and integrity of a message (definition used in ANSI X9.9).
Ciphertext

Encrypted (enciphered) data.

Cryptographic Equipment
Hardware or software that performs cryptographic functions (e.g., encryption, authentication, key generation).

Cryptographic Key (also Key)
A parameter that determines the transformation from plaintext to ciphertext or vice versa. For example, a DEA key is a 64-bit parameter consisting of 56 independent bits and eight bits which may be used as odd parity bits.

Cryptographic Service Message
Message used for transporting keys or related information used to control a keying relationship (ANSI X9.17).

Data Compression
Reduction in the size of a data stream by the use of specialized encoding techniques.

Data Key
A key used to encrypt and decrypt or to authenticate data (ANSI X9.17). Note: this standard requires that different keys be used for authenticating and encrypting the same plaintext.

Decryption
The process of transforming ciphertext into plaintext (ANSI X9.23).

Digital Signature
Electronic signature based on public key technology.

Digest
The output of a hashing algorithm.

Dis-embodied Signature
Electronic signature based on public key technology that is passed independent of the associated document.

Electronic Signature
Generic term covering techniques such as hashing, message authentication codes, or digital signatures.

Encryption
The process of transforming plaintext into ciphertext.

Hash
A (mathematical) function which maps values from a (possibly very) large set of values into a smaller range of values.

Initialization Vector (IV)
A value used as a starting point for the encryption of a data sequence to increase security by introducing additional cryptographic variance and to synchronize cryptographic equipment.

Interchange Partners
Two parties who have previously agreed to exchange data. A party and a key center exchanging cryptographic service messages do not constitute interchange partners.

Interoperability
The ability to exchange EDI data or cryptographic keys, both manually and in an automated environment, with any other party implementing this standard, providing that both implementations use compatible options of the relevant standards and compatible communications facilities.

Key
See *CRYPTOGRAPHIC KEY.*

Keying Material
The data (e.g., keys and IVs) necessary to establish and maintain cryptographic keying relationships (ANSI X9.17).

Keying Relationship
The state existing between interchange partners in which they share at least one data key or key-encrypting pair.

Message Authentication Code (MAC)
A cryptographically derived hash total that can be used to verify the security originator to the security recipient and so protects the integrity of the applicable data.

Non-Repudiation
Ability of a party to a data exchange to obtain irrevocable proof, verifiable that some aspect of the data exchange occurred.

Octet
Eight bits.

Parity
A measure of the number of "1" bits in a group of "0" and "1" bits; either odd, even, or not relevant (i.e., none).

Parity Bit
A bit added to a group of "0" and "1" bits to make the parity of the group odd or even.

Party
Business entity that controls the application system that originates or receives EDI formatted data.

Plaintext
Unencrypted (unenciphered) data; intelligible data that can be directly acted upon without decryption.

Private Key
One of two keys used with an asymmetric cryptographic algorithm. The private key is known only to one entity.

Pseudo-Random
A value is pseudo-random if it is approximately random.

Public Key
One of two keys used with an asymmetric cryptographic algorithm. The public key is known to everyone.

Random
A value is random if it has an equal probability of being selected from all possible values; hence it is unpredictable.

Replay
The process of recording and repeating a message at a later time.

Secret Key
A key, used with a symmetric cryptographic algorithm, that is shared between the two entities.

Security Originator
The entity responsible for and authorized to apply cryptographic protection to a secured-body. **NOTE:** the security originator is logically distinct from the application sender.

Security Recipient
The entity responsible for verifying that the secured-body has not been altered in transit, for decoding encrypted text, or validating the identity of the security originator. **NOTE:** the security recipient is logically distinct from the application receiver.

Symmetric Key
See *SECRET KEY.*

Transformed Data
Transformed data is the encrypted data string resulting from application of an encryption algorithm to the span of characters beginning with the first character of the initialization vector and ending with the character preceding the segment delimiter of the last segment in the range of data to be secured.

3.2 Abbreviations

ASCII: American Standard Code for Information Interchange
CBC: Cipher Block Chaining
CFB: Cipher Feedback
DEA: Data Encryption Algorithm
EBCDIC: Extended Binary Coded Decimal Interchange Code
ECB: Electronic Code Book
IDA: Identity of Key for Authentication
IDE: Identity of Key for Encryption
IV: Initialization Vector
MAC: Message Authentication Code

4 Data Security Related Concepts

4.1 General

This section describes general cryptographic concepts as they apply to authentication, encryption, and assurances.

The models for authentication, encryption, and compression presented in this Section are those of ANSI X9.9, ANSI X9.23, and X9.32. These standards use the Data Encryption Algorithm (DEA) as described in ANSI X3.92 and the Modes of Operation of the DEA as defined in ANSI X3.106. This document makes provision for other standards -- including future ANSI or other standards -- by providing data structures for the use of other cryptographic algorithms, including asymmetric algorithms.

The companion document *Cryptographic Service Message Transaction Set (815)* deals with the management and distribution of symmetric keys, asymmetric keys, and certificates of authority.

The model for electronic signatures is based on the public key cryptographic standards work of ANSI X9F1, and the Secure Hash Algorithm (SHA) and the Digital Signature Algorithm (DSA) work of NIST that is contained in Federal Information Processing Standards 186 (FIPS 186). The approach taken is flexible enough to encompass other public key standards and *de facto standards.*

4.2 Authentication

Authentication is a technique used to:

- Verify the identity of the security originator of a message to the security recipient in order to detect spoofing or impersonation
- Verify the integrity of the message by detecting changes (modifications) in a message (including transmission errors) introduced between the security originator and the security recipient

- Protect a unique message identifier which is used to detect attempts at insertion, deletion, or replay of messages

The ANSI X9.9 standard employs ANSI X3.92 to compute a Message Authentication Code (MAC). The MAC is a cryptographically derived hash total that can be used to verify the security originator to the security recipient or protect the integrity of the applicable data.

Authentication is designed to provide protection for a message between the security originator and the security recipient. To provide that protection, the key used to perform the authentication must be known only by those two parties (or their authentication devices), and the message must not be modified in any way after the authentication process is performed by the security originator.

Authentication does not modify the text that is input to the process. The authenticated message can pass through any communications equipment that can handle the unauthenticated message. The receiving process must be able to reconstruct the original message from the data delivered by the communication equipment in order for the message to pass the checking process.

Basic requirements for a secured business interchange include the need to detect attempts at insertion, deletion, and replay of messages. This requirement can be satisfied by having the originating application place a combination of date and message identifier (such as a purchase order number) -- according to the requirements of ANSI X9.9 -- into the authenticated message so that the receiving application can uniquely identify each message sent. Then the security recipient can detect a repeated message by its repeated date and message identifier. Deleted messages can be detected by the originating application by the lack of an authenticated response (when one is used) or by the receiving application by reconciling the messages received against a list sent by the originating application in a separate authenticated message. Text inserted into the secured-body can be detected by the receiving application by the failure of the message to authenticate. The following diagram depicts the authentication model.

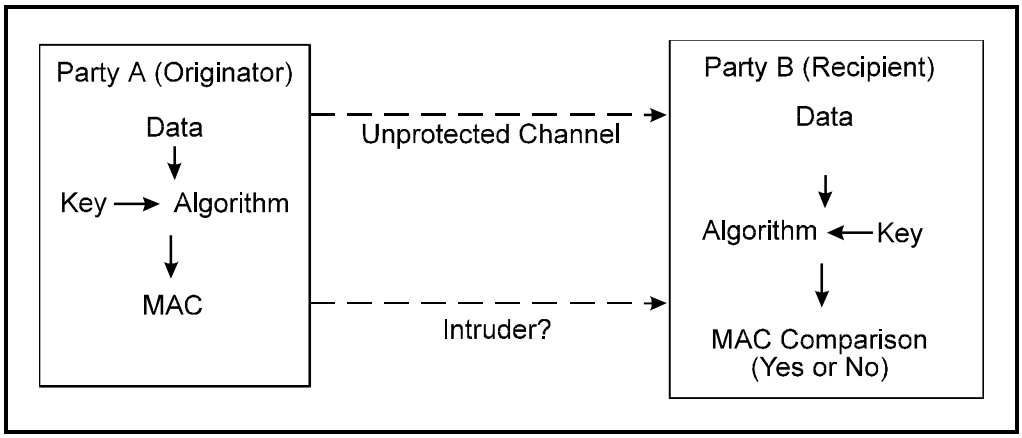

Figure 1: Authentication Model

This model shows the computation of the MAC by the security originator (Party A). The computed MAC is appended to the message before transmission to the security recipient (Party B). A similar computation must be performed by the security recipient; the computed MAC must be compared with the received MAC to verify the security originator and to detect changes in the message. If the MACs compare successfully, Party B knows that the message has not been modified (providing data integrity), and knows that the message was authenticated by the security originator (providing verification of the security originator to the security recipient).

4.3 Encryption

The confidentiality of EDI encoded data is protected by the use of encryption. This standard provides a method for protecting the confidentiality of ASC X12 formatted data that is independent of the actual structure and the data content. In all cases, the structures defined in this standard apply.

Use of the encryption option of this standard does not provide detection of alteration of messages between the security originator and the security recipient. If encryption were used without authentication or assurance and an error occurs in transmission, the error may not be detected and the decrypted message may not be usable nor able to be processed. Authentication or assurance can be used to provide integrity while using encryption to provide confidentiality of the same data. When both authentication or assurances and encryption are used, the authentication or assurances are performed before encryption of the original plaintext data.

4.3.1 Assurances

Assurances, as defined in this standard, are a generalization of the concept of “electronic signature.” The assurance originator expresses “business intent” by adding an assurance (one SxA segment and one SVA segment) to a transaction set or functional group that codifies the intent and computes a MAC or digital signature based on the text or the secured digest of the transaction set or group. The assurance recipient verifies the digital signature. Since the assurance originator is the only entity that could have generated the digital signature in a public key system, the assurance recipient can have certainty that expressed business intent is valid. The assurance provides for:

- Encoding of "business intent"
- Any text (e.g., signer's name, license number, etc.) desired by the signator to accompany the signature
- Date/time stamp
- Non-reputable signature
- Dis-embodied signature

4.3.2 Non-repudiation

Using a private-key/public-key approach, non-repudiation of origin of a message can be established since only the assurance originator can generate the signature. Any recipient that has access to the technology and that can verify the certificate chain, is able to verify the signature.

