EDI Standard Exchange Format

for Exchanging EDI Implementation Guidelines

4950 Blazer Parkway
Dublin, OH 43017-3305

Phone: (614) 791-1600
Fax: (614) 791-1609
Web: www.foresightcorp.com
E-mail: support@foresightcorp.com

Contents

Introduction to SEF Version 1.6

Target Audience

This document is for those experienced with EDI concepts and terminology. It explains the layout of Standard Exchange Format (SEF), used to exchange EDI implementation guidelines in a machine-readable form. EDI translators which can directly import SEF can save users considerable time in developing new translations or maps.

Foresight has made SEF an "open" standard. This means it can be used without permission or royalties for:

- Exchanging between trading partners as a means of conveying implementation guidelines information.

- Importing directly into translation or mapping software to ease implementation.

- Posting via networks to mailboxes or electronic bulletin boards for mass dissemination of implementation guidelines.

You can copy this publication, provided you preserve the copyright notice.

What's New in SEF 1.6

SEF 1.6 provides the ability to specify any number of level notes. Previous versions of SEF provided for up to three levels. See **Level Notes** on page 60.

In this document, changed text is marked with revision bars in the right margin.

Basics

- Each standard must be contained in its own plain text (ASCII) file. Any number of transactions or messages may be included as part of a standard in one SEF file.

- Use all upper case letters except within text such as descriptions, notes, etc.

- Omit spaces except within text such as descriptions, notes, etc.

- Omit blank lines.

- Some lines will be very long; if you edit this file, you will need to use a text editor (such as MS-DOS's EDIT or PCTOOLS' File Edit) that can save a file without breaking or truncating long lines. When typing the .SEF file on a computer screen, long lines will appear to break and wrap, but this is simply your editor's way to show the entire line.

- A comment takes up one complete line and must start with a * in the first column. Comments can appear anywhere in the file after the .INI section.

- High-ASCII and other non-printable characters can be included in the SEF file as 3-digit escaped values. Any character from x00 to xFF can be used.

Example message title in guideline: **`Purchase Order Message for André`**

Corresponding SEF:

```
.TEXT,SETS

ORDERS,0,Purchase Order Message for Andr\233
```

A single slash is interpreted as an escape character in SEF, as shown in the example above. To actually include a slash in guideline text, use two consecutive slashes.

Sections in a SEF File

Overview of Sections

Each section starts with a single record containing the section name preceded with a period and followed by a new line (N/L) character. Sections include:

Section	Purpose	Details on ...
.VER	The .VER section identifies the version and release of SEF, which is currently 1.6. It should be the first record in the file. If the .VER section is not present, SEF 1.0 is assumed.	page 5
.INI	The next section must be .INI. It contains basic information about the standard.	page 6
.PRIVATE and .PUBLIC	These sections can occur anywhere after the .INI section. The .PRIVATE section provides a place for companies to place information that is useful to themselves but is of no interest to others. The .PUBLIC section marks the end of the private section.	page 7
.STD	.STD is only included for these standards: • Newer EDIFACT standards in which groups have position numbers • Newer EDIFACT and X12 standards that have repeating elements • Fixed-length standards like GENCOD.	pages 8 and 15
.SETS	The .SETS section defines the transaction set or message directory, including: • Each transaction set or message in the standard. • For each transaction set or message, it lists each segment reference in the order in which it appears. It also describes the requirement and quantity for each segment when it appears in a particular position in that set. • Loops, groups, and tables are also set up. In this section, all information about loops, groups, and segments is hierarchical: for example, the quantity for a PER segment only applies to that particular position in that particular set or message.	page 8

.SEGS	The .SEGS section is the standard's segment dictionary: a list of all segments in the standard. It includes: • The segment ID or tag (ST, PER, etc.). • A list of each composite and element reference it contains, in order. • Each element's number and requirement when used in this position of this segment. • Element repeat counts. • Element relationals used within the segment. • Masks for variations in the structure of a segment.	page 18
.COMS	The .COMS section is the standard's composite data element dictionary: a list of all composites in the standard. It includes: • The composite name (C001, etc.). • A list of each subelement reference it contains, in order. • Each subelement's ID and requirement when used in this position of this composite. • Subelement relationals used within the composite. • Subelement repeat counts. • Masks for variations in the structure of a composite. If the standard has no composites, this section will be omitted.	page 28
.ELMS	The .ELMS section is the standard's data element dictionary: a list of each element, its type, and its minimum and maximum data value lengths.	page 31
.CODES	The .CODES section is a list of each element that has code values, along with its code values. It also provides information about code sets.	page 32
.VALREFS	The .VALREFS section shows where application value lists are used.	page 38
.VALLISTS	The .VALLISTS section is a list of each application value list, along with its values.	page 38
.SEMREFS	The .SEMREFS section provides semantic rules.	page 41
.OBJVARS	The .OBJVARS section lists variable names that are assigned to specific locations in a guideline. These names can be used in semantic rules in the .SEMREFS section.	page 40
.OVERHEAD	The .OVERHEAD section, if it exists, may be ignored. It is optional and shows how many duplicate or new records are included in the SEF file.	
.TEXT,SETS	The .TEXT,SETS section contains text information local to each transaction set or message: text describing the transaction or message itself, or text that will be used in a particular hierarchy in the set.	page 47
.TEXT,SEGS	The .TEXT,SEGS section contains segment dictionary text that differs from the standard.	page 51
.TEXT,COMS	The .TEXT,COMS section contains composite dictionary text that differs from the standard.	page 54
.TEXT,ELMS	The .TEXT,ELMS section contains element dictionary text that differs from the standard.	page 57

Order of sections

The sections may appear in any order, with these exceptions: the .VER, if present, must be first. The .INI section must be the first section in the file if there is no .VER, or the second section if there is a .VER. Nothing may appear before these, including a comment. The .STD record, if used, must appear before .SETS. Example:

```
.VER 1.6
.INI
KAVERPO,,004 010,X,X12-4010,Kaver Corp X12-4010 Purchase Order
.STD ,RE
.SETS
```

Omitted sections

Unneeded sections may be omitted. Example: if there are no composites, the .COMS section maybe omitted.

Duplicate sections

The contents of a section need not appear together. For instance, you can have a .SETS section defining a few transaction sets or messages, followed by a .SEGS section, then followed by another .SETS section defining more transaction sets or messages. Look at the following example:

```
.VER 1.6
.INI
  .
  .
  .
.SETS
810=ˆ ...
856=ˆ ...
.SEGS
  .
  .
  .
.SETS
810=ˆ ...
850=ˆ ...
```

Notice the two .SETS sections. The result would be equivalent to one .SETS section with the following:

```
810=ˆ ...
856=ˆ ...
850=ˆ ...
```

The 810 in the last .SETS section would have precedence (or override) the previous 810. In the case of duplicates, such as the 810s in this example, all except the last one are ignored.

The .VER Section

If the .VER section is included, it must be the first record in the SEF file. If omitted, version 1.0 is assumed. The following line shows the current version, 1.6:

```
.VER 1.6
```

Updates to previous SEF Versions	
Version	**Additions**
1.1	PUBLIC and PRIVATE.
1.2	Negative and zero increments for sequence numbers.
1.3	Element repeat counts, EDIFACT sequence number changes for newer standards, notes for EDIFACT groups, recommended and not recommended segment requirement, capability of overriding dictionary requirements or not.
1.4	Application value list descriptions and regular expressions. Dependent usage. High-ASCII and other non-printable characters can be included as 3-digit escaped values.
1.5	Level 3 notes, repeating elements, fixed-length standard support.
1.6	Unlimited number of level notes.

The .INI Section

The .INI section immediately follows the .VER section. If .VER is omitted, then the .INI section must be the first two lines in the SEF file. The following is the .INI section for our example local standard INVPO.

```
.INI
INVPO,,003 040,X,X12-3040,PO and INV for Slippers 'n Socks, Inc.
   ↑ ↑     ↑   ↑     ↑                        ↑
   1 2     3   4     5                        6
```

 A comma separates each field. The numbers marking the fields in the example above are, in order:

1. The standard or implementation name (INVPO in the example above), generally the same as the filename of the SEF file. It should be 1 to 8 characters that would be valid as an MS-DOS file name.

2. Reserved unused field that should be ignored.

3. The Functional Group Version, Release and Industry code which will identify the standard in any Functional Group Envelope Header Segment. Each code is separated by a space. In the example, there is no industry code. With an industry code, this field might contain: `003 030 UCS` .

4. The responsible agency code, which identifies the standards organization in the Functional Group Header:
    GC GENDOD
    T   for T.D.C.C. (EDIA)
    TD TRADACOMS
    UN for UN/EDIFACT
    X   for ASC X12 (DISA)

5. The standard on which this implementation guidelines is based.

6. The description (title) of the implementation guideline.

Please see page for an example .INI section along with the other sections in the SEF.

The .PRIVATE and .PUBLIC Sections

PRIVATE is a place for companies or individuals to store information that they need to augment information in the SEF file. It should be ignored by everyone else. The first entry in .PRIVATE is usually a company name. Subsequent lines contain whatever information is needed.

Top of Typical SEF File for Old EDIFACT Standard

```
.VER 1.6
.INI
SEFD93A,,D 93A,UN,D93A,Nomachi Corp ORDERS for vers. 93a
.PRIVATE NOMACHI CORP
.AUTHOR NOMACHI CORP EASTERN DIV
.PUBLIC
.SETS
```

Top of Typical SEF File for New EDIFACT Standard

```
.VER 1.6
.INI
SEF-E961,,D 96A,UN,D96A,Kaver Corp ORDERS for 96a
.PRIVATE NOMACHI CORP
.AUTHOR NOMACHI CORP EASTERN DIV
.PUBLIC
.STD ,LS
.SETS
```

Top of Typical SEF File for X12 Standard

```
.VER 1.6
.INI
SEF-3061,,003 061,X,X12-3061,Blue Mountain 850
.PRIVATE NOMACHI CORP
.AUTHOR NOMACHI CORP EASTERN DIV
.PUBLIC
.SETS
```

The .PRIVATE section in these examples consists of a company name (NOMACHI CORP), plus some local information (AUTHOR and DATE records) that should be ignored by everyone but Nomachi Corp. The .PUBLIC record ends the .PRIVATE section and restarts the public part of the SEF file.

Multiple .PRIVATE sections are allowed. Terminate each one with either another .PRIVATE or with a .PUBLIC:

```
.VER 1.6
.INI
CLASS850,,003 070,X,X12-3070,X12-3070 Purchase Order for EDISIM Training
Class
.PRIVATE FORESIGHT
.CTL *
.TDVER 6KDJ
.DATE 01/04/101:09:23:04
.PRIVATE KAVER
.AUTHOR Kaver Corp.
.PUBLIC
.SETS
```

The .STD Section

.STD is only included for these standards:

- Newer EDIFACT standards in which groups have position numbers
- Newer EDIFACT and X12 standards that have repeating elements
- Fixed-length standards like GENCOD.

The format is:

   .STD , *param,param,param*

► Important: The space and comma after the .STD are significant.

param can be (in any order):

LS (Available in EDIFACT D96A and later) The presence of **LS** means that individual .SETS records do not appear different, but they will be interpreted differently: LS stands for "Loop Sequence."

- Sequence numbers continue ascending through table boundaries, rather than starting over in each new table.
- Groups participate in the numbering.
- Groups have ordinals. @ operators, which indicate a change to the ordinal increment, can appear before loops as well as before segments.

RE (Available in starting with X12-4030 and EDIFACT D99A) The presence of **RE** means that the standard can contain repeating elements.

FX The presence of FX indicates a fixed-length, non-delimited standard like GENCOD.