Non-repudiation of receipt is also supported by this standard. Using a private-key/public-key approach, a message recipient can generate an electronic signature value that will assure the message originator that the message was received by the intended party, without modification.

Non-repudiation is achieved since the process provides irrevocable proof, to a third party, that the security originator has access to the private key component. The strength of public key technology relies on the mathematical insolubility of the reverse process.

4.3.3 Supporting Services

Compression

Data compression takes a contiguous stream of characters and transforms the stream into a series of codes. If the compression algorithm is effective, the resulting stream of codes is smaller than the original text.

The theory of data compression is based on redundancy in the original text. For X12-encoded data, the repetition of segment identifiers in conjunction with separators, the recurrence of alpha and numerical substrings, and the frequency of common codes -- all add up to considerable redundancy in the ASCII encoding stream.

The various common data compression algorithms provide different coding methods for single and ultiple character combinations that occur with different probabilities.

Modern compression techniques fall into several families: Shannon-Fano coding, Huffman coding, statistical modeling, and methods based on an adaptive dictionary. These techniques are often used for compressing files on bulletin boards or in modems that are able to perform compression/decompression "on the fly." If compression is used, it must be completed before the encryption process.

Security Error Reporting

While security products and algorithms provide a reliable and stable output, errors can occur. It is imperative that any security-related error be detected and reported immediately. A series of error codes and conditions have been defined to support this reporting requirement. It is possible to communicate these codes in the ASC X12 Functional Acknowledgment (997); however, it is recommended that the ASC X12 Secured Receipt or Acknowledgment (993) be the primary method for reporting security protocol errors.

4.3.4 Filtering of Encrypted or Compressed Data

The result of the encryption process is a pseudo-random binary string. Some communications systems will interpret some of the pseudo-random binary output as control characters or control sequences which may have a detrimental effect on transmission services (e.g., spurious XON/XOFF or EOF characters).

Filtering is a reversible process that converts the binary output from the encryption process to a coded character set output that does not contain control characters or sequences. The receiving process uses the inverse filtering process to convert the data back to the binary data so that it can be decrypted.

Filtering is performed after all other security processes are complete.

This process will normally increase the size of the message. ANSI X9.23 defines several optional filters that provide a tradeoff between character set flexibility and message expansion:

- The hexadecimal filter converts each byte of binary data into two hexadecimal characters (0..9, A..F) expressed in the character set of transmission. This filter works well with nearly all character sets, but it produces an expansion of 100%.
- The ASCII filter converts the binary data to a string of ASCII printable characters and normally produces an expansion of about 23%.
- The ASCII/Baudot filter converts the binary data to a string of characters belonging to both the ASCII and Baudot character sets and normally produces an expansion of about 86%.

This standard also allows the use of no filter or the use of a filter specified by mutual agreement among the options provided by the code set maintained by ASC X12.

4.4 Sequencing of Cryptographic Techniques

In practical situations, the users of this standard will choose combinations of features rather than just a single feature. The supported combinations of features, not including assurances, are specified in the codes for DE 990 - Security Type. All features are designed to be used in isolation or in any combination.

Authentication does not protect the confidentiality of the message because the information is interchanged in its plain text form. Message encryption can be used to provide confidentiality while using authentication to provide integrity protection of the same data. When both authentication and encryption are used, the authentication is performed before encryption of the original plain text data.

Where more than one service is selected at a specific level, the order of processing is:

- Before applying any security services, the data must first be translated into an EDI format
- Addition of one or more assurances

- Authentication
- Compression
- Encryption
- Filtering for data communications

When assurance segments (S4A and SVA) are used, they must be added to unsecured (not authenticated or encrypted) transactions. If a transaction set is received (with or without assurances) with encryption and/or authentication applied by the originator, the transaction set must be either decrypted or authenticated prior to the addition of any further assurances. Once any assurances have been added, the transaction set can be encrypted or authenticated prior to being forwarded to additional parties.

When applying security services at the functional group level, all security services at the transaction set level must be completed before applying security services at the functional group level.

The receiving organization processes the received message in the reverse order, starting with inverse filtering, followed by decryption, followed by decompression, validation of authentication and validation of the assurances. When processing inbound security services at the transaction set level, all security services except assurances at the functional group level must be removed before processing inbound security services at the transaction set level. (Note that to maintain the validity of the assurances associated with S3A, S4A assurances must not be removed before verification of the S3A assurances.) In this manner the receiving organization unwraps the EDI message by processing the security services and removing the S3S/S3E pairs from the message before processing the next security service.

5 ASC X12 Security Segment Specifications

5.1 General

Security services (authentication, encryption and assurances) are provided at two levels within ASC X12 in conjunction with the following envelopes:

- Functional Group (GS/GE envelope)
- Transaction Set (ST/SE envelope)

The Version/Release of the Security segments is independent of the Version/Release/Sub-release version identified in the Functional Group Header (GS). The Version/Release for the security structures is contained in the first data element of the S3A, S4A, S3S, & S4S segments. This version/release code indicates the specific version/release of the standard that is applicable to that security segment and the associated security trailer or security value segment. The same transaction set or functional group may have different version/releases of the security standard applied to it or there may be different version/releases of the security standard applied to multiple transaction sets or functional groups.

At each of these levels, authentication, encryption and assurances are each optional. Assurances are independent of authentication or encryption. In addition, any service used at one level is independent of a service used at the other level.

Assurances are applied before any other security processes (authentication, encryption, and compression). Therefore, the assurance segment (S3A or S4A) immediately follows the segment initiating the beginning of this level (GS or ST), and precedes any existing assurance segments (S3A or S4A). The Security Value (SVA) segment follows any existing Security Value (SVA) segments and precedes the segment terminating the level (GE or SE).
If encryption and/or authentication is provided, the security header segment (S3S or S4S) immediately follows the segment initiating the beginning of this level (GS or ST); the security trailer segment (S3E or S4E) precedes the segment terminating the level (GE or SE). If encryption and/or authentication at both

levels is provided and if assurances are used at both levels, the sequence of segments, illustrating these levels, is:

ISA - Interchange Header

GS - Functional Group Header

S3S - Security Header Level 1

S3A - Assurance Header Level 1

ST - Transaction Set Header

S4S - Security Header Level 2

S4A - Assurance Header Level 2

(The Transaction Set Segments)

SVA - Security Value (Level 2)

S4E - Security Trailer Level 2

SE - Transaction Set Trailer

SVA - Security Value (Level 1)

S3E - Security Trailer Level 1

GE - Functional Group Trailer

IEA - Interchange Trailer

There is one exception to the segment placement defined above. The Secured Receipt or Acknowledgment (993) uses a unique sequencing of the assurance segments. The Secured Receipt or Acknowledgment (993) contains assurances for information not contained in the transaction set. The transaction set designers believed that using the S4A/SVA pair standard placement would mislead implementers. Therefore, to provide greater clarity, the S4A/SVA pair were placed together just prior to the SE segment.

5.2 Standard Structures for Security Segments

5.2.1 Segment Constructs in Backus-Naur Form (BNF)

The security related segments are constructed as follows:

S3A - Assurance Header Level 1

```
<function_assurance_header>  ::=    S3A  <gs> <Security_Version/Release_Identifier_Code>
                                         <gs> <Business_Purpose_of_Assurance>
                                         <gs> <Computation_Methods>
                                         <gs> <Domain_of_Computation_of_Assurance_Digest>
                                         <gs> [<Assurance_Originator>]
                                         <gs> [<Assurance_Recipient>]
                                         <gs> [<Assurance_Reference_Number>]
                                         <gs> [<Date/Time_Reference>]
                                         <gs> [<Assurance_Text>]
                                         <gs> [<Certificate_Look-up_Information>]
                                         <gs> [<Assurance_Token_Parameters>]
                                         <tr>

<Security_Version/Release_Identifier_Code>    ::= <id>
<Business_Purpose_of_Assurance>  ::=    <id>
<Computation_Methods>            ::=    <Assurance_Algorithm>
```

```
                                                <us> <Hashing_Algorithm>
<Domain_of_Computation_of_Assurance_Digest> ::= <id>
<Assurance_Originator>              ::=     <string>
<Assurance_Recipient>               ::=     <string>
<Assurance_Reference_Number>        ::=     <string>
<Date/Time_Reference>               ::=     <string>
<Assurance_Text> ::= <string>
<Certificate_Look-up_Information>   ::=     <Look-up_Value_Protocol_Code>
                                                <us> <Filter_ID_Code>
                                                <us> <Version_Identifier>
                                                <us> <Look-up_Value>
                                                <us> [<Look-up_Value_Protocol_Code>]
                                                <us> [<Filter_ID_Code>]
                                                <us> [<Version_Identifier>]
                                                <us> [<Look-up_Value>]
                                                <us> [<Look-up_Value_Protocol_Code>]
                                                <us> [<Filter_ID_Code>]
                                                <us> [<Version_Identifier>]
                                                <us> [<Look-up_Value>]
<Assurance_Token_Parameters>        ::=     <Assurance_Token_Parameter_Code>
                                                <us> <Assurance_Token_Parameter_Value>
                                                <us> [<Assurance_Token_Parameter_Code>]
                                                <us> [<Assurance_Token_Parameter_Value>]
                                                <us> [<Assurance_Token_Parameter_Code>]
                                                <us> [<Assurance_Token_Parameter_Value>]
                                                <us> [<Assurance_Token_Parameter_Code>]
                                                <us> [<Assurance_Token_Parameter_Value>]
                                                <us> [<Assurance_Token_Parameter_Code>]
                                                <us> [<Assurance_Token_Parameter_Value>]
                                                <us> [<Assurance_Token_Parameter_Code>]
                                                <us> [<Assurance_Token_Parameter_Value>]
                                                <us> [<Assurance_Token_Parameter_Code>]
                                                <us> [<Assurance_Token_Parameter_Value>]
                                                <us> [<Assurance_Token_Parameter_Code>]
                                                <us> [<Assurance_Token_Parameter_Value>]
                                                <us> [<Assurance_Token_Parameter_Code>]
                                                <us> [<Assurance_Token_Parameter_Value>]
                                                <us> [<Assurance_Token_Parameter_Code>]
                                                <us> [<Assurance_Token_Parameter_Value>]

<Assurance_Algorithm> ::= <id>
<Hashing_Algorithm> ::= <id>

<Look-up_Value_Protocol_Code> ::= <id>
<Filter_ID_Code> ::= <id>
<Version_Identifier> ::= <string>
<Look-up_Value> ::= <string>

<Assurance_Token_Parameter_Code> ::= <id>
<Assurance_Token_Parameter_Value> ::= <string>
```

S4A - Assurance Header Level 2

```
<trans_assurance_header>      ::=     S4A  <gs> <Security_Version/Release_Identifier_Code>
                                           <gs> <Business_Purpose_of_Assurance>
                                           <gs> <Computation_Methods>
                                           <gs> <Domain_of_Computation_of_Assurance_Digest>
```

```
                                            <gs> [<Assurance_Originator>]
                                            <gs> [<Assurance_Recipient>]
                                            <gs> [<Assurance_Reference_Number>]
                                            <gs> [<Date/Time_Reference>]
                                            <gs> [<Assurance_Text>]
                                            <gs> [<Certificate_Look-up_Information>]
                                            <gs> [<Assurance_Token_Parameters>]
                                            <tr>
<Security_Version/Release_Identifier_Code> ::= <id>
<Business_Purpose_of_Assurance>     ::=     <id>
<Computation_Methods>               ::=     <Assurance_Algorithm>
                                               <us> <Hashing_Algorithm>
<Domain_of_Computation_of_Assurance_Digest> ::= <id>
<Assurance_Originator>              ::=     <string>
<Assurance_Recipient>               ::=     <string>
<Assurance_Reference_Number>        ::=     <string>
<Date/Time_Reference>               ::=     <string>
<Assurance_Text>                    ::=     <string>
<Certificate_Look-up_Information>   ::=     <Look-up_Value_Protocol_Code>
                                               <us> <Filter_ID_Code>
                                               <us> <Version_Identifier>
                                               <us> <Look-up_Value>
                                               <us> [<Look-up_Value_Protocol_Code>]
                                               <us> [<Filter_ID_Code>]
                                               <us> [<Version_Identifier>]
                                               <us> [<Look-up_Value>]
                                               <us> [<Look-up_Value_Protocol_Code>]
                                               <us> [<Filter_ID_Code>]
                                               <us> [<Version_Identifier>]
                                               <us> [<Look-up_Value>]
<Assurance_Token_Parameters>        ::=     <Assurance_Token_Parameter_Code>
                                               <us> <Assurance_Token_Parameter_Value>
                                               <us> [<Assurance_Token_Parameter_Code>]
                                               <us> [<Assurance_Token_Parameter_Value>]
                                               <us> [<Assurance_Token_Parameter_Code>]
                                               <us> [<Assurance_Token_Parameter_Value>]
                                               <us> [<Assurance_Token_Parameter_Code>]
                                               <us> [<Assurance_Token_Parameter_Value>]
                                               <us> [<Assurance_Token_Parameter_Code>]
                                               <us> [<Assurance_Token_Parameter_Value>]
                                               <us> [<Assurance_Token_Parameter_Code>]
                                               <us> [<Assurance_Token_Parameter_Value>]
                                               <us> [<Assurance_Token_Parameter_Code>]
                                               <us> [<Assurance_Token_Parameter_Value>]
                                               <us> [<Assurance_Token_Parameter_Code>]
                                               <us> [<Assurance_Token_Parameter_Value>]
                                               <us> [<Assurance_Token_Parameter_Code>]
                                               <us> [<Assurance_Token_Parameter_Value>]
                                               <us> [<Assurance_Token_Parameter_Code>]
                                               <us> [<Assurance_Token_Parameter_Value>]
<Assurance_Algorithm> ::= <id>
<Hashing_Algorithm> ::= <id>

<Look-up_Value_Protocol_Code> ::= <id>
<Filter_ID_Code> ::= <id>
<Version_Identifier> ::= <string>
<Look-up_Value> ::= <string>
```

<Assurance_Token_Parameter_Code> ::= <id>
<Assurance_Token_Parameter_Value> ::= <string>

SVA - Security Value

<security_value> ::= SVA <gs> <Filter_ID_Code>
<gs> <Version_Identifier>
<gs> <Security Value>
<tr>

<Filter_ID_Code> ::= <id>
<Version_Identifier> ::= <string>
<Security_Value> ::= <Security_Value_Qualifier>
<us> <Encoded_Security_Value>

<Security_Value_Qualifier> ::= <id>
<Encoded_Security_Value> ::= <string>

S3S - Security Header Level 1

<function_security_header> ::= S3S <gs> <Security_Version/Release_Identifier_Code>
<gs> <Security_Type>
<gs> <Security_Originator_Name>
<gs> [<Security_Recipient_Name>]
<gs> [<Authentication_Key_Name>]
<gs> [<Authentication_Service_Code>]
<gs> [<Certificate_Look-up_Information>]
<gs> [<Encryption_Key_Information>]
<gs> [<Encryption_Service_Information>]
<gs> [<Length_of_Data>]
<gs> [<Transformed_Data >]
<tr>

<Security_Version/Release_Identifier_Code> ::= <id>
<Security_Type> ::= <id>
<Security_Originator_Name> ::= <string>
<Security_Recipient_Name> ::= <string>
<Authentication_Key_Name> ::= <string>
<Authentication_Service_Code> ::= <id>
<Certificate_Look-up_Information> ::=<Look-up_Value_Protocol_Code>
<us> <Filter_ID_Code>
<us> <Version_Identifier>
<us> <Look-up_Value>
<us> [<Look-up_Value_Protocol_Code>]
<us> [<Filter_ID_Code>]
<us> [<Version_Identifier>]
<us> [<Look-up_Value>]
<us> [<Look-up_Value_Protocol_Code>]

<us> [<Filter_ID_Code>]
<us> [<Version_Identifier>]
<us> [<Look-up_Value>]
<Encryption_Key_Information> ::= <Encryption_Key_Name>
<us> [<Protocol_ID>]
<us> [<Keying_Material>]
<us> [<One-time_Encryption_Key>]
<Encryption_Service_Information> ::= <Encryption_Service_Code>
<us> [<Algorithm_ID>]

<us> [<Algorithm_Mode_of_Operation>]
<us> [<Filter_ID_Code>]
<us> [<Version_Identifier>]
<us> [<Compression_ID>]
<us> [<Version_Identifier>]

<Length_of_Data> ::= <numeric>
<Transformed_Data> ::= <string>

<Look-up_Value_Protocol_Code> ::= <id>
<Filter_ID_Code> ::= <id>
<Version_Identifier> ::= <string>
<Look-up_Value> ::= <string>

<Encryption_Key_Name> ::= <string>
<Protocol_ID> ::= <id>
<Keying_Material> ::= <string>
<One-time_Encryption_Key> ::= <string>

<Encryption_Service_Code> ::= <id>
<Algorithm_ID> ::= <id>
<Algorithm_Mode_of_Operation> ::= <id>
<Filter_ID_Code> ::= <id>
<Version_Identifier> ::= <string>
<Compression_ID> ::= <id>
<Version_Identifier> ::= <string>

S3E - Security Trailer Level 1

<function_security_trailer> ::= S3E <gs> <Hash_or_Authentication_Code>
<tr>

<Hash_or_Authentication_Code> ::= <string>

S4S - Security Header Level 2

<trans_security_header> ::= S4S <gs> <Security_Version/Release_Identifier_Code>
<gs> <Security_Type>
<gs> <Security_Originator_Name>
<gs> [<Security_Recipient_Name>]
<gs> [<Authentication_Key_Name>]
<gs> [<Authentication_Service_Code>]
<gs> [<Certificate_Look-up_Information>]
<gs> [<Encryption_Key_Information>]
<gs> [<Encryption_Service_Information>]
<gs> [<Length_of_Data>]
<gs> [<Transformed_Data>]
<tr>

<Security_Version/Release_Identifier_Code> ::= <id>
<Security_Type> ::= <id>
<Security_Originator_Name> ::= <string>
<Security_Recipient_Name> ::= <string>
<Authentication_Key_Name> ::= <string>
<Authentication_Service_Code> ::= <id>
<Certificate_Look-up_Information> ::= <Look-up_Value_Protocol_Code>
<us> <Filter_ID_Code>
<us> <Version_Identifier>

```
                                        <us> <Look-up_Value>
                                        <us> [<Look-up_Value_Protocol_Code>]
                                        <us> [<Filter_ID_Code>]
                                        <us> [<Version_Identifier>]
                                        <us> [<Look-up_Value>]
                                        <us> [<Look-up_Value_Protocol_Code>]
                                        <us> [<Filter_ID_Code>]
                                        <us> [<Version_Identifier>]
                                        <us> [<Look-up_Value>]
<Encryption_Key_Information>      ::=   <Encryption_Key_Name>
                                        <us> [<Protocol_ID>]
                                        <us> [<Keying_Material>]
                                        <us> [<One-time_Encryption_Key>]
<Encryption_Service_Information>  ::=   <Encryption_Service_Code>
                                        <us> [<Algorithm_ID>]
                                        <us> [<Algorithm_Mode_of_Operation>]
                                        <us> [<Filter_ID_Code>]
                                        <us> [<Version_Identifier>]
                                        <us> [<Compression_ID>]
                                        <us> [<Version_Identifier>]

<Length_of_Data> ::= <numeric>
<Transformed_Data> ::= <string>

<Look-up_Value_Protocol_Code> ::= <id>
<Filter_ID_Code> ::= <id>
<Version_Identifier> ::= <string>
<Look-up_Value> ::= <string>

<Encryption_Key_Name> ::= <string>
<Protocol_ID> ::= <id>
<Keying_Material> ::= <string>
<One-time_Encryption_Key> ::= <string>

<Encryption_Service_Code> ::= <id>
<Algorithm_ID> ::= <id>
<Algorithm_Mode_of_Operation> ::= <id>
<Filter_ID_Code> ::= <id>
<Version_Identifier> ::= <string>
<Compression_ID> ::= <id>
<Version_Identifier> ::= <string>
```

S4E - Security Trailer Level 2

```
<trans_security_trailer> ::=    S4E  <gs> <Hash_or_Authentication_Code>
                                     <tr>

<Hash_or_Authentication_Code> ::= <string>
```

5.2.2 General

In this Section, the principles of the standard structures that are implemented at the various levels are established. Thus, in the description of this Section, the generic "SxS", "SxE", and "SxA" segments are referenced. Specifications for the interchange segments are provided in the segment directory.