Example top of SEF file containing a .STD record:

```
.VER 1.6
.INI
MYSTD,,D 99B,UN,D99B,UN/EDIFACT Draft Messages and Directories
Version D.99B - publ. Sep. 1999
.PRIVATE FORESIGHT
.CTL
.TDVER QLRW
.DATE 02/24/100:11:31:55
.PUBLIC
.STD ,LS,RE
.SETS
```

The lines between .PRIVATE and .PUBLIC are proprietary. They are not defined by the SEF file format.

For details about .SETS, see the next section.

The .SETS Section

The .SETS section defines each transaction set or message within the standard or implementation guideline. A typical .SETS section might look like the following. Each transaction set or message is one very long record, although we show it wrapped on this printed page. There are only 3 "new lines" (carriage return-line feeds) in this section: after .SETS, after the 810's definition, and after the 850's definition. They are shown as the *n/l* symbol.

```
.SETSn/l
810=^[ST,M][BIG*1,M][NTE,F,100][CUR][REF,,12][.PER,,>1]{:200[.N1][.N2,,2][.N3,,2][.N
4][.REF,,12][.PER,,3]}[.ITD,,5][.DTM,,10][.FOB][.PID,,200][.MEA,,40][.PWK,,25][.PKG,,
25][.L7][.AT,,3]+2[.BAL,,2]{:10+8[.LM]+10[.LQ,M,100]}{:1[N9][MSG,M,10]}^{:200000[IT1]
+2[CRC]+3[.QTY,,5]+5[.CUR]+10[.IT3,,5][.TXI,,10][.CTP,,25]+9[.MEA,,40]{:1000+1[PID]+1
0[MEA,,10]}[PWK,,25][.PKG,,25][.PO4][.ITD,,2][.REF,,1][.PER,,5][.SDQ,,500][.DTM,,10]
[.CAD,,>1][.L7,,>1]+5[.SR]{:25[.SAC]+10[.TXI,,10]}{:1000[.SLN][.REF,,>1][.PID,,1000][
.SAC,,25]+5[.TC2,,2]}{:200[.N1]+10[.N2,,2][.N3,,2][.N4][.REF,,12][.PER,,3]}{:10[.LM][
.LQ,M,100]}}^[TDS,M][.TXI,,10][.CAD]{:25[.SAC][.TXI,,10]}[.ISS,,5][CTT,M][SE,M]n/l
850=^[ST,M][BEG,M][NTE,F,100][CUR][.REF,,12][.PER,,3][.TAX,,>1][.FOB,,>1][.CTP,,25]+
20[.CSH,,5]+10[.SAC,,25][.ITD,,5][.DIS,,20]+5[.INC][.DTM,,10]+10[.LDT,,12]+20[.LIN,,5
]+5[.SI,,2][.PID,,200]+10[.MEA,,40][.PWK,,25][.PKG,,200][.TD1,,2][.TD5,,12][.TD3,,12]
[.TD4,,5][.MAN,,10][.CTB,,5]+5[.TXI,,>1]{:1000[N9]+10[MSG,,1000]}{:200[N1][N2,,2][N3,
,2][N4]+5[.NX2,,3][.REF,,12]+10[.PER,,3][.FOB][.TD1,,2][.TD5,,12][.TD3,,12][.TD4,,5][
.PKG,,200]}^{:100000[PO1,M]+8[SI,,5]+2[CUR]+10[.PO3,,25][.CTP,,25]+9[.MEA,,40]{:1000+
1[PID]+10[MEA,,10]}[PWK,,25]+20[PO4]+10[.REF,,12][.PER,,3]+20[.SAC,,25]+10[.IT8][.ITD
,,2][.DIS,,20]+5[.INC][.TAX,,>1]+10[.FOB][.SDQ,,500][.IT3,,5][.DTM,,10][.LDT,,12][.SC
H,,200]+5[.TC2,,2][.TD1]+10[.TD5,,12][.TD3,,12][.TD4,,5][.MAN,,10][.AMT]+5[.TXI,,>1]{
:200[PKG]+10[MEA,,>1]}{:1000+20[N9]+5[MEA,,40][MSG,,1000]}{:200+10[N1][N2,,2][N3,,2][
N4]+5[.NX2,,3][.REF,,12]+10[.PER,,3][.FOB]+5[.SCH,,200][.TD1,,2]+10[.TD5,,12][.TD3,,1
2][.TD4,,5][.PKG,,200]}{:1000[.SLN][.SI,,5][.PID,,1000][.PO3,,104]+5[.TC2,,2][.SAC,,1
0]+10[.DTM,,10]+2[.CTP,,25][.PO4]{:10+6[.N1]+10[.N2,,2][.N3,,2][.N4][.NX2,,3][.REF,,1
2][.PER,,3]}}}^[CTT,M][.AMT][SE,M]n/l
```

This .SETS section contains two transaction sets: an 810 and an 850. If this is the only .SETS section in the file, then the standard contains just two transaction sets. Other .SETS sections may appear elsewhere in this file, containing more transaction sets to be added to the standard. If the same transaction set appears in more than one .SETS section, then the last one is used and the others are ignored.

The 810 and the 850 in the example are in larger, bold letters so that you can see them better, but the SEF file actually contains no font changes.

Let's take the first half line and examine its parts:

```
810=^[ST,M][BIG*1,M][NTE,F,100][CUR][REF,,12][.PER,,>1]
    ↑     ↑        ↑       ↑      ↑     ↑         ↑
    1     2        3       4      5     6         7
```

These numbers are explained below.

1.   810=   Each transaction set or message starts with its number or name, followed by the equal sign (=) . For EDIFACT, this might look like: **ORDERS= .**
This is often, but not always, followed by a ^ (circumflex) indicating the start of a table.

Transaction sets or messages cannot have empty tables, so two consecutive ^ are treated as one.

The rest of the .SETS section is basically a list of the segment references, in the order in which they appear in the transaction set or message. Each segment is enclosed in square brackets along with other information:

[*segment,requirement,maximum*]   example: [DTM,M,35]

requirement and *maximum* are optional. If only *requirement* is omitted, both commas will be included as field separators.

requirement is the requirement designator from the published standard (X12-3040, for example). Values for *requirement* can include: M, O, F, or C. If omitted, the default is O for optional (X12 or TDCC) or C for conditional (EDIFACT).

The default requirement for a loop or group trigger (first segment in the loop or group) is *always assumed to be M if the loop or group is used*, and so the requirement shown in the SEF file on a loop or group trigger is to be interpreted as the requirement for the loop or group as a whole. Please see Loops or Groups on page 11 for examples.

Values for Maximum Usage can range from 1 to 200000 or it can be **>1**, which is ASC X12 notation for unlimited repetition. If maximum use is omitted, the default is 1.

The maximum usage for the first segment in a loop or group is always 1, regardless of what is specified in the *maximum* field. If the max use for such a segment is omitted, or coded as something else, it should nevertheless be interpreted as 1.

In certain segments, you may see:

[*segment*n,requirement, maximum use*]	example: [DTM*2,M,35]
[*segment@n,requirement, maximum use*]	example: [DTM@1,M,35]
[*segment*n@n,requirement, maximum use*]	example: [DTM*2@1,M,35]

The asterisk (*) followed by a number means the segment is using a "mask" – the structure of the segment varies from the dictionary. Example 3 below has a mask. They are described in "'Masks' or Alternate Variations of a Segment" on page 23.

The at-sign (@) followed by a number means the segment's ordinal number is out of sequence. This may be caused by inserting or deleting a segment. Because it is out of sequence, its number must appear with it. Ordinal numbers are described in Ordinal Numbers in the .SETS Section on page 13. They correlate notes and other text with the segments in the transaction set or message.

These numbers correspond to the numbers in the previous example 810 transaction set.

2. **[ST,M]** The first segment in the 810 is an ST and it is mandatory according to the published standard. Each segment appears in order, enclosed in square brackets. The other fields within the square brackets are, in order:

   requirement (default=O for X12 and C for EDIFACT).
   maximum usage (default=1).

   Since the maximum usage for the ST is 1, it can be omitted.

3. **[BIG*1,M]** The second segment in the 810 is a BIG segment and its requirement is M for mandatory. Since maximum usage is omitted, its usage is assumed to be 1.

   The *1 means that this instance of the BIG segment, in this position of this transaction set, is to use mask 1. Masks are variations of a segment that differ from the dictionary. They are explained in 'Masks' or Alternate Variations of a Segment on page 23.

4. **[NTE,F,100]** The third segment in the 810 is a NTE segment. Since it is not using defaults for requirement or maximum quantity, these are both included.

5. **[CUR]** The fourth segment in the 810 is CUR. Since it uses defaults for both requirement (O) and maximum quantity (1), it omits these fields.

6. **[REF,,12]** The fifth segment in the 820 is REF. It uses the default for requirement (O). The comma delimiters must be included so that the 12 will be interpreted correctly as the maximum quantity.

7. **[.PER,,>1]** The period in front of the segment name means that this segment is marked as "not used" at this position, according to the convention. The maximum is ">1", meaning unlimited.

   [!ZZZ,,>1] If this appeared, it would mean that segment ZZZ must be used, according to the convention.

   [$ZZZ,,>1] If this appeared, it would mean that segment ZZZ is recommended, according to the convention.

   [-ZZZ,,>1] If this appeared, it would mean that segment ZZZ is not recommended, according to the convention.

   [&ZZZ,,>1] If this appeared, it would mean that segment ZZZ is dependent, according to the convention. The dependent usage might be due to a semantic rule in the .SEMREFS section. However, it can be dependent without any involvement in a semantic rule.

If a segment is at the beginning of a loop or group, its requirement is actually referring to the loop or group as a whole. Please see the next section for details.

Loops or Groups

Looking at the example on page 8, we see these segments at the end of the first line and the beginning of the second line:

```
{:200[.N1][.N2,,2][.N3,,2][.N4][.REF,,12][.PER,,3]}
 ↑  ↑  ↑                                        ↑
 1  2  3                                        4
```

This is an N1 loop containing segments N1, N2, N3, N4, REF, and PER.

1. Loops are enclosed in {curly braces}. The first field, the loop ID, can be omitted if it is the same as the first segment's tag - the usual case. It is followed by a colon.

2. The 200 is the loop repeat count. The repeat count is always preceded by a colon. If omitted, a maximum repeat count of 1 is assumed: `{[.N1]...}` is the same as `{:1[.N1]...}`.

3. The first segment (the "loop trigger") in this loop is preceded with a dot, and therefore the entire loop is unused. The other segments in the loop are also marked unused, but they don't have to be since usage of the entire loop is determined by the first segment.

   The loop trigger's maximum repeat count is always 1. If it is omitted, or coded as something else, it should always be interpreted as 1. This is true of EDIFACT groups also.

4. The loop ends with a closing curly brace.

Loops can be nested:

`{:10[TDT]{:10[LOC][DTM,,5]}}`

The inner loop LOC can appear up to 10 times with each iteration of the outer loop TDT. Since the TDT loop can appear up to 10 times, this means the LOC loop can appear up to 100 times at this point.

The loop ID is not normally included because it is assumed to be the same as the loop trigger's segment tag. If these two differ, then the loop ID is explicitly included in the SEF file before the colon:

`{0100:10[LOC][DTM,,5]}`

In this example, the loop ID is 0100 but the first segment is LOC:

Loop **0100**

   LOC segment

   DTM segment

The loop ID is most often seen in X12 standards and guidelines, providing IDs for the LS and LE segments in bounded loops.

Requirements for Loops, Groups, and Trigger Segments

A loop or group trigger is the first segment in the loop or group. In this EDIFACT example, the RFF is group 1's trigger:

the group is conditional here

0030	DTM	Date/Time/Period	M	15
0040	PAI	Payment Instructions	C	1
0041		GROUP 1 STARTS - 10	(C)	10
0042	RFF	Reference	M	1
0043	DTM	Date/Time/Period	C	5
		GROUP 1 ENDS		

but, if the group is used, the trigger is always mandatory

The group as a whole is conditional, but, if the group is included, then the RFF segment is mandatory. The first part of the SETS section for this would be:

`ORDERS=^[UNH,M][BGM,M][DTM,M,35][PAI]+1@213{:10[RFF][DTM,,5]}` ... *etc.*

The absence of a requirement on the RFF would normally imply that the RFF's requirement is "C" (for EDIFACT; the default for X12 is O). However, the requirement for a group trigger such as RFF is always assumed to be M. The spot where its requirement would be is used by SEF to show the requirement for the group as a whole. Therefore, in SEF, the requirement (of the published standard and of the user) on the group trigger is to be interpreted as the requirement for the group as a whole. In the SEF portion above, [RFF] shows no explicit requirement, meaning the group gets the default EDIFACT requirement of C.

In the next example from a convention, the group as a whole is mandatory. The group trigger, RFF, is of course mandatory as always.

the group is mandatory here

0040	PAI	Payment Instructions	C
0041		GROUP 1 STARTS - 10	(M)
0042	RFF	Reference	M
0043	DTM	Date/Time/Period	C
		GROUP 1 ENDS	
0050	ALI	Additional Information	C

The first part of the SETS section for this would be:

`ORDERS=^[UNH,M][BGM,M][DTM,M,35][PAI]+1@213{:10[RFF,M][DTM,,5]}` ... *etc.*

The M in [RFF,M] means the group as a whole is mandatory. Again, the RFF segment itself, as the group trigger, is always mandatory.

In this next example, the first two groups are both conditional according to EDIFACT, but their requirement has been changed by the user. (Underlining shown below is not actually in the SEF file.)

`DELFOR=^[UNH,M][BGM,M][DTM,M,10]{:10[.RFF][DTM]}{:20[!NAD][LOC,,10] ... }`

The RFF has a dot in front of it, meaning unused. Since it is a group trigger, this dot is actually referring to the group as a whole. The next group's trigger, NAD, has an exclamation mark in front of it, indicating that, while EDIFACT says it is conditional, it has a user requirement of "must be used." Since it is a group trigger, its requirement or user requirement always refers to the group as a whole.

For a list of user requirement symbols, see Summary of Symbols used in .SETS on page 17.

Tables

A circumflex (^) marks the beginning of a table or area. The third line in the example looks like this:

`[.PKG,,25][.L7][.AT,,3]+2[.BAL,,2]{:10+8[.LM]+10[.LQ,M,100]}{:1[N9][MSG,M,10]}^{:`

The `^` near the end of the line marks the start of a new table: in this case, Table 2. Table 1 was marked by the `^` right after the "810=" at the beginning of the transaction set. A third `^` in the next to the last line marks Table 3. If used at all, there are usually three tables: Heading, Detail, and Summary. Transaction sets and messages need not be divided into tables or areas.

Since a table cannot be empty, two consecutive circumflexes (^^) are treated as one.

Ordinal Numbers in the .SETS Section

Each segment reference has its own number, which is:

- unique within a transaction set or message
- permanent

This is called its *ordinal* number. All segment references have ordinal numbers, although most do not display them.

Ordinal numbers in the .SETS record provide permanent keys that can be used in the .TEXT sections. Because of permanent ordinal numbers, these keys do not change each time a segment is added or deleted from the transaction set or message.

Ordinal numbers are normally assigned in order, with the first segment in the transaction set or message having the number 1, the second having 2, etc.

```
        850=^[ST,M][BEG,M][NTE,F,100][CUR][.REF,,12][.PER,,3]
              ↑        ↑        ↑         ↑      ↑          ↑
Ordinal num→   1        2        3         4      5          6
```

In the example above, the number below each segment shows its ordinal number. The ordinal numbers for most segments do not usually show up in the .SETS section because they can easily be determined by counting through the segments.

The ordinal number must show up if the flow is disrupted by inserting or deleting a segment in the transaction or message. If a segment is inserted into the transaction or message, it will be given the first unused number. That number will show up in the .SETS section along with the segment. Let's assume that an NTE segment has been inserted right after the CUR segment. Let's further assume that, before the insertion, there were 114 segments included, so they have been assigned ordinal numbers 1-114. The inserted NTE will be assigned the next available number: 115. It will show up in the .SETS section preceded with an @ sign:

```
        850=^[ST,M][BEG,M][NTE,F,100][CUR]+5[NTE@115][.REF@5,,12]+10[.PER,,3]
              ↑        ↑        ↑         ↑       ↑          ↑               ↑
Ordinal num →  1        2        3         4      115         5               6
```

The ordinal numbers are shown below each segment. The ones that actually appear in this section are 115 and 5. The inserted NTE is followed by "@115" since its ordinal number is out of sequence. The REF segment after it also contains an ordinal number. If it didn't, its ordinal number would be assumed to be 116 (by adding 1 to the previous segment's number). The last segment shown, PER, does not need to show its ordinal number, because the number is one more than that of the previous segment.

If the SEF file has the .**STD, LS** record (for newer EDIFACT standards; see page 3), then segment groups have ordinal numbers too:

`GENRAL=[UNH,M][BGM,M]@29{:1+5[BUS]+2[CAV]}+3[DTM@3]`

In this example, segment UNH has ordinal 1, BGM has ordinal 2, group 1 (BUS) has ordinal 29, segment BUS has ordinal 30, segment CAV has ordinal 31, and DTM has ordinal 3. An ordinal for a group precedes the opening curly bracket for the group. An ordinal for a segment immediately follows the segment tag, inside the square brackets and before the requirement, if any.

Ordinal numbers are used to identify segments in the .TEXT section and in the .CODES section.

Position Number Increments

Position (or sequence) numbers reflect the position numbers of the X12 or EDIFACT standards. Unlike ordinal numbers, they have no other function within SEF.

Position numbers are assumed to increment by 10 for segments. In places where the increment is something else, the SEF file will show the change with a plus or minus sign. Let's say the convention has these segments and position numbers (among others):

```
10 ST
15 NTE
20 G50
30 LS
40 N1
```

This part of the SEF file would look something like this:

```
875=ˆ[ST,M]+5[NTE@32,M,2][G50@2,M]+10[LS,M]{:10[N1,M][N2][N3,,2][N4]}
     ↑     ↑              ↑           ↑         ↑
     1     2              3           4         5
```

1. The ST segment is at position 10 (the default if no position number is included for the first segment in a transaction or table).

2. The next segment would normally be at 20, but since this NTE segment has been inserted, it will be squeezed in at position 15. Therefore, the +5 appears here, saying that the position numbers should increment by 5 until further notice.

3. The increment is not changed before the next segment, G50, so it continues incrementing by 5.

4. The increment is reset to 10 right before the LS.

5. Since the increment is not reset before the N1 loop, it remains at 10.

Position number of the first segment: If the position number of the first segment is not 10, the SEF file will include a position number increment in front of it. For example, assume the position numbers are:

```
001  ST              040   CUR
002  BEG             050   REF
030  NTE
```

The SEF file might show: 850=ˆ+1[ST,M][BEG,M]+28[NTE,F,100]+10[CUR][REF,,12]

Re-using position numbers: The only time it's acceptable to recycle sequence numbers is for alternative definitions of segments or loops[1] such as a "Bill To" N1 loop defined separately from a "Ship To" N1 loop. This would cause

---

[1] Position (or "sequence") numbers are not re-used within a published ASC X12 transaction set table or area, or a UN/EDIFACT message. User conventions show a company's or trade group's use of a standard transaction set or message, and in general it would be useful for the guideline to preserve the sequence numbers of the underlying standard. This makes it easier for translators to align the .SETS definition with their translator tables when auto-loading maps with SEF input. Currently, conventions by voluntary organizations rarely re-use sequence numbers when defining alternate definitions. Example: 003040VICS, published by the Uniform Code Council. The underlying ASC X12 transaction set 856 contains a single HL loop in Table 2, starting at position 100 and extending to 350, generally in increments of 10. VICS defines five different varieties of the HL loop (Shipment, Order, Tare, Pack and Item, respectively). To squeeze the five loops into Table 2, the HL loops start at 100, 200, 300, etc. In other cases, a table may be added for each alternate. It may be harder for some auto-load facilities to handle the situation where new tables are invented.

the same segment tag to be defined twice in the convention. If the N1's in the SEF file keep the same position numbers as in the underlying ASC X12 standard, it will be easier for some translators to auto-load SEF accurately.

Since SEF files might include more than one variation of a loop, group, or segment, the increment may be negative or zero at times. These alternate variations of loops, groups, or segments must be consecutively defined in the SEF file. For example, if you are using two variations of the N1 loop, one must immediately follow the other.

The following piece of a SETS section shows two variations of the N1 loop, both using the same position numbers, and then the next loop (an unused LM). On this page, each loop is underlined separately so that you can easily see where each loop starts, but underlining is not actually included in the SEF file.

```
{:1+10[N1][N2,,2][N3,,2][N4,,>1]+5[.NX2,,3][.REF,,12]+10[.PER,,>1][.FOB][.T
D1,,2][.TD5,,12][.TD3,,12][.TD4,,5][.PKG,,200]}{:1-110[N1@166]+10[N2,,2]
+20[N4,,>1]}{:>1+90[.LM@51]+10[LQ,M,>1]}
```

Assuming the last used sequence number before the first N1 is 300, the position numbers of the first N1 loop are:

```
N1=310        FOB=370
N2=320        TD1=380
N3=330        TD5=390
N4=340        TD3=400
NX2=345       TD4=410
REF=350       PKG=420
PER=360
```

The second variation of the N1 loop must have the same position numbers for the segments they include, so that they match the position numbers of the underlying ASC X12 850. (You will have to explicitly include ordinals for the inserted segments; each segment must have a unique ordinal even though the position numbers are being reused.):

```
N1=310   An out-of-sequence ordinal (in this example, 166) has to be included.
N2=320   Ordinal is assumed to be 167
N4=340   Ordinal is assumed to be 168
```

Here is how the position numbers are adjusted. The last used position number (for PKG) was 420. The increment is therefore reset to -110 (420-310) for the N1. Now that the N1 is at 310, the SEF code needs to reset the increment to +10 for the N2. Since the N3 is omitted entirely (rather than not-used), manually increment it to +20 so that the N4 can be at 340.

The LM loop must have these position numbers to match the X12 850:

```
LM=430        The LM's ordinal must be included so it won't be assumed to be 169
```

```
LQ=440
```

The LM must be preceded with an increment of +90 (430-340). Finally, the increment is reset to +10 so that the LQ will be at 440.

Zero position increments: Consecutive variations of the same segment can be coded with position number increments of +0 or -0:

```
[REF,M]+0[REF*2@179,,>1]+10[PER@6,,3]
```

If the first REF had a position number of 50, then the second REF would also. The PER would have a position number of 60, since the increment was reset to +10.

New EDIFACT Standards and Position Numbers

Starting with D96A, new EDIFACT standards have these differences in sequence numbers:

- Sequence numbers continue ascending through table boundaries, rather than starting over in each new table. In X12 standards, and older EDIFACT standards, sequence numbers start over at each new table.

- Groups participate in the numbering.