Security at each potential level is optional and is independent of security at other levels.

The security header (SxS, x = 3 or 4), trailer (SxE), and assurance (SxA) segments are defined for the two levels (functional group and transaction set) using identical formats differing only in the naming of the segments and the reference designators.

The same key shall not be used for more than one function (i.e., authentication or encryption) at any one level. A key used to provide protection at one level shall not intentionally be used at the other level for the same or a different purpose.

When used, the security header segment (generic SxS) is placed immediately after the first segment of the authenticated group of segments. The security header segment identifies which security services are to be used and identifies authentication and encryption keys to be used. The format of the SxS can be found in the segment dictionary.

If the SxS segment is used at the beginning of a functional group or transaction set, the security trailer segment (SxE) precedes the last segment of the same functional group or transaction set.

The security trailer segment is used to end the authentication and encryption process initiated by the corresponding security header segment. SxE01 is the message authentication code (MAC) or hash generated by applying the authentication algorithm against the designated segments.

If one or more assurances are used at either level, one SxA segment and one SVA segment are added. The first Assurance (SxA) and any subsequent Assurances are placed immediately after the first segment of the assured group of segments (GS or ST) and before any previous SxAs.

If the SxA segment is used at the beginning of a functional group or transaction set, the Security Value (SVA) will immediately precede the last segment of the same functional group or transaction set. The SVA for any additional assurances will follow any previous SVAs and precede the last segment of the group or transaction set (GE or SE).

The preparation of ASC X12 text and formatting of security header and trailer segments (see Section 5.1) consists of:

(1) Determining whether a transaction set or functional group will be authenticated, encrypted or assured.

(2) Inserting an SxA segment after the first segment of the group or transaction set (GS or ST) that will be assured and before the SxA of the last assurance at the same level, if present.

(3) Inserting an SVA segment before the last segment of the group or transaction set (GE or SE) and after the SVA of the last assurance at the same level, if present.

(4) Inserting the SxS segment after the first segment of the group of transaction set (GS or ST) that will be authenticated or encrypted and before any Assurances (SxA) if present. The SxE segment is inserted after any Security Values (SVA), if present, and before the last segment of the group or transaction set (GE or SE).

When the S4S and S4E segments are added, the segment count in the SE segment is increased by 2. When the S4A and SVA segments are added, the segment count in the SE segment is increased by 2. When S3S, S3E, S3A or SVA segments are added to the functional group, the count of transaction sets in the group is not changed.

The scope of the authentication process is defined as follows:

- Authentication begins with the first character of the segments designated for authentication (i.e., the first Character of the segment preceding the "SxS" segment).
- If encryption is also to be performed, the authentication process is performed through the SxS09 element and then continues with the segment terminator of the SxS segment (i.e., as if the Length of Data segment, the Initialization Vector element, and their preceding segment separators were not present).
- Authentication ends with and includes the data element separator preceding the hash code.

The scope of the compression process is defined as follows:

- Compression starts with the first character of the segment that follows the SxS segment.
- Compression ends with the last character of the segment that precedes the SxE segment (and does not include the segment separator).

The scope of the encryption process is defined as follows:

- Encryption begins with the Initialization Vector field (SxS11).
- The Initialization Vector (IV) is encrypted using Electronic Code Book (ECB). A new IV shall be used for each message and shall not be intentionally reused. Text following the IV is encrypted using the mode specified in the Encryption Service Code (SxS09).
- Encryption ends with the last character before the segment terminator of the segment preceding SxE.
- Filtering is applied to the encrypted IV and, optionally, to the encrypted data.
- When present, the Length of Data data element counts the number of octets in the encrypted and filtered IV and data.
- When certain block ciphers (i.e., CBC mode of DES) are used for encryption, the plaintext data is padded to a multiple of the block size octets by adding text consisting of 0 to 7 ASCII-zeroes followed by a length value consisting of one ASCII character with value ranging from "1" to "8" that specifies the number of octets used in the padding. This padding meets the requirements of ANSI X9.23 and is completely deterministic. Because of the padding, the pre-encrypted text will always be a multiple of the block size prior to filtering and thus fit any block-oriented encryption mode of operation. The padding is added immediately prior to encryption and is removed by the decryption process; it is not included in the calculation of the hash that is computed prior to encryption.

The scope of the assurance process is defined as follows:

- The scope of each assurance is defined for that assurance by the contents of the Domain of Computation of Assurance Digest (SxA04) data element in the SxA segment.

5.3 Functional Group Security (GS/GE, Level 1)

Assurances are applied before any other security processes (authentication, encryption, and compression). Therefore, the assurance segment (S3A) immediately follows the segment initiating the beginning of this level (GS), and precedes any existing assurance segments (S3A). The Security Value (SVA) segment follows any existing Security Value (SVA) segments and precedes the segment terminating the level (GE).

Encryption and/or authentication is available for individual functional groups through the use of the S3S and S3E segments. Both the S3S and S3E segments can be present or absent for any functional group.

Security (authentication and/or encryption) at this level is optional and is independent of security at any other level.

If functional group level security is used, the key used for authentication and the key used for encryption shall be different, and the keys shall not be used for any other security purpose. Separate functional groups, within the same interchange or in separate interchanges, may use the authentication key or the encryption key used for the same purpose in other functional groups.

Definitions of security segments and data elements are given in the segment and data dictionaries.

The placement of security segments is defined below.

- The S3A segment immediately follows the GS segment.
- The S3S segment immediately follows the GS segment and is inserted immediately before any S3A or S3S segments, if present.
- The S3E segment precedes the GE segment and is inserted immediately after any SVA or S3E segments, if present.

The scope of security segments is defined as given below.

- Authentication begins with the first character of the functional group (the "G" of GS*.....).
- Authentication ends with and includes the data element separator preceding the hash code.
- Compression begins with the first character of the segment following the S3S segment (i.e., the first character of the first enclosed transaction set).
- Compression ends with the last character before the segment terminator of the segment preceding the S3E.
- Encryption begins with the first character of the Initialization Vector (S3S11).
- Encryption ends with the last character before the segment terminator of the segment preceding S3E.
- Assurance segments have a scope defined by the code in S3A04.

5.4 Transaction Set Security (ST/SE, Level 2)

Assurances are applied before any other security processes (authentication, encryption, and compression). Therefore, the assurance segment (S4A) immediately follows the segment initiating the beginning of this level (ST), and precedes any existing assurance segments (S4A). The Security Value (SVA) segment follows any existing Security Value (SVA) segments and precedes the segment terminating the level (SE).

Encryption and/or authentication is available for individual transaction sets through the use of the S4S and S4E segments. Both the S4S and S4E segments can be present or absent for any transaction set. Security (authentication and/or encryption) at this level is optional and is independent of security at any other level.

If transaction set level security is used, the key used for authentication and the key used for encryption shall be different, and the keys shall not be used for any other security purpose. Separate transaction sets, within the same functional group or in separate functional groups or interchanges, may use the authentication key or the encryption key used for the same purpose in other transaction sets. Definitions of security segments and data elements are contained in the segment and data element dictionaries.

The placement of security segments is defined below.

- The S4A segment immediately follows the ST segment, except when used in the Secured Receipt or Acknowledgment (993). In the Secured Receipt or Acknowledgment (993), the S4A will immediately precede the SVA segment.
- The S4S segment immediately follows the ST segment and is inserted immediately before any S4A or S4S segments, if present.
- The S4E segment precedes the SE segment and is inserted immediately after any SVA or S4E segments, if present.

The scope of security segments is defined as given below.

- Authentication begins with the first character of the transaction set (the "S" of ST*.....).
- Authentication ends with and includes the data element separator preceding the hash code.
- Compression begins with the first character of the segment following the S4S segment (i.e., the first character of the transaction set body).
- Compression ends with the last character before the segment terminator of the segment preceding the S4E.
- Encryption begins with the first character of the Initialization Vector (S4S11).
- Encryption ends with and includes the last character before the segment terminator of the segment preceding S4E.
- Assurance segments have a scope defined by the code in S4A04.

6 Auditing and Responding to Secured ASC X12 Data

Since the security structures are embedded in the ASC X12 structures, the responses will be found embedded in the response to a group of transactions, the Functional Acknowledgment Transaction Set (997).

These security structures are an integral part of the management of the ASC X12 EDI environment. When an audit journal is required, it may be required for a variety of EDI management purposes. This Section also attempts to reconcile these disparate purposes.

Only a single 997 response is allowed for a functional group, and it must report on the syntactic integrity of the entire received group of transaction sets.

Appendix A

(This Appendix is not part of the standard but is included for information only.)

The example shows authentication and the use of the assurance segment -- with authentication applied at both levels and multiple assurances segments ("signatures") on the first transaction at transaction set level only. The security recipient for the transaction set (shown in the S4S segment) in this example is different from the security recipient of the functional group (shown in the S3S segment). The original transaction was signed by the original preparer of the ASC X12 message (code "PRP" indicates "preparer"), but the second signature was applied by a subsequent handling party (code "ASG" indicates authorizer).