A SEF file containing a new EDIFACT standard needs the **.STD ,LS** record at the front of the file. The space before the comma is necessary.

```
.VER 1.6
.INI
SEF-E961,,D 96A,UN,D96A,Kaver Co. ORDERS message 96A - publ. Mar. 1996
.PRIVATE FORESIGHT
.PUBLIC
```
`.STD ,LS`
```
.SETS
```

The individual .SETS records do not appear different, but they will be interpreted differently. In the following section of a standard, the groups have position numbers:

```
0010  UNH
0020  BGM
0030  DTM
0040  PAI
0041  Group 1 Starts
0042  RFF
0043  DTM
      Group 1 Ends
0050  ALI
0060  IMD
0070  FTX
0080  Group 2 Starts
0085  RFF
0100  DTM
      Group 2 Ends
```

This section of the standard would appear as follows in the SEF file. Notice the .STD ,LS before the SETS section and the changing position numbers before the groups as well as before segments. The position number increments here are shown in large, bold type for emphasis, but the font changes are not actually in the SEF file:

ORDERS=^[UNH,M][BGM,M][DTM,M,35][PAI]**+1**@213{:10[RFF,M][DTM,,5]}**+7**[ALI@5,,5]

+10[IMD][FTX,,99]{:10**+5**[RFF]**+15**[DTM,,5]}

The following section of a standard has a consistent position number increment from 0910 in Table 1 to 0950 in Table 2:

```
.
.
.
0910  DTM
0920  FTX
---Table 2---
0930  Group 26 Starts
0940  LIN
0950  PAI
.
.
.
```

Therefore, no adjustments for position numbers are needed in this part of the SEF file:

[DTM,,5][FTX,,5]^{:200000[LIN][PIA,,25]   ...*etc.*

Let's take the same section of the standard, but change the position number of the FTX to 915.

.
.
.
```
0910  DTM
0915  FTX
---Table 2---
0930  Group 26 Starts
0940  LIN
0950  PAI
```

Since the position number increment varies now, this will require adjustments in the SEF file position numbers:

[DTM,,5]**+5**[FTX,,5]**^+15**{:200000**+10**[LIN][PIA,,25]   *... etc.*

These adjustments would be the same whether the table changed (^) or not.

Summary of Symbols used in .SETS

[]	encloses each segment, its requirement, quantity
{}	encloses loops
+ or - (followed by number)	changes position number increment
:	loop repeat count
^	new table
*	mask number
@	ordinal number (if out of order)
. (before segment ID)	marked "not used" in the convention's transaction set or message
! (before segment ID)	marked "mandatory" in the convention's transaction set or message
$ (before segment ID)	"recommended" in the convention's transaction set or message
- (before segment ID)	"not recommended" in the convention's transaction set or message
& (before segment ID)	"dependent" in the convention's transaction set or message
M, O, F, or C	requirement designator in the originating standard's transaction set or message

The .SEGS Section

The .SEGS section is the standard's segment dictionary: a list of all segments in the standard. The segments are usually, but not always, in alphabetic order by segment ID. Each record includes the segment ID followed by a list of each composite and element reference it contains, in order. The following is part of an example .SEGS section:

```
.SEGS
A1=[131,M][126,M][44][43,M]
A2=[329,M]
A3=[329][126,M][44][206,M][207,M][128][127][647][717,M]+P0607
A4=[329][206,M][207,M][128][127][186][373][86]+P0405
AAA=[900,M][559][901][889]
AC=[23,M][22,M][635,M][692,M]{2[309,M]}{2[156][310]}[498][499]{2[373]}{2[140]}+R0708R
0910
ACK=[668,M][380][355][374][373][326]{10[235][234]}[559][822][1271]+C0203C0405C0708C09
10C1112C1314C1516C1718C1920C2122C2324C2526P2728C282729
```

The first line shows a typical segment:

```
A1=[131,M][126,M][44][43,M]
  ↑   ↑           ↑
  1   2           3
```

1. The first part is the segment ID followed by an = sign.

2. This is the number of the first element in the segment, followed by its requirement according to the published standard. If it is an EDIFACT standard, it can be **M** (mandatory) or **C** (conditional, meaning it is optional). If it is an X12 standard, it can be **M** (mandatory), **O** (optional), **C** (conditional, meaning it is the dependent of an element relational in an older standard), or **X** (meaning it is a dependent of element relational in a newer standard). It may be omitted if the element uses the default requirement of **O** (X12) or **C** (EDIFACT). If the element omits the requirement designator, but it is the dependent of an element relational, then **X** is assumed. You can see an example of an implied X requirement designator in the section on Syntax Rules for Segments on page 21.

3. The third element in the example segment omits the requirement, thereby using the default (O for ASC X12).

The format for an element or composite reference in the .SEGS section is:

| [*element,requirement,count*] | Examples: `[128]` `[206,M]` `[2345,M,5]` `[309,,2]` |
| [*composite,requirement,count*] | Examples: `[C002]` `[C002,M]` `[C002,,10]` |

requirement is the requirement according to the published standard: M, O, C, X. If an element or composite requirement is other than O (optional) for X12 or C (conditional) for EDIFACT, then it is included.

count is the element repeat count. This is not the same as the repeating pattern of elements described below, which is simply a shorthand way for SEF to represent a fixed number of consecutive identical elements.

The *element* section extends until the first comma, and can include several parts. These are, in order:

$$[\textit{element}@\textit{ordinal};\textit{min}:\textit{max},\textit{requirement},\textit{count}]$$

element	The element number.	`127`	
ordinal	Normally, ordinals within a segment start with 1 for the first element and increment by 1 for the next element. They can be omitted unless there is an out-of-sequence ordinal, caused by inserted or deleted elements in the segment. Out-of-sequence ordinals are included, and preceded with an @-sign. In the example, an element has been inserted into or deleted from this segment in the segment dictionary. Element 127's ordinal is out of sequence and must therefore be specifically set. See Ordinal Numbers in the .SEGS Section on page 20.	`127@6`	*@n* where *n* is the out-of-sequence ordinal number

min:*max*	The element's minimum and maximum length, if changed from that of the underlying standard. In this example, element 127's min has been changed to 5 and its max has been changed to 10. The semi-colon acts as a separator before the min and max.
	
	In the first example, the ordinal is out of sequence, the min has been changed to 5 and the max has been changed to 10.
	
	In the second example, the ordinal is again out of sequence, and the max has been changed. Since the min has not been changed, it can be omitted.
		`127@6;5:10`
		
		`127@6;:10`
			;*min***:***max*
			
			where:
			
			; separator preceding the min and max section.
			
			min minimum length (changed from underlying standard)
			
			: separator preceding max.
			
			max non-default maximum length (changed from underlying standard)

Full example: `[127@6;5:10,M,3]`			Element 127 has ordinal 6. Its minimum has been changed to 5 and its maximum has been changed to 10. It is Mandatory and has an element repeat count of 3.

User Requirement in the .SEGS Section

If the SEF file describes a convention (customized for a particular trading relationship) rather than a published standard, the .SEGS section can include the requirements specified by the particular convention, as well as the requirements specified by the published standard on which it is based. This information is indicated by a symbol before the element ID within the square brackets.

This line is from the segment dictionary of an EDIFACT published standard. Therefore, there are no user requirements:

```
TDT=[8051,M][8028][C220][C228][C040][8101][C401][C222][8281]
```

This line is from the segment dictionary of a convention based on the above published standard. User requirements have been added to the TDT in the segment dictionary:

```
TDT=[8051,M][.8028][.C220][$C228][-C040][8101][C401][&C222][!8281]
```

where:

`[8051,M]`	the absence of any user requirement means to use the requirement of the published standard. Elements and composites with a requirement of M are always considered mandatory, regardless of any user requirement to the contrary.
`[.8028]`	the dot indicates that this element is not used
`[.C220]`	the dot indicates that this entire composite is not used
`[$C228]`	the $ indicates that this composite is recommended
`[-C040]`	the minus indicates that this composite is not recommended
`[8101],` `[C401]`	the absence of any user requirement means to use the requirement of the published standard, which is "O" as implied by the absence of a requirement designator.
`[&C222]`	the ampersand means this element or composite is dependent. This element or composite might be used in a semantic rule in the .SEMREFS section, but it can be dependent without semantic rule involvement.
`[!8281]`	the exclamation mark means this element must be used, even though the requirement designator is "O" by default.

Ordinal Numbers in the .SEGS Section

Please refer to Ordinal Numbers in the .SETS Section on page 13, which introduces the concept of ordinal numbers. The .SEGS section also has them, and they follow the same rule: only ordinal numbers for out-of-sequence elements are shown. It would be unusual to see out-of-sequence ordinals in .SEGS and .COMS because conventions would generally not change the layout of a segment or composite.

```
CUR=[98,M][100,M][280][3@22][98@4][100][669][374][373][337][374][373]   (etc.)
        ↑      ↑     ↑    ↑     ↑    ↑     ↑     ↑     ↑     ↑     ↑     ↑
        1      2     3    22    4    5     6     7     8     9    10    11
```

The numbers below the partial example above show each element's ordinal number. Normally, the first element in each segment will have the ordinal number 1, the second one will have 2, etc. The default increment is 1. If an element is inserted or deleted (as opposed to marked as not used), then the flow is disrupted. Whenever the flow is disrupted, the ordinal number will appear with its element, preceded by an @ sign. Element 3 received ordinal number 22, which was the first available number (the CUR had 21 elements already). Element 98 at CUR05 had to show its ordinal number to re-establish the flow.

Repeating Patterns of Elements

This partial line contains two repeating patterns, enclosed in curly brackets:

```
AC=[23,M][22,M][635,M][692,M]{2[309,M]}{2[156][310]}
                                   ↑          ↑
                                   1          2
```

1. `{2[309,M]}` is equivalent to `[309,M][309,M]` and may appear that way in the SEF file.

2. `{2[156][310]}` is equivalent to `[156][310][156][310]` .

The repeating patterns can be nested, as in this PR1 segment from X12 3041:

```
PR1=[23,M][22,M][22]{2[309]{2[310]}[156][140]}
```

This is equivalent to:
```
PR1=[23,M][22,M][22]{2[309][310][310][156][140]}
```

which is equivalent to:
```
PR1=[23,M][22,M][22][309][310][310][156][140][309][310][310][156][140]
```

Element Repeat Counts in .SEGS

Element repeat counts are included as a third parameter for elements in the segment dictionary and subelements in the composite dictionary. If omitted, the default is assumed to be 1.

In the following examples, the repeat counts are shown larger for emphasis.

Example segment dictionary entry in the .SEGS section

```
XYZ=[C817][3164][9208,M][3207,M,4][3456,,5]
```

Segment XYZ has two elements with repeat counts. Element 3207 is mandatory and can be repeated up to 4 times in segment XYZ. Since element 3456 is conditional, its requirement can be omitted. The requirement's place must be held with commas, as shown, if a repeat count follows.

Syntax Rules or Dependency Notes for Segments

This section also houses the element relationals (syntax rules or dependency notes) for each segment that has them. Segment A4 has a syntax rule:

```
A4=[329][206,M][207,M][128][127][186][373][86]+P0405
                        ↑     ↑                   ↑
                        04    05             syntax rule
                                               (paired)
```

A single plus sign introduces any number of relationals. The code at the end, P0405, is a paired syntax rule: if either A404 or A405 is present, then the other must be also. A404 is the element in the 4th position in A4 − which is element 128. A405 is element 127. Since the requirement designator is omitted for [127] and [128], and since the presence or omission of either element depends on that of another element, the implied requirement designator is **X** for both elements.

Syntax rules and dependency notes refer to elements by their position in the segment. In the example above, P0405, these would be 04 and 05. These are not ordinal numbers. If an element is inserted at the beginning of the A4 segment above, the syntax rule would change to P0506.

Example X12 Syntax Rules		
Required	R0708R0910	At least one of 07 and 08 is required, and at least one of 09 and 10 is required.
Conditional	C0102C030405	If 01 is present, then 02 is required. If 03 is present, then both 04 and 05 are required.
Exclusion	E060810	Only one of 06, 08, or 10 may be present.
List Conditional	L060810	If 06 is present, then at least one of 08 or 10 is required.
Paired	P060708	If any of 06, 07, or 08 are present, then all are required.

Example EDIFACT Dependency Notes		
One and only one	D1(010,020,050)	Exactly one of the items at position 01, 02, and 05 must be present.
All or none	D2(010,020)	The items at 01 and 02 are both included or both excluded.
One or more	D3(010,030,040)	At least one of the items at 01, 03, and 04 must be included.
One or none	D4(030,040,050)	At most, one of the items at 03, 04, and 05 can be included, or all may be excluded.
If first, then all	D5(030,040,050)	If the first item (at position 03) is present, then all of the others (at 04 and 05) must be present. However, if items at 04 and 05 are present, the first need not be present.
If first, then at least one	D6(030,040,050)	If the first item (at position 03) is present, then at least one of the others (at 04 and 05) must be present. However, if the item at 04 and/or 05 is present, the first need not be present.
If first, then none	D7(030,040,050)	If the first item (at position 03) is present, then the items at 04 and 05 cannot be included.

The end of a segment may combine different rules.

X12 example: R0708C0910 combines a required (R0708) and a conditional (C0910).

EDIFACT example: D3(010,020)D7(030,040,050) combines a D3 "one or more" with a D7 "if first, then none."

'Masks' or Alternate Variations of a Segment

If a segment in a transaction set or message differs from the dictionary, then a mask will be used in the SEF file. If no mask number is used, then the dictionary definition is used. Otherwise, the variation of the segment described by the mask is used. Masks apply to a specific segment in a specific location in the convention. A typical use of a mask would be to show that certain elements in a segment or composite are unused at this location, even though the dictionary segment or composite shows all of the elements used.

Mask Basics

Let's say we have an implementation guideline based on an X12-3041 850. We are using three variations of the PER segments:

Segment	Element Changed	Changes
PER at 060	(no changes from dictionary)	
PER at 360	PER02	Mandatory Max 30
	PER04	Min 40 Max 70
	PER05	Not Used
PER at 110, Table 2	PER02	Min 5
	PER03	Must be Used
	PER04-PER06	Not Used

There is only one PER segment defined in the .SEGS section, and it has to accommodate all three variations above. That is where masking helps.

To see if a mask is being used, and, if so, which one, look in the .SETS section. For our PER example, these codes will show up in the appropriate position in .SETS:

The PER in position 060:   [PER,,3]   This PER was unchanged from the dictionary, so no mask shows up here.

The PER in the N1 :   [PER**1**,,3]   This PER uses mask 1, as shown by the ***1** .

The PER in the P01's position 120:   [PER**2**,,3]   This PER uses mask 2, as shown by the ***2** .

The .SETS section shows if a mask is being used, but it does not show the mask itself. For that, you need to look in the .SEGS section. In our example, we can see the following under .SEGS:

```
                                                         describes particular instance of
      describes segment in dictionary                      segment in transaction set


                                                         mask 1            mask 2
                                                            ↓                 ↓
PER=[366,M][93][365][364][365][364]+P0304P0506,..M[:30]..[40:70]#.,..[5]@
       ↑                                         ↑   ↑   ↑    ↑   ↑ ↑   ↑  ↑
366 is mandatory in originating                  1   2   3    4   5 6   7  8
standard's dictionary
```
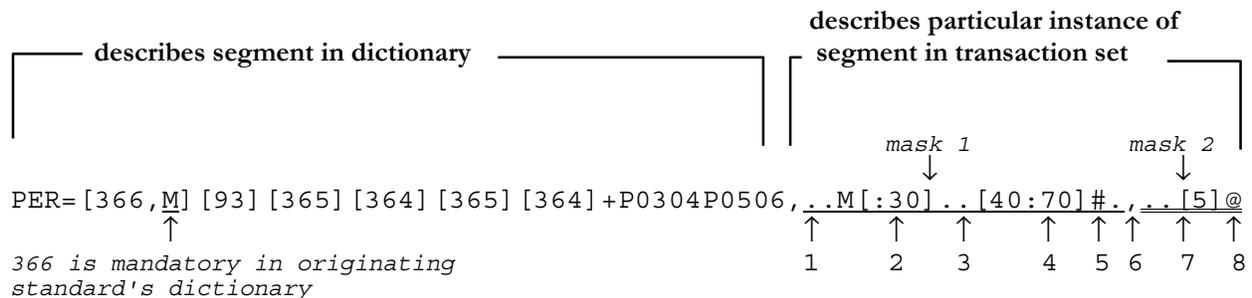
The PER is shown as it occurs in the dictionary, and any variations of PER in the transaction set show up as masks at the very end. **Each mask is preceded by a comma**. In this mask, each element in the segment is represented by one of these symbols, which represent how it is used in the convention:

.   (dot) inherit the requirement from the segment dictionary (normally this means used). See "The Dot Versus the Plus Sign" below.

not used

@   must be used

These symbols could have appeared in the mask, also:

$ recommended

-       not recommended

&   dependent (the usage might depend on a semantic rule set up in the SEMREFS section, but this is not always the case).

+   used, regardless of segment dictionary's requirement for this element or composite. See **The dot versus the plus sign** below.

*n composite mask (example: **..*2** means the first item is used (it may be a composite or an elementary data element) and the second item is a composite using composite mask 2.)

If an element's min, max, or req. des. have been changed at a particular location in the transaction set, the dot (.), at-sign (@), plus sign (+), dollar sign ($) or minus sign (-) will be followed by characters representing the changes.

As we saw above, the PER in the N1 is using mask 1. It can be interpreted as follows:

1.   The first dot after the comma represents the first element, PER01 (366). Since it is a dot, it inherits the user requirement from the dictionary and therefore that element is to be used in the transaction set at the location that uses this mask.

2.   The second dot represents the second element, PER02 (93). Since it is a dot, that element also inherits the user requirement from the dictionary, and therefore it is to be used in the transaction set at the location that uses this mask. However, right after the PER02 dot, we see M[:30], which describes some changes to PER02. The **M** is the originating standard's requirement for PER02 in the PER at the specific location in the transaction set where this mask is used – which differs from its requirement in the originating standard's **dictionary**. The **[ ]** enclose changes to the Min length, Max length, and repeat count, with a colon separating each: [*min*:*max*:*repeat*]. Since Min and repeat count are omitted, they were not changed. Max was changed to 30.

3.   The third dot means PER03 (365) inherits the dictionary's user requirement and is to be used.

4.   The fourth dot means PER04 (364) inherits the dictionary's user requirement and is to be used. Since it is followed by [40:70], its Min and Max differ from that in the dictionary. Its Min is 40 and its Max is 70. Since there is no colon and repeat count after the Max, the repeat count is 1.

5.   The next symbol is #, meaning PER05 (365) is not used in the PER at this location in the transaction set, according to the convention. The symbol after that is a dot, meaning PER06 (364) is to be used.

6.   The next symbol is a comma, meaning mask 1 has ended. This concludes all changes to any PER segments in transaction sets that were using mask 1. In our example, this means the PER at position 360.

Mask 2 starts after the second comma. As we saw above, the PER at 110 in Table 2 is using mask 2. It can be interpreted as follows:

7.   The first symbol is a dot, which means PER01 (366) inherits the user requirement from the dictionary and therefore is being used. The second symbol is a dot, which means PER02 (93) is being used. This element has a change: [5]. Square brackets enclose changes to Min and/or Max. In this case, the Min has been changed to 5. Since there is no colon or additional number, the Max has not been changed.

8.   The next symbol is an @ which means PER03 must be used, according to the convention, in the PER at this location in the transaction set. Since no requirement or Min/Max changes follow it, that is the only change to

---

PER03. What about PER04-PER06? We might have been expecting ###, since the last three elements are unused. However, if all remaining elements are unused, they are omitted from the mask.

Two consecutive commas:

This indicates an empty mask, which means all element positions are marked as not used. This would generally be meaningless, as you would mark the segment as not used instead.

The dot versus the plus sign:

A plus sign in a mask means "this is used regardless of what the segment dictionary says." A dot in a mask means "look at the segment dictionary to see whether this is used." Normally, this would mean that it is used. Example:

`PER=[366,M]` **`[-93][-365]`**`[364][365][364][365][364][443]+P0304`**`,.+.......`**

The segment dictionary shows the second and third elements, 93 and 365, as "not recommended" (notice that they are preceded by minus signs). The mask shows, in order:

- The first dot in the mask means the user requirement for element 366 is as shown in the segment dictionary. The absence of any symbol in front of 366 means that it is used, and therefore it will be used in the mask also.

+ The plus sign means that element 93 is used in the mask. It overrides the dictionary, which shows

 [-93], indicating that 93 is not recommended here.

- The next dot indicates that the user requirement for element 365 should be inherited from the segment dictionary. Since the segment dictionary shows it as [-365], this means it is not recommended in this mask.

- The next dot indicates that the user requirement for element 364 should be inherited from the segment dictionary, where it is used.

Composite Masks

Like segments, composites contain a list of element references; therefore they can also have masks.

This MEA has a mask because the third element (739) has been designated as Must be Used in a convention's transaction set. It does not have a composite mask for C001, the 4th item, since no subelement was changed:

`MEA=[737][738][739][C001][740][741][935][936][752][1373]+R030506,..@.......`

Element 739

The mask below indicates that the fourth element (C001 − the composite) requirement has been changed to M; there is still no composite mask required because no subelement has been changed:

`MEA=[737][738][739][C001][740][741][935][936][752][1373]+R030506,`**`....M......`**

Now, let's make these changes to the elements in the convention's transaction set:

MEA03 (739):   Min changed to 2, Max changed to 8
MEA04 (C001):   second and third subelements changed to Must be Used

In the .SEGS section, we would have:

`  MEA=[737][738][739][C001][740][741][935][936][752][1373]+R0305060,`**`...[2:8].*1......`**
```
                                                                      ↑   ↑  ↑↑    ↑
                                                                      1   2  3 4   5
```

1. We can see the three dots for 737, 738, and 739, respectively. This means the user requirements are inherited from the dictionary (where they are used, since they appear within square brackets in the dictionary portion of the line with no symbol in front of them).

2. After the third dot, for 739, we see that its Min and Max have been changed to 2 and 8.

3. The fourth dot for C001 means that, since C001 is used in the dictionary, it will be used in this mask also.

4. After C001's dot, we see: ***1** . This means that C001's mask 1 is used here. It modifies the way C001 will appear, as opposed to the C001 from the composite dictionary.

5. After that, we see dots for the last six elements in the segment: 740, 741, etc. Subelements are not listed separately in the segment dictionary.

We can look up C001's mask 1 in the .COMS section, where we see:

```
C001=[355,M][1018][649][355][1018][649][355][1018][649][355][1018][649][355][1018][649],.@@.........
...
```

We can tell by the comma delimiters that C001 has only one mask: mask 1, which was referred to by *1 within a segment mask. It says:

- first subelement (355) in composite: no change from composite dictionary usage
- second subelement (1018):   Must be Used, according to the convention
- third subelement (649):   Must be Used, according to the convention

The symbols used in a composite mask are just like those in a segment mask. If we had changed the Min or Max for one of the subelements to 2 and 8, we would have seen the [2:8] after the dot or after the @ , ! , $, or - that represented that element position. If we had changed the requirement to M for a particular instance of a composite in a convention, we would have seen the M after the dot or other usage symbol.

Summary of Symbols used in Segment Masks

Segment masks describe elements in segments at a particular location in the transaction set or message.