The integrity indicated by the functional group hash would have been calculated after the application of this second signature.

NOTE: Cryptographic values are for explanation purposes only and do not represent real output of the stated cryptographic algorithms.

GS*IN*012345678*087654321*960214*2210*000001*X*003050~
S3S*003072*AA*SEC_ORIG*SEC_RECV*KEY_ONE*1~
ST*810*0001~
S4S*003072*AA*SEC_ORIG*SEC_RECV_2*KEY_TWO*1 ~
S4A*003072*ASG*RSA:MD5*E*AUTH ORG*AUTH RCV**199602040932782-
0900*CHARLESSMITH/A=1234567/F=987623123492*CI:3456789*A9012B3AA1231D3F1343860AF0C99106E45B
10FA ~
S4A*003072*PRP*RSA:MD5*E*AUTH ORG*AUTH RCV**199602031623EST-0600*JOHN
JONES*CI:12345678:KN:ABC923456*AB40C0DFF174AF90CAAD299CA478FFB45611A012 ~
BIG*960213*101*960215*P996320 ~
N1*BT*ACME DISTRIBUTING COMPANY ~
N3*P.O. BOX 333555 ~
N4*ANYTOWN*NY*12345 ~
IT1*01*3*JR*53.75*WE ~
PER*1C*J. DOE*TE*2125551234 ~
CAD*M****CONSOLIDATED ~
TDS*5111 ~

SVA*HDC*1.0*ASV:F5A9012B3AA1231D3F1343AA59D299CA478FFB456112B3AA1231D3F1343AA51232343453
45FFACB98456FAB4345769C2AB452452ABD4567DF67A ~

SVA*HDC*1.0*ASV:3134FA3918CB2A8586AFDF45690A9B3C120FD45860AF0C99AB40C0DFF174AF90CAAD299
CA478FFB45611A912106E45B10FA55678CB98A300A6BAE ~

S4E*AF57 B3AC ~
SE*16*0001 ~

ST*810*0002 ~
S4S*003072*AA*SEC_ORIG*SEC_RECV_2*KEY_TWO*1 ~
BIG*960213*101*960216*P996321 ~
N1*BT*ACME DISTRIBUTING COMPANY ~
N3*P.O. BOX 333555 ~
N4*ANYTOWN*NY*12345 ~
IT1*01*3*JR*53.75*WE ~
PER*1C*J. DOE*TE*2125551234 ~
CAD*M****CONSOLIDATED ~
TDS*5111 ~
S4E*5AC1 1834 ~
SE*12 ~
S3E*F739 2A55 ~
GE*2*000001~

Many other combinations are possible, including authentication at one level (S3S or S4S) and other combinations of the same or different security originator or security recipient names.

The actual MACs generated by the security originator are dependent on the values of the cryptographic keys corresponding to the key names KEY_ONE and KEY_TWO. The values of the cryptographic keys are known only to the security originator and the security recipient. Only the names of the keys are sent in the above interchange.

If encryption in the CBC mode was applied to the second transaction set in the example above in addition to authentication, the S4S segment would have the following appended to it immediately prior to encryption:

*KEY_THREE*20*192*8 octets of IV

and the TDS segment in the encrypted form only would be:

TDS*511100000008

where "00000008" is the padding (seven ASCII zeroes followed by a total length of padding of eight). This padding is removed after the decryption process.

Following encryption, the length of the ciphertext data is placed in the LOD field. For this example, if filtration is not used, the value placed in the LOD field is "192" consisting of the sum of the following:

- Length of encrypted IV (always eight)
- Length of plain text segments including segment terminator character (176). The count includes the segment terminator of the S4S segment (encrypted) but not the segment terminator of the TDS segment (not encrypted).
- Length of padding (eight, in this example)

If filtration is used, the LOD is the length of the filtered, encrypted text and thus not necessarily a multiple of eight octets. The hash at the transaction set level (in S4S01) is not affected by the encryption of the transaction set. The hash at the functional group level (in S3S01) would be affected since the computation is performed on the transaction set as enveloped, i.e., as both authenticated and encrypted.

X12.59 IMPLEMENTATION OF EDI STRUCTURES—SEMANTIC IMPACT

1 Purpose and Scope

While the X12 standards provide a clear distinction between the syntax of X12 structures and the semantics of transaction set usage, there are cases where the relative positioning of segments within those structures provides semantic information in their implementation.

The purpose of this standard is to describe the semantic relationships inherent in the implementation of those X12 structures:

- The meaning that is to be associated with data due to their positioning within the exchange of X12 information, and
- The data relationships that can be inferred from the data structure.

The clearest description of this issue arises from the definition of each individual transaction set. Syntax and semantic notes provide *explicit* rules to be followed in the creation (for the sender) and decoding (for the receiver) of any use (instance) of that transaction set.

This standard provides the rules inherent in the structure of X12 interchanges that govern such creation and decoding. These rules shall be considered part of the standard, to be followed in using a specific transaction set, unless superseded by syntax or semantic notes for that transaction set. Such rules, if any, that apply to structures at levels higher or lower than the transaction set level will be provided as well.

2 Referenced and Related Standards

The X12 series of standards on Electronic Data Interchange are interdependent. The following standards are required to interpret, understand, and use this standard:

X12.3	Data Element Dictionary
X12.5	Interchange Control Structures
X12.6	Application Control Structure
X12.22	Segment Directory

The following guideline is also referenced in this standard:

ASC X12 Model of X12 Functions Guideline

3 Terms and Definitions

X12.5 Interchange Control Structures provides formal definition of terminology related to the interchange of information formatted as Electronic Data Interchange, as developed by ASC X12. X12.6 Application Control Structure provides formal definition of terminology related to the Electronic Data Interchange standards developed by ASC X12.

The following are terms and definitions for Semantic Structures:

ADDRESSING
"Addressing" is a term used to cover both the naming and semantic address of the logical or physical location of a person or position.

ADJACENT LEVEL
The term “adjacent” is used in a hierarchical sense. Two levels are adjacent to one another by reason of their relative positioning in the hierarchy of levels; one is at an immediately adjacent level to the other within the hierarchy.

AREA
An area is a distinct section of an X12 transaction set definition. The areas of a transaction set are “heading,” “detail,” and “summary.”

BELONGING
The term “belonging” is used to identify the relationship of the data contents of a child structure to its parent structure. For example, line items are said to belong to an order.

CHILD
A child is a semantic entity that has meaning, in a transaction set instance, only through the parent structure to which it belongs. For example, an order line item has relevance only in the context of a total order.

DETAIL AREA
A detail area is any looping structure between the heading area and summary area of a transaction set. Detail areas encompass the actual body of the business transaction.

FUNCTIONAL EQUIVALENCE
Functional equivalence is the determination that the data contents of a segment or loop at one semantic level are meant to convey the same semantic intent as the data contents of a segment or loop at an adjacent level. For example, a line item ship-to address is functionally equivalent to a ship-to address in the order heading area. See the definition of PRECEDENCE below.

HEADING AREA
The heading area is an area at the beginning of a transaction set containing information pertaining to the entire transaction set.

INTERCHANGE
See X12.5 Interchange Control Structures.

NAMING
X12 uses naming structures to allow for the identification of a person or position. See ADDRESSING.

ORDER INDEPENDENCE
Order independence denotes a lack of any semantic significance in the order within the same semantic entity.

PARENT
A parent is a semantic entity that has a scope of applicability over at least one other semantic entity. For example, the data contents of the heading area of a purchase order are valid for all information in the order line items, barring precedence override.

PRECEDENCE
The term “precedence” describes a relationship among semantic levels such that, in the case of conflict between levels, the data contents at one level override (replace) the data contents of the other level. For example, if a line item ship-to address has precedence over a ship-to address in the heading area, it overrides the heading area address.

SEMANTIC ENTITY
A semantic entity is a segment or set of segments that, when taken together, convey meaning with the same scope of applicability. For example, the segments of a heading area of a purchase order provide the data contents that apply over the whole of that purchase order. Similarly, the scope of applicability of a GS segment is over the semantic entities included after it, up to and including its corresponding GE segment.

SEMANTIC ENTITY DEFINITION
The structure of a semantic entity, as published in a standards document, is its definition. The definition includes the sequence of data segments, their requirement for inclusion in any semantic entity instance, their maximum repeat counts, their placement within loops, and the requirement and repeat characteristics of such distinct loops. Segments may also be grouped in distinct areas (heading, detail, summary) to impart additional semantic significance.

SEMANTIC ENTITY INSTANCE
The data contents of a semantic entity, formatted in accordance with the rules of the semantic entity definition, comprises an instance of the semantic entity.

SEMANTIC LEVEL
The identification of semantic level is a designation allowing the description of a hierarchical relationship among semantic entities.

SEMANTICS
The term “semantics” typically pertains to the meaning of the information contained within a data structure, independent of the structure itself, such that the conversion from one structure to another would not change the meaning of the data. In this paper, the discussion focuses on what meaning can be inferred by the placement of information in the structure of EDI data.

SUMMARY AREA
The summary area is an area at the end of a transaction set containing information that addresses the results of summarizations of information in the detail area.

X12 STRUCTURE
An X12 structure is defined both in terms of its design (see *SEMANTIC ENTITY DEFINITION*) and its use (see *SEMANTIC ENTITY INSTANCE*). The design of an X12 structure is the sequencing of data elements within segments, and then, if applicable, the combination of segments in a sequence to make up a semantic entity. A use of an X12 structure is the formatting of specific data according to the structure's design, to convey a specific meaning.

4 Introduction

4.1 Syntax and Semantics in EDI Implementations

The terms “syntax” and “semantics,” when used in the context of EDI implementations, refer to the structure and meaning of X12-formatted information, respectively.

- *Syntax is the structure of the data.* This includes establishing the method of encoding a piece of data by its attributes and identifying that data in the transfer, and is provided by the following aspects of the following X12 standards:
  - X12.6 Application Control Structure describes the basic structure of the components of an X12 functional group and provides the control segments used to bound loops of data segments, transaction sets, and groups of related transaction sets, and the special control segments used to bound transaction sets and groups of transaction sets for security purposes.
  - X12.3 Data Element Dictionary provides the attributes of type, minimum and maximum length, and if applicable, the relevant code list, of each data element that can be used in an X12 segment.