,	(comma) delimiter before each mask.
.	(dot) the user requirement for this element or composite is unchanged from that in the segment dictionary.
#	This element or composite is not used in the convention's transaction set or message.
@	This element or composite must be used in the convention's transaction set or message.
$	This element or composite is recommended in the convention's transaction set or message.
-	(minus sign) this element or composite is not recommended in the convention's transaction set or message.
&	This element or composite is dependent in the convention's transaction set or message. Sometimes the dependency is defined in semantic rules (see .SEMREFS).
+	This element or composite is used, regardless of what the dictionary says for this item earlier in the record.
[*min*:*max*:*repeat*]	This element's minimum length, maximum length, or repeat has changed from that in the segment dictionary. Any of the three parts can be omitted as long as enough colons are included to maintain the positions.
M	The requirement in the originating standard's transaction set or message is mandatory.
O	The requirement in the originating standard's transaction set or message is optional.
C	The requirement in the originating standard's transaction set or message is conditional.
X	The requirement in the originating standard's transaction set is X.
*n	This composite contains a mask (explained in Composite Masks on page 25).

Summary of Symbols used in .SEGS (Exclusive of Masks)

These symbols describe elements in a dictionary segment.

@	Out-of-sequence ordinal number for element or composite: [86@5] means element 86's ordinal number is 5.
[ ]	Surrounds an element or composite: [329][206,M] show two elements: 329 and 206.
M	Mandatory: [206,M] means element 206 is mandatory in the originating standard's dictionary.
C	Conditional in the originating standard's segment dictionary.
O	Optional in the originating standard's segment dictionary.
X	Dependent of element relational (generally in X12 only).
;	Precede a changed min and/or max length with a semi-colon: [127;5:10] means the new min is 5 and the new max is 10.
:	Precedes the changed maximim length. See explanation for semi-colon.
{ }	One or more repeating elements or composites: {2[22,M]} is equivalent to [22,M][22,M].
+	Beginning of syntax rules (see Syntax Rules or Dependency Notes for Segments on page 21).
,	Beginning of a mask (see 'Masks' or Alternate Variations of a Segment on page 23).
.	(Before element or composite ID) marked "not used" in the convention's segment dictionary.
!	(Before element or composite ID) marked "must be used" in the convention's segment dictionary.
$	(Before element or composite ID) marked "recommended" in the convention's segment dictionary.
-	(Before element or composite ID) marked "not recommended" in the convention's segment dictionary.
&	(Before element or composite ID) marked "dependent" in the convention's segment dictionary.
*n	(After element or composite ID) uses a composite mask described under that composite in the .COMS section. Example: [C001*1] uses mask 1 on the C001 composite.

The .COMS Section

If your standard has composites, the SEF file has a .COMS section. It is the standard's composite dictionary: a list of all composites in the standard. It includes the composite ID followed by a list of each subelement reference it contains, in order.

The following is an example .COMS section (the C001 line is wrapped in this example, but it is actually one long record in the file). The **n/l** means new-line:

```
.COMSn/l
C001=[355,M][1018][649][355][1018][649][355][1018][649][355][1018][649][355
][1018][649]+P0203,.@@.............,.@@...[2:8].........n/l
C002=[704,M][704][704][704][704]n/l
C003=[235,M][234,M]{4[1339]}[352]n/l
C004=[1328,M]{3[1328]}n/l
C005=[1369,M]{4[1369]}n/l
C006=[1361,M]{4[1361]}n/l
C999=[1361,M,4][9200,,7]}n/l
```

It resembles the .SEGS section, with each subelement reference listed in the order in which it appears in the composite. The format of each subelement reference is:

> **[***userattributeID***@***ordinal***;***min***:***max***,***requirement designator***]**

Example:

> `[&704@3;4:8,X]`

where:

userattribute	&
ID	704
ordinal (preceded with @)	*3*
min (preceded with ;)	4
max (preceded with :)	8
requirement (preceded with ,)	X

Information inside the square brackets:

- User attributes (. ! $ - &), if specifically set by the guideline author, appear before the subelement ID. In this example, the subelement has been marked as Dependent:

  `[&704]`

  See Summary of Symbols used in .COMS on page 30 for a list of user attribute symbols.

- The subelement's ordinal within the composite, if it is out of sequence, appears after the ID and is preceded with an at-sign. See Ordinal Numbers in the .COMS Section on page 29.

- Minimum and maximum length changes follow the subelement ID and ordinal (if included) and are preceded with a semicolon. The minimum is separated from the maximum with a comma. Examples:

  > Minimum length changed to 2: `[704;2]`

  > Maximum length changed to 3: `[704;:3]`

  > Minimum length changed to 4 and maximum length changed to 8: `[704;4:8]`

  > Minimum length changed to 4 and maximum length changed to 8, user attribute changed to Dependent, and requirement designator changed to X: `[&704;4:8,X]`

- If a subelement's requirement is other than O (optional) for X12 or C (conditional) for EDIFACT, then it is included at the end, before the closing square bracket. It is preceded with a comma: `[704,M]`.

After the list of subelement references:

- Any element relationals (syntax rules or dependency notes) for the composite follow the last subelement and are preceded with a `+` sign: `+P0203`. See Syntax Rules and Dependency Notes in the .COMS Section on page 29.

- If the composite has mask(s), they appear at the very end, separated by commas:

  `@O@.[3:3]..[2:8].........`

  These are explained in 'Masks' or Alternate Variations of a Segment on page 23 and Composite Masks on page 25.

There is no element repeat for subelements within composites.

Ordinal Numbers in the .COMS Section

Please refer to Ordinal Numbers in the .SETS Section on page 13, which introduces the concept of ordinal numbers. The .COMS section also has them, and they follow the same rule: only out-of-sequence ones appear in the SEF file.

```
                                    this one was inserted

   C001=[355,M][1018][649][3@17][355@4][1018][649]   (etc.)
          ↑        ↑      ↑  ↑      ↑        ↑     ↑
          1        2      3  17     4        5     6
```

The numbers below the partial example above show each subelement's ordinal number. Normally, the first subelement in each composite will have the ordinal number 1, the second one will have 2, etc. If a subelement is inserted or deleted (as opposed to marked as not used), then the flow is disrupted. Whenever the flow is disrupted, the ordinal number will appear with its subelement, preceded by an `@` sign. In the partial example above, the fourth subelement (element ID 3) was inserted. It received ordinal number 17, which was the first available number (the C001 had 16 elements already). Subelement 355 had to show its ordinal number to re-establish the flow.

Repeating Patterns in the .COMS Section

For efficiency, repeating patterns of subelements may be enclosed in curly brackets and include a multiplier:

`C001=[355,M]{4[1018][649][355]}` (*etc.*)

This composite definition contains a set of three subelements that are enclosed in { } and have a multiplier of 4. It is equivalent to:

`C001=[355,M][1018][649][355][1018][649][355][1018][649][355][1018][649][355]`

Syntax Rules and Dependency Notes in the .COMS Section

Syntax rules show up in the .COMS section just like those in the .SEGS section (see Syntax Rules or Dependency Notes for Segments on page 21). Example:

`C002=[704][704][704][704][704]+C0102`
```
                                    ↑
                                syntax rule
```

This is a conditional syntax rule: If C002-01 is present, then C002-02 is required.

Summary of Symbols used in .COMS

Outside of masks (refers to dictionary)

These symbols can appear within the square brackets

[ ]	Enclose a subelement.
{ }	Enclose a repeating pattern of subelements.
.	(Before subelement ID) marked "not used" in convention's composite dictionary.
!	(Before subelement ID) marked "mandatory" in convention's composite dictionary.
$	(Before subelement ID) marked "recommended" in convention's composite dictionary.
-	(Before subelement ID) marked "not recommended" in convention's composite dictionary.
&	(Before subelement ID) marked "dependent" in convention's composite dictionary.
@	Delimiter preceding an ordinal subelement number that is out of sequence.
;	Delimiter preceding the mininum and maximum length changes.
:	Separator between minimum and maximum length.
,	Delimiter preceding requirement designator (M, O, C, X as explained below).
M	Mandatory, according to the originating standard's dictionary.
O	Optional, according to the originating standard's dictionary.
C	Conditional, according to the originating standard's dictionary.
X	Dependent of element relational in originating standard's dictionary.

Within masks (refers to particular location in the particular segment using the mask)

,	Delimiter before each mask.
.	The usage is the same as that in the composite dictionary (earlier in same line).
#	This subelement is not used in this location in the segment using this mask.
@	This subelement must be used in this location in the segment using this mask.
$	This subelement is recommended in this location in the segment using this mask.
-	This subelement is not recommended in this location in the segment using this mask.
&	This subelement is dependent at this location in the segment using this mask.
+	This subelement is used in this location in the segment using this mask, regardless of what the dictionary says (earlier in the same line).
[*min*:*max*]	The minimum and/or maximum has been changed in this location in the segment using this mask. Either part may be omitted, and the colon can be omitted if the max is not changed.
M	The requirement has been changed to mandatory in this location in the segment using this mask. M generally refers to the requirement according to a published standard, not a convention.
O	The requirement has been changed to optional in this location in the segment using this mask. O generally refers to the requirement according to a published standard, not a convention.
C	The requirement has been changed to conditional in this location in the segment using this mask. C generally refers to the requirement according to a published standard, not a convention.
X	The requirement has been changed to X in this location in the segment using this mask. X means this item is the dependent of element relational, and generally refers to the requirement according to a published standard, not a convention.

The .ELMS Section

The .ELMS section is the standard's data element dictionary. It is a list of each element, its type, and its minimum and maximum usage, and is usually sorted by element ID:

```
.ELMS
1=AN,1,13
2=N0,1,6
3=AN,1,60
4=ID,3,3
5=ID,3,5
6=DT,6,6
.
.
.
```

The fields are, in order:
```
                   5=ID,3,5
                   ↑  ↑ ↑ ↑
                   1  2 3 4
```

1. The element number followed by an = sign.
2. The type: `R, ID, AN, A, DT, TM, B, N, N0, N1 - N9.`
3. The minimum length.
4. The maximum length. If max length is specified as **0** it is understood to mean not applicable, or no limit.

The .CODES Section

The .CODES section is a numeric list of each element and its dictionary code values. Elements without code values are not listed. Any element may have code values defined; in ASC X12, such elements are usually but not always ID type (example: data element 103). EDIFACT elements with code values are usually type A or AN. Example beginning of the .CODES section:

```
.CODES
8=E,R
9=01,02
11=A,B,C,D,E,F,G,H,J,M,P,Q,R,S,T,U,V,W,X,Y
16=A,B,C,D,E,P
23=2,A,B,C,D,E,F,G,H,I,J,K,L,N,P,Q,S,T,U,V,Z
```

Code Sets

This section also provides information about code sets. If the implementation guidelines use codes just as they are in the dictionary, element 52's code values might look like this:

```
52=1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17
```

Now let's say the implementation guidelines differs from the dictionary. Its codes are 1,2,3,4,5,15,16,17,18,19,20,50 for element 52. The SEF file .CODES section might contain something like this:

```
          1       2a            2b                   2c           3     4     5
          ↓       ↓             ↓                    ↓            ↓     ↓     ↓
52=1:9,10:17%[(1)1,2,3,4,5,15,16,17{18,19,20,50}]+850/119///5
                  └──────────────────────────────────────┘
                                    2
```

1. The first section lists the dictionary code values. To conserve space, ranges of codes are listed with a colon, so that 1:9 would mean all codes from 1 through 9. To use the colon with ranges, the codes must be the same length. A1:B5 would represent A1, A2, A3, A4, A5, A6, A7, A8, A9, B0, B1, B2, B3, B4, B5. The % marks the end of the dictionary codes.