  - X12.22 Segment Directory provides the segment label, data elements within the segment (in sequence), requirements for use of each element, and syntax notes indicating relationships among data elements, if any.
  - Management transaction sets provide both the format and data content of transactions related to the interoperation of X12 systems. For example, the Functional Acknowledgment Transaction Set (997) describes the syntax-level acknowledgment of the receipt of an X12 functional group. Also, the Electronic Form Structure Transaction Set (868) describes the transfer of the EDI standards, or portions thereof, in electronic form.
  - All other transaction sets (defined as "business application" transaction sets) provide the structure of the segments to be used within each, including a heading area (Table 1, unless otherwise identified in the transaction set definition), detail area (Table 2), summary area (Table 3), looping structures within those areas (and nested within each other), and the segment sequence within the transaction set. In addition, each segment and loop is further defined in terms of requirement for use and maximum use.

- The term *"semantics"* relates to the meaning of the data transferred to the business application. The meaning of X12 data is derived from the descriptions of the components of the standard used in a specific implementation and is provided by the following aspects of the following X12 standards:
  - X12.3 Data Element Dictionary provides the name and definition of each data element that can be used in an X12 segment, and if a code list is included, the description of each code. For ID data elements where the code list is maintained external to the X12 Secretariat, that external source provides the meaning of each code in the list.
  - X12.22 Segment Directory provides the name and definition of each segment that can be used in an X12 transaction set, as well as any semantic notes indicating relationships in the meaning of the use of one or more data elements in any one instance of the segment.
  - Business application transaction sets provide semantics in the name, description of purpose and scope, and transaction set notes and comments.
- *Additional control:* There are additional standardized components of the structure of X12 that control the distribution of X12 information through the enveloping of X12-formatted information, including:
  - X12.58 Security Structures defines the structures used for authentication and encryption of X12 transaction sets and functional groups.
  - X12.5 Interchange Control Structures provides the control structures for the electronic interchange of one or more functional groups.
  - X12.56 Interconnect Mailbag Control Structures defines the control segments used to bound a mailbag containing one or more X12 interchanges.

Taken together, the control components of X12 provide the structure in which the semantics of X12 can be exchanged between trading partners. The question to be addressed in this standard is whether the structure itself, by reason of the positioning of control information, provides any semantic content in the exchange of X12-formatted information.

The positioning of X12 control information is in terms of hierarchical levels of control, which will be described in detail in Section 5. For the purposes of highlighting the focus of this paper on the semantic content of any information provided at a particular control level, these levels will be referred to as "semantic levels." Of course, these semantic levels correspond to the hierarchy of the control levels.

4.2 Issues of Semantic Impact in X12 Syntax

This standard, therefore, will address the semantics implied by positioning within an EDI interchange. The document will address such issues as:

- Scope of Applicability
  For example, does the information content present in the heading area of a transaction set have any applicability to the detail area of the transaction set? (See example 2 in Section 6.1.)

- Precedence of Areas
  Assuming the answer to the previous question is “yes,” how should a transaction set receiver treat the appearance of content in the detail area that is similar to content in the heading area? How can the receiver determine that such a “similarity” is intended? Does such information replace or add to the information content in the heading area? (See Section 6.3.)
- Intra-interchange and Intra-group Relationships
  Can a sender imply any relationship among transaction sets within the same functional group, or groups within the same interchange?(See Sections 5.1 and 5.2 for descriptive information, and example 1 in Section 6.2 for a review of the topic.)
- Order of Segments
  Is there any meaning in the order of segments within an area? Within a loop? (See example 3 in Section 6.2.)
- Loops
  What is the relationship among segments contained in an instance of a loop? (See example 2 in Section 6.2.)
- The HL Segment
  What is the meaning in the use of an HL segment to begin a loop? (See Section 5.6 for a description of the handling of loops beginning with the HL segment.)
- Routing
  What is the relationship, if any, between addressing information in the ISA and the GS? (See Section 6.4.)

This standard begins with definitions of the concepts and semantic relationships under discussion, followed by detailed descriptions of the rules governing those relationships. Examples are provided where applicable.

5 Semantic Levels of an EDI Interchange

The purpose in this section is to introduce the terms that are relevant to any discussion of the structure of an EDI interchange. A particular interest is to present these terms in such a way that the semantic relationship to their adjacent levels may be defined as well.

Items preceded by an asterisk under each section are the rules for the use of particular semantic levels that would have impact on their relationship to adjacent levels.

5.1 Interchanges Within a Logical Communications Exchange

Logical Communications Exchange. A logical communications exchange is any environment in which EDI information can be exchanged between trading partners in electronic format.
Interchanges. Each interchange between EDI trading partners occurs during a logical communications exchange. Each exchange includes the transfer of one or more EDI interchanges, possibly in both directions.

Each instance of a logical communications exchange shall be considered to be an external, enveloping structure of the interchanges that are transferred within that session. However, the semantic relationship between the interchanges and the logical communications exchange of which they are a part is beyond the scope of this standard.

5.2 Groups within an Interchange

Interchange Control Segments. Each interchange between EDI trading partners may contain interchange-level control segments related to prior interchanges. The only segment defined as an interchange control segment, which can be included within the interchange header and trailer, is the TA1, which references the acceptance or rejection of a prior envelope.

Functional Groups. Each interchange between EDI trading partners may contain functional groups. Each instance of a functional group applies to a specific business function, defined by the specific application to which it applies.

Each instance of an interchange shall be considered the parent structure of the interchange control segments and functional groups that are enclosed within the boundaries of that interchange.
An interchange control segment is the parent structure of the data elements defined to be part of that segment.

5.3 Transaction Sets Within a Group

Transaction Set. Each functional group exchanged between EDI trading partners will contain one or more transaction sets. Each instance of a transaction set will identify a separate business transaction within the business function to which that group applies.

Each instance of a functional group shall be considered the parent structure of the transaction sets that are enclosed within the boundaries of that functional group.

5.4 Heading/Summary Areas of a Transaction Set

Areas. Most EDI transaction sets are designed so as to be divided into discrete “areas,” which are typically identified in the transaction set definition by distinct tables of segments.

For example, the Price Sales Catalog Transaction Set (832) contains:

- A heading area, composed of reference information for the entire catalog
- A detail area, with individual line item information for each item in the catalog
- A summary area, containing a hash total count for the line items

The split of transaction sets into heading, detail, and summary areas is very common among EDI transaction set definitions. Unless otherwise defined in the publication of a transaction set definition, the heading, detail, and summary areas of a transaction set will be referred to as Tables 1, 2 and 3, respectively, excluding the transaction set header (ST) and trailer (SE) segments. Note that Table 2 would encompass multiple detail areas, if applicable.

The ST and SE segments are considered boundaries of the transaction set itself. However since these segments contain no semantic content (meaning) intended for the business application, they are part of the transaction set, but not part of any area in the transaction set.

Heading Area. Information in the heading area pertains to the entire transaction set:

- Every transaction set *definition* will have one and only one heading area, composed of at least one segment.
- At a minimum, the heading area of a transaction set definition typically contains data elements that identify each *instance* of a transaction set and its date. These elements are included to prevent replay of transaction set instances in a secure EDI environment.
- The use of any one segment in the heading area in any one *instance* of a transaction set is required only if the segment has been designated as “mandatory.” If the transaction set definition does not include a mandatory segment in the heading area, then it is possible for one instance of the transaction set to have no heading area.
- The entire heading area comprises a segment group (see Section 5.6), which is not repeated.

Summary Area. Information in the summary area addresses the results of summarizations of information in the detail area (see the next section), or additional information about the entire transaction that cannot

be expressed until the heading and detail areas have been generated. The summary area contains information such as control totals, balance totals, hash totals, etc.

- A summary area is not required in a transaction set *definition*.
- The use of any one segment in the summary area in any one *instance* of a transaction set is required only if (a) the summary area for the transaction set exists and (b) the segment has been designated as “mandatory” in that area.
- The entire summary area comprises a segment group, which is not repeated.

Each instance of a transaction set shall be considered the parent structure of the heading and summary areas that are enclosed within the boundaries of that transaction set.

5.5 Detail Areas of a Transaction Set

Detail Area. A detail area is any looping structure between the heading area and the summary area (or the SE, if no summary area exists). Detail areas encompass the actual body of the business transaction.

- There can be multiple detail areas in a transaction set definition.
- There is no requirement for a detail area in a transaction set definition.
- The use of any one detail area in any one transaction set instance is required only if the transaction set is designed to require at least one instance of the area.

Each instance of a transaction set, and the contents of its heading area, shall be considered the parent structure of any and all detail areas that follow the last segment of the heading area (or the ST segment if no heading area exists), and precede the first segment of the summary area (or the SE segment, if no summary area exists).

The summary area is NOT defined as a parent structure to the detail area.

5.6 Looping Structures Within an Area

Single Segment Repeat. Each segment in a transaction set is given an ordinal position assignment indicating its sequence within the transaction set relative to the other segments in that transaction set. Any individual segment may be repeated, in that same ordinal position, in a transaction set instance, if permitted by the transaction set design. However, the design should not allow repeat of the first segment in any loop (see below). If an individual segment is repeated in any one instance of a transaction set, and it is the first segment in a loop, a repeated occurrence of the segment indicates a new loop occurrence.

Segment Group. An ordered group of segments (more than one) within a transaction set is identified as a segment group. In the definition of a transaction set, segment groups are identified by the following: the entire transaction set; each area of the transaction set; the segments in an area not inside loops (see below) in that area; all of the segments of a loop or nested loop; and those segments in a loop not inside nested loops within that loop.

Loop. A segment group may be repeated in a transaction set instance, if permitted by the transaction set definition. Such an ordered group is called a loop. Note that each instance of a loop is a segment group.

Nested Loop. A loop may contain a loop. This condition is referred to as “nesting.” In a transaction set *definition*, a loop whose first segment appears after the first segment of an immediately preceding loop, and whose last segment appears no later than the last segment of that preceding loop, is a nested loop within that preceding loop.

HL-initiated Loop. A looping structure in a transaction set definition may begin with an HL (“Hierarchical Level”) data segment. When used in any transaction set instance, successive occurrences of the HL structure could indicate a parent-child relationship among the occurrences. The HL segment, therefore,

allows the identification of an explicit nested loop in a transaction set. This nesting is not identified in the transaction set definition, but rather in the instance of the loop, in the data content in the HL segment. All semantic relationships that apply to nested loops will, then, apply equally to relationships between an HL-initiated parent loop and its HL-initiated child loops.

Loop Levels. For the purposes of this discussion, a loop which is nested within another loop is defined to be at a higher nesting level than the loop which nests it. The outermost loop is the lowest level loop; the innermost loop is the highest level loop.

- There are no limitations on the placement of loops. Looping structures may exist in any area of any transaction set.
- Transaction sets are defined to include a minimum number of occurrences of the loop. If the requirement designator of the loop's first segment is optional, the minimum number of occurrences of the loop is zero. If the requirement designator is mandatory, the minimum number of occurrences is one.
- Transaction sets are defined to include a specific maximum number of occurrences of the loop, or will be identified as having no maximum (>1).
- There is no limit to the level to which loops may be nested.

Each instance of an area shall be considered the parent structure of any looping structures enclosed within the boundaries (the first and last segments, per the transaction set definition) of that area.

Any loop that contains a nested loop shall be considered the parent structure of that nested loop.

5.7 Segments Within a Loop

Segments. Each instance of a loop (i.e., a segment group; note again that the header and summary areas are segment groups as well) contains one or more segments:

- Each segment within a loop structure is identified by its ordinal positioning within the complete transaction set definition. Each segment's order within the loop, relative to other segments within the loop, is defined by the sequential order of the ordinal positioning identifiers.
- The beginning segment of a loop defines an instance of a loop. For any instance of a transaction set, for any loop within any area of a transaction set, the first occurrence of the beginning segment of the loop (which must be present if any other segments in the loop are present) will identify the first instance of the loop, the second will identify the second instance, etc.

Each instance of a loop shall be considered the parent structure of any segments enclosed within the boundaries of that loop (the first and last segments, inclusive, of the loop, per the transaction set definition).

Each instance of a segment shall be considered the parent structure of any data elements used in that segment, as allowed by the segment definition.

5.8 Hierarchy of Semantic Levels

The following graph is a summary, hierarchical picture of the semantic levels outlined in the previous sections.

Note that "segment" has been identified as the bottom level, even though "data element" is certainly a level under segment. The semantic meaning of a data element or composite in relationship to the segment which contains it, and of data elements in relationship to a composite, is defined by X12.6, and will not be discussed further in this document.

Note also that the hierarchical flow of this graph applies equally to the cases of *definition* and *instance* of the standards. However, the level numbering in Figure 1 and the discussion about level numbering following it pertain to *instances* of EDI communications only.

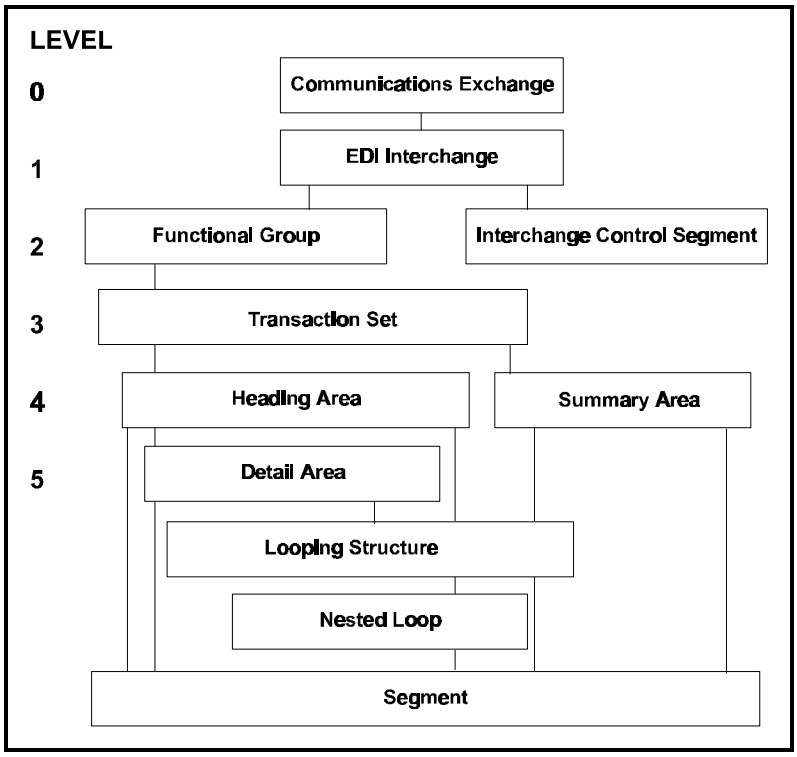

Figure 1: Semantic Levels in the Structure of EDI

Figure 1 presents a level numbering system that is useful for the determination of parent-child and same-level relationships among semantic entities. Note that outermost layers are identified by lower numbers. While this may be counter-intuitive, it reflects the fact that there is no limit to the nesting of loops at the innermost layers. So, the hierarchical level of the parent will be numerically lower than that of the child. To complete the numbering system, for any instance of a semantic entity within any one communications exchange, the following is noted:

- From levels 1 through 5, an instance of a semantic entity will be sub-indexed. For example, the first instance of an interchange within a communications exchange can be identified as entity 1.1, the second as 1.2, etc. (the ".1" an ".2" represent the sub-indexes). Similarly, the first instance of a detail area would be 5.1, the second 5.2, etc. Since only one heading area and one summary area are included in any transaction set, these levels are not sub-indexed.
- Below levels 4 and 5, instances of loops are identified by sub-indexing, within the semantic entity to which they belong, to the level at which they are nested. For example, the following level identifiers would pertain:

Level	Semantic Entity
4.1	**Heading Area**
	Beginning Segment(s)
4.1.1	Loop 1
	Loop 1 Segment(s)

4.1.2	Loop 2 Loop 2 Segment(s)
5	**Detail Area**
5.1	Loop 1 Loop 1 Single-Occurrence Segment(s)
5.1.1	Nested Loop 1 Nested Loop 1 Segment(s)
5.1.2	Nested Loop 2 Nested Loop 2 Segment(s)
5.2	Loop 2 Loop 2 Single-Occurrence Segment(s)
5.2.1	Nested Loop 1 Nested Loop 1 Segment(s)
5.2.2	Nested Loop 2 Nested Loop 2 Segment(s)
4.2	**Summary Area** Summary Area Single-Occurrence Segment(s)
4.2.1	Loop 1 Loop 1 Segment(s)

Figure 2: Semantic Level Numbering Example

Note that, under this numbering scheme, level 4 is the combination of the header and summary areas, which are at the same level but distinct, so that any looping structure in one area is not a child structure to the other area.

The example leads to rules for identifying the hierarchical positioning among levels:

- If one semantic entity is a component of another, and the numerical value of the component entity's level is greater than that of the entity of which it is a component, the latter is a parent of the former. For example, each of the two semantic entities at level 4 (header and summary areas) will be a child of its corresponding semantic entity at level 3 (a transaction set). As another example, note that while the detail area (level 5) is a child of the header area at level 4.1, it is not a child of the summary area at level 4.2, because the detail area is considered a component of the header area, but not of the summary area.
- Two semantic entities are related in a parent-child structure if the numerical value of the subscripting of the two entities is the same, except that the latter has one more digit. For example, loop 5.1.1 is a child of loop 5.1. Conversely, loop 5.2.1 is not a child of loop 5.1.
- Two semantic entities are at the same level in an instance if they share the same subscripting up to the last digit. Thus, loops 5.1.1 and 5.1.2 above are at the same level. Conversely, note that there is no similar semantic level relationship between loops 5.1.1 and 5.2.1.

This, of course, does not apply to loops only. Two transaction sets in the same functional group would be identified as 3.1 and 3.2, for example, and therefore be at the same semantic level.

The discussion that follows identifies the rules governing the definition and instances of parent-child and same-level semantic entity relationships.

6 Relationships Between Semantic Levels

6.1 Belonging

The data contents of each parent structure are relevant to the data contents of the levels to which it is a parent. In this way, the instances of data content at one level *belong* to its parent structure.

EXAMPLES:

1. *Inter-level Referencing*. Each level in the EDI structure (other than the communications exchange; see the note above in describing that level) contains a unique referencing component between adjacent levels.

Interchange	Sender/Receiver Codes/IDs and Interchange Control Number
Functional Group	Sender/Receiver and Group Control Number
Transaction Set	Transaction Set Code and Control Number
Area	Table Designator
Loop	Loop Name
Segment	Segment Position Number

Each instance of data content at one level, therefore, can be referenced to its parent structure. Multiple instances at any one level, for the interchange, functional group, and transaction set levels, can be distinguished uniquely by their respective control numbers.

In addition, since the heading and summary areas can be uniquely identified within any one transaction set instance, their contents can share the unique reference of the transaction set.

However, in the case of multiple detail areas, and with respect to loops and segments, the transaction set *definition* must allow a means of uniquely identifying each instance. Examples are:

- The Payment Order/Remittance Advice Transaction Set (820) has two detail areas, which are distinguished by the loop names (ENT and TXP) that start each.
- The N1 loop starts with an N1 segment, the contents of which should be unique for every instance of its use.
- The LIN segment is often used to begin a detail area definition. This segment allows a unique “line

2. *Transaction Set Areas*. The definition of the heading area above has indicated that it maintains a scope of applicability over the entire detail area. A discount factor in the heading area would apply to each line item in the detail area. A discount factor for a specific line item in the detail area would apply to that line item only (note the issue on “functional equivalence” below).