2. The second section lists all codes allowed in the implementation guidelines. It is enclosed in [ ] .

    2a. This is the code set ordinal, enclosed in ( ) . This is only needed if one of the .TEXT sections needs to refer to one of the codes. This would be true, for instance, if a local code were added in the transaction set or message, and it included a description. Otherwise, it can be omitted.
    2b. This is the list of dictionary codes that are being used.
    2c. This is the list of codes that have been added to the implementation guidelines. They are enclosed in { } to show that they are not in the element dictionary. If the {} are omitted or empty, no codes have been added anywhere in the standard that are not listed for this element in the element dictionary.

3. This is the transaction set that uses the codes, preceded with a + sign.

4. This is the ordinal number of the segment that uses this code set. Ordinal numbers are explained on page 13.

5. This is the ordinal number of the element that uses these codes. (The /// will always appear if the code set is attached in the transaction set or message. If it is attached in the dictionary, the entire part after the plus sign will be different; see Code Sets Attached in Dictionary on page 36 for details).

In the following simple example, the first element in the segment with ordinal 2 in transaction set 856 is only allowing dictionary codes 00 and 07 and doesn't add any codes that are not in the element dictionary. Notice that no code set ordinal is needed.

```
353=00:08,10:22,24:28,30:38,41:54,5C,77,CN:CO,EX,PR,RH,RV,SU,ZZ%[00,07]+856/2///1
```

Multiple Code Sets for One Element

If the implementation guidelines has two places where an element is used, the .CODES section might contain something like this:

```
52=1:9,10:17%[(1)1,2,3,4,5,15,16,17{18,19,20,50}]+850/119///5[(2)1,2,3,4,5{99}]+850/120///5
```

The part that describes the second code set is shown here in bold. Like the first code set, it is enclosed in square brackets. It allows values 1-5 and 99. Immediately after that, we see where it is used: transaction set 850, segment with ordinal number 120, element with ordinal number 5 in that segment

Same Code Set Used in Multiple Places

Let's say an 850's REF segment is used three times with the following codes in REF01 (element 128):

Segment	Codes	Codes in Dictionary?
REFs at 050 and 350	5, 10, 15	(in dictionary)
	ZX	(not in the dictionary**)**
REF at 100, Table 2	01, 02, 03	(in dictionary)
	ZY	(not in dictionary)

The last part of element 128's entry in the .CODES section would look like this:

```
[(1)05,10,15{ZX}]+850/5///1+850/38///1[(2)01,02,03{ZY}]+850/57///1
        ↑               ↑           ↑               ↑                ↑
        1               2           3               4                5
```

1. The first section, enclosed in square brackets, starts with (1) so it is the first code set with an ordinal for element 128. This means it is the first code set that has text that is not from the standard. That text would be the description for the new local code, ZX. Inside the square brackets we can see the three codes from the dictionary and the ZX code, which is not in the dictionary.

2. The second section, which starts with a +, shows one place where this code set is used: transaction set 850, the segment with ordinal number 5, the element with ordinal number 1 in that segment.

3. The third section, which also starts with a +, shows another place where this code set is used: transaction set 850, the segment with ordinal number 38, the element with ordinal number 1 in that segment.

4. The fourth section, enclosed in square brackets, starts with (2) so it is the second code set ordinal for element 128. It lists the three codes from the dictionary and the ZY code, which is not in the dictionary. Since this is a code set, it is enclosed in square brackets.

5. The fifth section shows where the second code set is used: transaction set 850, segment with ordinal number 57, the element with ordinal number 1 in that segment.

Code Set Entirely from Dictionary Codes

If the subset was chosen entirely from dictionary codes, the element's entry in the .CODES section will resemble this example:

```
52=1:9,10:17%[1,2,3,6]+850/115///5
                ↑
        no ordinal number for code set
```

The ordinal number for the code set is omitted if all codes are in the dictionary. This record says the dictionary codes are 1-17, and the codes allowed in the implementation guidelines are 1,2,3, and 6 (which are all in the dictionary). If another code set followed this one, and it contained new text (such as a new local code value with a description), then it would be ordinal 1.

Code Set Contains No Codes from Dictionary

If dictionary codes exist, but the subset does not use any of them, the element's entry in the .CODES section will resemble this example:

```
52=1:9,10:17%[(1){90,97,98,99}]+850/115///5
                ↑
            no dictionary
            codes here
```

The ordinal number of the code set will be followed immediately by the curly bracket and the list of local codes, with no dictionary codes between.

Code Set Contains all Dictionary Codes

It is possible to define a code set consisting of every dictionary code. The resulting entry in the .CODES section will resemble this example:

```
52=1:9,10:17%[*]+850/115///5
                ↑
```

The entire code set area inside the [ ] is a single * , which indicates that the code set consists of all dictionary codes with nothing added, nothing removed.

Code Set for Element that has No Dictionary Codes

If no dictionary codes exist, but local code values are added, the element's entry in the .CODES section will resemble this example:

```
1306=%[(1)*{A,B,Z}]+850/115///15
        ↑      ↑
        1      2
```

1. No dictionary codes are listed in the area before the %.

2. * means all dictionary code values are in the code set. This is a special case, since the dictionary has no codes for this element.

Code Set Contains All Codes Except Specified Ones

A minus sign preceding one or more codes within the square brackets indicates that all dictionary codes are used except the ones shown.

```
399=2:9,10,11%[-4,5]+857/136///15
```

In this example, element 399 has dictionary codes 2-11. The code set appears as [-4,5], meaning it contains all dictionary codes except 4 and 5. It is attached in the 857 transaction set at the location shown.

Referring Back to Previous Code Sets

A pound sign immediately after an opening square bracket refers back to a previous code set for the same element. Example:

```
399=2:9,10,11%[-4,5]+857/136///15[#1,5]+204/13///4+204/34///4+856/189///15
```

The first code set is all dictionary codes except 4 and 5. This means it includes codes 2, 3, and 6-11. The second code set appears as [#1,5], meaning it is the same as code set number 1, but with the addition of **5**. The code set would therefore consist of 2, 3, 5-11. It is used in 2 locations in transaction set 204 and 1 location in transaction set 856.

The refer-back could subtract a code from the original code set. If the second code set had been [#1,-6], this would mean it is the same as the first code set, but with code 6 removed. The result would be 2, 3, and 7-11.

All Codes are Unused

It is possible to have an element with codes values, but all are marked unused and no local codes have been provided. These empty code sets appear as an empty set of square brackets.

```
147=1:9,A:Z%[]+859/2///1
```

In this example, element 147 has dictionary codes 1-9 and A-Z, but none of these are used at the location shown in transaction set 859. This is the same as having no code values. Any value that matches the type and length can be used.

Codes with Partitions

A vertical bar separates partitions of code values. For example (this actually appears on one line in the SEF file):

```
103=AMM,AMP,BAG,BAL,SLP,SRW,WRP|01,04,07,10,48:55%[AMM,AMP|01,07,50,53]+856
/109///1+856/250///1[#1|SLP]+180/11///1[#2|MIX,SRW|54]+856/15///1[BAG,BAL|1
3,16]+850/25///1
```

The code sets are underlined here so that you can find them, but they are not underlined in the SEF file.

Dictionary codes for element 103 are: `AMM,AMP,BAG,BAL,SLP,SRW,WRP` (part 1). The vertical bar after WRP indicates the end of part 1. Part 2 codes are 01, 04, 07, 10, and 48-55. Codes set for element 103 include:

Code Set	Part 1 Codes	Part 2 Codes
first code set: [AMM,AMP|01,07,50,53]	AMM, AMP	01, 07, 50, 53
second code set: [#1|SLP] (1st code set plus SLP for part 1, no changes to part 2)	AMM, AMP, SLP	01, 07, 50, 53
third code set: [#2|MIX,SRW|54] (2nd code set plus MIX, SRW for part 1, plus 54 for part 2)	AMM, AMP, MIX, SLP, SRW	01, 07, 50, 53, 54
fourth code set: [BAG,BAL|13,16]	BAG, BAL	13, 16

Codes that Contain Hyphens

Some TRADACOMS codes contain hyphens, which should not be confused with minus signs. This example combines some hyphenated and some non-hyphenated code values (it is on one line in the SEF file):

```
TTYP=CANCEL-ORDER,COMPLETE-CUS,COMPLETE-PRI,DA:DE,WHRECEIPT,XFM-
NOTE%[CANCEL-ORDER,COMPLETE-CUS,DC,DD,DE,XFM-NOTE]+ACKHDR/2///2
```

TRADACOMS codes can be up to 12 characters and include hyphens. If the hyphen or minus is within square brackets and occurs right after an opening square bracket or minus sign, it indicates "all but these codes" as explained

in Code Set Contains All Codes Except Specified Ones. Otherwise, a hyphen or minus within the square brackets indicates a hyphenated code value. Hyphens or minuses within the dictionary codes list always indicates a hyphenated code value.

Code Sets in Composites

Segment/Composite/Subelement	Codes	Codes in Dictionary?
MEA at 049/C001 at MEA-04/355 at C001-01	01	(in dictionary)
	99	(not in the dictionary)
MEA at 049/C001 at MEA-04/355 at C001-04	(same code set as above)	

The last part of element 355's entry in the .CODES section would look like this:

```
[(1)01{99}]+850/52///4-1+850/52///4-4
```

The parts that show the composites and subelements are in bold. In place of the element, we have the composite's ordinal number in the segment, a dash, and the subelement's ordinal number in the composite.

Code Sets Attached in Dictionary

If code sets are attached in the dictionary, the entire section after the plus sign will be different, as follows:

Code set attached in transaction set or message (no composite involved). Example**:** One code set on transaction set 850, the segment with ordinal 4, the element with ordinal 1 (assume this is the CUR01, element 98). The code set includes codes 01, 02, 05, 06, and 09:

```
%[01,02,05,06,09]+850/4///1
```

Code set attached in segment dictionary (no composite involved). Now assume that this same element (98) has another code set on the CUR segment, this time on the CUR01 in the segment dictionary. Element 98 still has ordinal 1:

```
%[01,02,05,06,09]+850/4///1[01,02,03,04,05,06,07,08,09,10,11,12,16]+//CUR//1
```

Code set attached in transaction set or message (composite involved): One code set is attached to transaction set 850, the MEA segment (which has ordinal 22), the C001 composite (which has ordinal 4), element 355 (which has ordinal 1). The code set includes only the codes 01 and 1N:

```
%[01,1N]+850/22///4-1
```

Code set attached in segment dictionary (composite involved): A second code set has been added in the segment dictionary on the MEA, the C001 composite (which as ordinal 4), subelement 355 (ordinal 1). The code set includes codes 02-05:

```
%[01,1N]+850/22///4-1[02,03,04,05]+//MEA//4-1
```

Code set attached in composite dictionary: A third code set has been added to the C001 in the composite dictionary, on the 355 (which has ordinal 1):

`%[01,1N]+850/22///4-1[02,03,04,05]+//MEA//4-1[3B,3C]+///C001/1`

Empty Code Sets

If a dictionary element has code values, but they are all unused at a particular location in the transaction set or message, this is an empty code set. This means that any value that matches the type, min, and max will be acceptable. The part of the line dealing with code sets will have empty square brackets: `%[]+155/2///1`

Unattached Code Sets

You may see code sets that have no addressing information after the square brackets.

`8=E,R%[*]`
`66=1:9,10:50,53:59,61:63,71:78,81:82,90:99,A,A1%[1,9]`

For all practical purposes, this is the same as having no code set. All dictionary code values are valid in both examples above ... even though a set consisting of 1 and 9 has been defined in the second example. It is not being used anywhere.