6.2 Order Independence

There is no implied relationship among instances at the same level (two functional groups within the same envelope; two transaction sets within the same functional group, etc.).

Each instance of data content at one level is independent of any other instance of data content at the same level.

EXAMPLES:

1. *Transaction Set Independence*. Each transaction set instance in a functional group is a separate unit. The data contents in one cannot reference the data contents in another, except through explicit application data, such as a reference number. Such references are outside the scope of this standard.

2. *Loop Independence*. There is no semantic relationship implied in the order of instances of the same loop. Often, in the transmission of multiple detail line items, the sender would like to identify data content (e.g., a discount factor) in the loop containing the first line item and have it apply to all subsequent line items. This is a violation of "order independence," since the data contents of the second loop are in no way referenced to the data contents of the first.

Note also that the standards are moot on the multiple use of "functionally equivalent" information in loop instances at the same level. For example, there is no specific prohibition for a sender to use the N1 loop twice, and identify a different "bill-to address" in each instance.

Such an occurrence may represent an error in the transaction, but it is also conceivable that there could be a trading partner agreement in which the duplication is understood. If a semantic note is provided that justifies such a duplication, the structure of any instance of the transaction set must reflect the meaning of the semantic note. Unless such an agreement is desirable, the transaction set *definition* should indicate semantic notes to eliminate or justify such conflicts.

3. *Segment Independence in a Loop*. All of the segments within a loop are semantically related to one another only in that they are of the same instance at the same level. However, there is no semantic content implied by the order of segments within the same level of a loop. Rearranging the order of segments within the same level does not alter the semantics.

This applies equally to the first segment in a loop, which is often assumed to have special meaning to the loop. This meaning must be explicitly stated in a semantic note to the transaction set for its meaning to be different from that of the other segments in the loop. The exception is when the loop begins with an HL segment, the content of which identifies the nesting level of each instance.

Note, though, that the first segment of a loop, however, is *syntactically* significant, in that its requirement designator determines the requirement of the loop, and it must be present if any other segments in the loop are present.

4. *Multiple Detail Areas in the Transaction Set Definition*. Each detail area shall be considered to be at the same semantic level. There is no precedence among detail areas.

6.3 "Functionally Equivalent Data" Override

When the data contents of a parent structure contain information that is perceived to be duplicated in the child structure, the data contents in the two areas are considered to be *functionally* equivalent. There are three distinct possibilities for an instance of functional equivalence.

Unqualified segment functional equivalence. In any transaction set instance, a segment will be considered functionally equivalent to a segment in its parent structure when:

- The two segments have the same segment identifier.
- No qualifier is required to identify the use of that segment uniquely.
- The maximum use of that particular segment is once in each structure.

Qualified segment functional equivalence. When the second requirement above is not the case (e.g., the N1 segment requires the" Entity ID Code" to identify its use uniquely), the following set of conditions will identify a case of functional equivalence:

- The two segments have the same segment identifier.
- For that segment, a data element must exist that qualifies the entire segment.

- The two instances of the qualifying data element must be equal.
- The maximum use of that particular segment, with that particular qualifying data element, is once in each structure.

Loop functional equivalence. In order for an instance of a loop to be considered functionally equivalent to a loop in its parent structure, the following must be true:

- In the transaction set definition, each loop must have the same beginning segment.
- The instance of that loop in the child structure must begin with a segment that is functionally equivalent to the segment in the parent structure loop, whether qualified or unqualified.

Whenever functionally equivalent data are provided in a child structure instance, the data contents in the child structure will override (i.e., replace) the data contents in the parent structure. The over-ride is in effect for that child instance only.

Functional equivalence can be circumvented in a transaction set instance through the assignment of different code values in the (otherwise) functionally equivalent entities.

Also note, to generalize the point made earlier regarding loop ambiguity (see Section 6.2, example 2), that an instance of functionally equivalent data at the same level creates an ambiguity that cannot be resolved, except possibly by prior trading partner agreement.

EXAMPLES:

1. *Heading and Detail Area "Contradiction".* The heading area often contains an N1 looping structure allowing for the identification of, for example, a delivery address to which all of the line items to be described in the detail area are to be delivered (i.e., "ship-to address").

If an N1 loop exists within the detail area, an instance of that loop may specify a delivery address different from that indicated in the heading area. In this case, the information content on that specific line item (to which that N1 loop applies) overrides the delivery address in the heading area. That is, the detail area ship-to address takes precedence.

There is one cautionary note about this example, however. Consider the following two segment instances:

N1*ST*.....
N1*BS*.....

The first of these indicates a "ship-to address" and the latter a "bill-to and ship-to address." The user might then assume that an occurrence of the second segment in a detail area would override an occurrence of the former in the heading area of the same transaction set.

This is NOT true in this case, however, because:

- A unique use of the N1 segment is identifiable only by both the N1 segment name and the first data element, the "Entity ID Code."
- The requirements for functional equivalence include equality in the use of the Entity ID Code, which is not the case here.

2. *Summary and Detail Area "Contradiction".* By definition, information in the summary area is intended to summarize the information content in the detail area. As such, incompatibilities between the two areas would indicate an application-related error.

Further, information in the summary area has a semantically different relationship to the detail area from that in the heading area. That is, the scope of applicability of heading area information is on each of the line items (unless overridden at the line item level),while the scope of applicability of the summary area is on the totals derived from the detail area.

3. *Heading and Summary Areas "Contradiction".* Therefore, there is no semantic contradiction (and thus there are no precedence rules) for seemingly "equivalent" information in the heading and summary areas. Any functional equivalencies between these two areas would represent an unacceptable transaction set definition.

6.4 Semantic Addressing

The first example in the previous section introduces a topic that is of considerable semantic importance in X12 transfers, that of naming and addressing.

- A *name* may be thought of as a label for a person or position (e.g., John F. Doe, President of XYZ Company, etc.). A name is independent of location.
- A *semantic address* is a set of attributes which usually describes the logical or physical location of the person or position. Examples are the traditional address on an envelope or a telephone number. Addresses are usually structured to provide sufficient information for identifying or locating the person or position (which may or may not be for routing purposes).

Much of the name and address information used in X12 is coded, with no distinction between a name and an address because the code encapsulates both. This standard, therefore, identifies name and semantic address information as a single concept, called *addressing.*

This section will identify and clarify the differences and relationships among four types of addressing within the structures of X12 transfers:

1. Business application
2. Internal identification
3. External routing
4. Data communications

6.4.1 Business Application Type—Within Transaction Sets

A number of different types of addressing can be transferred within instances of X12 transaction sets. Usually this information is provided in a segment group, frequently starting with the N1 data segment.

This segment group has provisions for qualifying the information to follow (e.g., "payer" or "ship-to address"), and a coded or full-text version of the addressing information itself. Through the qualifier, many different types of addressing may be provided, each serving a different business purpose.

All qualifications of business addressing are considered to be the same business application type, with no implied semantic relationship among different qualifications at the same instance level (note the N1 example in the section on functional equivalence).

However, semantic relationships can be inferred from qualifications at different levels. Each of the relationships among transaction set levels defined in the previous sections applies to business application addressing in the same way it applies to all other business information.

Business application information is conveyed from one business application to another (which may or may not be in the same organization).This business information is independent of the transfer method (e.g., printed matter, X12 transfer, voice calls to order entry clerks, etc.).

All business application addressing is included within the boundaries of a transaction set instance. There is no business application information in the addressing information at the functional group, interchange, or communications exchange level.

6.4.2 Internal Identification—Functional Group Level

Internal identification addressing information is carried as codes in the Application Sender's Code and the Application Receiver's Code control elements in the functional group header (GS segment). The values for these codes can be used for any routing internal to the organization, or any other identification purpose providing further information for trading partner record keeping.

There is no universal meaning to the data content of instances of those codes. For example, if an instance of the Application Receiver's Code contains the value “9876543”, the internal identification could be a telephone number, a department (e.g., accounts payable), etc.

Because there is no universal meaning for these values, they can only be used for internal identification by the organization that understands their meaning. Meaning for particular values can be established by an individual organization, by trading partner agreements, or by industry groups.

Because of the internal nature of functional group addressing, there is no implied semantic relationship between instances of addressing at this level and that of the interchange and communications exchange levels (which are external routing levels).

Similarly, because this information has no universal meaning, there is no implied semantic relationship between instances of addressing at this level and that at the business application level (i.e., information within transaction sets).

Such relationships, if defined by trading partner agreements or by industry groups, would be outside the scope of the ASC X12 standards.

6.4.3 External Routing—EDI Interchange Level

Addressing information provided in the Interchange Sender ID and the Interchange Receiver ID of the Interchange Control Header (ISA control segment) is used for external routing.

In contrast to the functional group level, the values of instances of these control elements are qualified by the Interchange ID Qualified Code (e.g., a DUNS number, a phone number, the Standard Point Location Code, etc.). The values may then have universal meaning.

To restate the point in the previous section, addressing information provided at the EDI interchange level has no semantic relationship to addressing information at the functional group level.

Such a relationship, if defined by trading partner agreements or by industry groups, is outside the scope of ASC X12.

6.4.4 Data Communications—Communications Exchange Level

Addressing information in the interchange control header is usually not sufficient for delivery.

This information is typically mapped into a communications address for delivery, such as a modem telephone number, a CCITT X.121 address for a public data network, etc. This additional addressing information will depend on the particular data communications method being used.

Because of this close connection between the addressing at the data communications and interchange control levels, there is clearly a relationship that will have impact on proper processing (e.g., distribution, security, etc.).

For example, the receiver of an X12 logical communications exchange may determine that the interchange sender, as identified in the interchange level addressing information, is not authorized to send X12 information through the particular log-in channel that is (was) in use. This might be the case, for instance, when a VAN provides different log-in areas to indicate different distribution service levels.

Even so, such a relationship is not universal (since there are multiple, nonstandard mappings) and not part of the ASC X12 standards in any case. Therefore, there is no semantic content generated at the communications exchange level that is useful at any level inside the interchange.