Summary of Symbols used in .CODES

n:n	A range of codes in the dictionary: 05:15 means 5 through 15. The beginning and ending code must have the same length. Example: 05:15 is good because both numbers have 2 digits. A:Z is good because both have the same number of letters. 5:15 is not good because 5 has one digit, while 15 has two digits. A:ZZ is not good because A has one character while ZZ has two.
%	The end of the list of dictionary codes.
[ ]	Marks the beginning and end of a code set in the implementation guidelines.
( )	Surrounds the code set ordinal number.
{ }	Surrounds codes that are not in the dictionary.
+	Appears after the code set and immediately before the location of the transaction set or message/segment/element where it is used.
/	Separates the parts of the location where the code set is used. See Code Sets Attached in Dictionary on page 36 for details.
-	(in location area) Appears between the ordinal number of the composite within the segment, and the ordinal number of the subelement within the composite. Example: 4-1 means the composite with ordinal number 4 within the segment, and the subelement with ordinal number 1 within that composite.
-	(inside square bracket area) Code set includes all codes in dictionary or in referred-to code set except the ones listed after the minus sign. See Code Set Contains All Codes Except Specified Ones on page 34.
-	(inside square bracket area) TRADACOMS codes can actually contain hyphens. See Codes that Contain Hyphens on page 35.
*	Indicates that a code set uses all dictionary code value.
#*n*	Refers back to code set *n*. See Referring Back to Previous Code Sets on page 35.
|	A vertical bar separates the partitions in partitioned codes. See Codes with Partitions on page 35.

The .VALREFS and .VALLISTS Sections

These two sections define the implementation's application value lists and show where they are attached. Application value lists are not officially part of the standard, although they can be attached to elements in the dictionary as well as in the implementation guidelines. They are not as restrictive as code values, since they can be any length that will fit in the element, can include capital and lower case letters, and can include special characters and blanks.

The .VALREFS section shows where application values lists are attached. It is a list of ordinal paths and/or element IDs, plus the application values that can be used with them. Elements without application values are not listed.

The .VALLISTS section shows what each application values list contains.

```
.VALREFS                              ←1
//ISA//6=DUNS                         ←2
///310/=LOC-CODE                      ←3
850/185///2=REFID                     ←4


.VALLISTS                             ←5
DUNS="Kaver DUNS number"              ←6
'78-250-12250001
LOC-CODE="Kaver Corp. regions"   ⎤
'Eastern division                │
'Great Lakes N.                  │
'Great Lakes S.                  ⎬  7
'Midwest 1                       │
'Midwest 2                       │
'Northeast N                     ⎦
REFID=                                ←8
&^REF[A-Z][A-Z][0-9][0-9][0-9]$
```

Explanation of numbers in example:

1	Start of .VALREFS section.
2	Application value list DUNS is attached to the element with ordinal 6 in the ISA segment in the dictionary.
3	A dictionary element with an attached application value lists (note the lack of transaction set and segment ordinal). Element 310 has application value list "LOC-CODE" attached.
4	The REFID application value list is attached in transaction set 850. It is attached to the 2nd ordinal in the segment with ordinal 185.
5	Start of the .VALLISTS section, which shows the names and contents of each application value list.
6	This is application value list "DUNS" It's description is "Kaver DUNS number." Its one value is preceded by a single quote, but the quote is not part of the value.
7	This is application value list "LOC-CODE," which has 6 values.
8	This is the start of application value list "REFID." It has no description, and so the area after the equals sign is blank. Since the value that it contains starts with an ampersand (&), this line is a regular expression rather than a literal value. The expression &^REF[A-Z][A-Z][0-9][0-9][0-9]$ means that the value starts with **REF**, followed by two capital letters and then three digits. There can be no characters before the REF, since the expression starts with a circumflex (^). There can be no characters after the three digits, since the expression ends with a dollar sign.

Application values can be attached in the transaction set or message, or in any of the dictionaries. They would look like this in the .VALREFS section:

ORDERS/185///2=REFNUM	**Attached in EDIFACT message:** Value list **REFNUM** is attached to the element with ordinal 2 in the segment with ordinal 185 in message ORDERS.
850/185///2=REFNUM	**Attached in X12 transaction set:** Value list **REFNUM** is attached to the element with ordinal 2 in the segment with ordinal 185 in transaction set 850.
//REF//2=REFNUM	**Attached in segment dictionary:** Value list **REFNUM** is attached to the element with ordinal 2 (usually REF02) in the segment dictionary.
///C001/2=EXPS4	**Attached in composite dictionary:** Value list **EXPS4** is attached to the subelement with ordinal 2 (usually the second subelement) in composite C001 in the composite dictionary.
//MEA//4-3=MULT2	**Attached in element dictionary:** Value list **MULT2** is attached to the MEA's item with ordinal 4 (which is composite C001), on the subelement with ordinal 3 within that composite in the segment dictionary.

Regular Expressions

Application values can support regular expressions: This affects the .VALREFS and .VALLISTS sections.

X12 example

```
.VALREFS
850/76///7=UPC

.VALLISTS
UPC="upc codes"
&13-11111-222[0-9][0-9]
'12-11111-12345
'12-34567-89012
```

This .VALREFS section shows that an application value list called **UPC** is attached to the 850 transaction set, 76th segment ordinal, 7th element ordinal.

This .VALLISTS section shows the description and contents of the UPC application value list. Its description is **upc codes** and it contains a regular expression and two literal values.

The & at the beginning of the line indicates that a regular expression follows. The & is not part of the regular expression. In this case, the regular expression is:

```
13-11111-222[0-9][0-9]
```

This means the value is 13-11111-222 followed by any two digits.

A single quotation mark at the beginning of a line means a literal value follows. The quotation mark is not part of the value.

EDIFACT example

```
.VALREFS
//UNB//2-1=SENDER
ORDERS/16///1-2=REFID
.VALLISTS
REFID="Reference ID"
&^[A-Z][A-Z][A-Z][0-9][0-9][0-9]$
SENDER="Sender Identification"
'78-250-12250001
```

The .VALREFS section shows that an application value list called SENDER is attached in the dictionary to the UNB segment, 2nd element ordinal, 1st composite subelement ordinal.

A second list called REFID is attached in the ORDERS message, 16th segment ordinal, 1st element ordinal, 2nd composite subelement ordinal.

The .VALLISTS section shows that the REFID list has a description of **Reference ID** and contains a regular expression `^[A-Z][A-Z][A-Z][0-9][0-9][0-9]$`, which means the first three characters are each A-Z, and the last three are digits. The `^` means nothing can precede these values, and the $ means nothing can follow. It must have exactly 3 letters followed by exactly 3 digits.

The & at the beginning of the line means this is a regular expression, not a literal value.

The SENDER list has a description of **Sender Identification** and consists of one literal value. The single quote at the beginning of the line means a literal value follows.

The .OBJVARS Section for Variable Names

The .OBJVARS section lets you give variable names to objects in your guideline. You can then use the variable names in the equivalent of "if" statements in the SEMREFS section. The format is:

> *location=variablename*

X12 example

```
.OBJVARS
850/2///8=beg08invtype
```

In this X12 purchase order (850), the 2nd segment ordinal, the 8th element ordinal, is being assigned the variable name `beg08invtype`. Although this example includes the segment and element (beg08) in the variable name, this is not necessary. The name could have been `invtype`.

Variable `beg08invtype` can then be used in business rules in the .SEMREFS section to refer to the object at location ordinal `850/2///8`.

EDIFACT example

```
.OBJVARS
ORDERS/4///1-3=pai0103paymethod
```

In this ORDERS message, the 4th segment ordinal, the 1st element ordinal, the 3rd composite subelement, is being assigned the variable name `pai0103paymethod`.

The .SEMREFS Section for Semantic Rules

The .SEMREFS section sets up specialized semantic rules that reflect very specific needs. Simple example of a semantic rule in an X12-4011 850 (paraphrased):

If the BEG08 contains the code "IBM" then the REF segment at 0500 is used

SEF captures such a rule as follows:

1. In the .OBJVARS section, the guideline creator has assigned the variable name **beg08invtype** to the BEG08.

2. Assuming the REF segment is at ordinal 850/4///, the REF segment in this if-then condition can be captured in SEF as follows:

    parameters go here – if any.

    ```
    .SEMREFS

    850/4///=[beg08invtype'EQ'IBM:USAGE:0:]
    ```
    location   *condition*   *type*   *setting*

The format follows. The underlined parts are mandatory. There are no spaces, and a colon separates each section.

location= [*conditon*:*type*:*setting*:*parameter*]

     "If"   "Then"

Where:

location   (mandatory) The ordinal path of the object that varies according to the "if" test. Ordinals are explained in the .SETS, .SEGS., and .COMS sections of the SEF manual.

             Important: *location* must be marked as dependent if the *type* is USAGE. This allows the semantic note to control its usage. If it is a segment, place & (an ampersand) in front of it in the .SETS section: `[&REF@4,,>1]` .

             If *location* is a composite or element, an & in the segment's mask in the .SEGS section indicates that its usage is dependent. In this example, the second and third items (element 4453 and composite C107) in the FTX segment are both dependent:
`FTX=[4451,M][4453][C107][C108][3453],.&&*1..`

             If *location* is a composite subelement, an & in the composite's mask in the .COMS section shows that its usage is dependent. In this example, the second subelement (the 1131) in the C107 composite is dependent: `C107=[4441,M][1131][3055],.&`.

             The location consists of up to five parts, each separated by a slash:

             (1) Transaction Set ID (3 alphanumeric characters for X12, 6 for EDIFACT)

             (2) Ordinal number of segment reference within transaction set

             (3) Segment tag, applicable to dictionary segment only

             (4) Element ID, applicable to dictionary element only

             (5) Element position (ordinal) within the segment (or element- sub-element positions).

             Examples:

850/3///1	Transaction set 850, 3rd segment, 1st element.
850/59///4-1	Transaction set 850, 59th segment, fourth item (a composite, like MEA04-C001), first subelement.
///355/	Element dictionary: element 355.
//ISA//5	Segment dictionary: 5th element of ISA.

//UNG//2-1	Segment dictionary segment UNG, 2nd element (S006), 1st subelement (0040)
///C001/1	Composite dictionary composite C001, 1st subelement.

The four sections to the right of the equals sign and within square brackets include:

condition (optional) The test that must be passed in order for the "then" action to be taken. If omitted, the rest of the expression always occurs. The colon separator between *condition* and *type* must be included, even if *condition* is omitted. The c*ondition* section is explained below.

type (mandatory) The fate of the object at *location*, if *condition* is met. The *type* section is explained in The Type, Setting, and Parameter Areas on page 43. A colon always appears at the end of the *type* section.

setting (mandatory) A setting that depends on the *type*. The *setting* section is explained in The Type, Setting, and Parameter Areas on page 43. A colon always appears at the end of the *settings* section, even if *parameter* is omitted.

parameter A parameter that is needed by some *type*s. The *parameter* section is explained in The Type, Setting, and Parameter Areas on page 43.

colon separators The three colons separating the *condition*, *type*, *setting*, and *parameter* sections are mandatory, even if an optional section is omitted.

The *condition* ("if") area

variable The variable name assigned to a location that is being tested. Variable names are assigned in the .OBJVARS section.

operator One of the following, surrounded by single quotes.

EQ	the variable's data **equals**
NE	the variable's data does **not equal**
GT	the variable's data is **greater than**
GE	the variable's data is **greater than or equal to**
LT	the variable's data is **less than**
LE	the variable's data is **less than or equal to**
EXISTS	the variable's data **exists** in the EDI data file
NEXIST	the variable's data **does not exist** in the EDI data file

value If the condition is EQ, NE, GT, GE, LT, or LE, then supply a data value immediately after the closing single quote. This is an actual value of the data in the element identified by *variable*. If the condition is EXISTS or NEXIST, there will be no value, and a separator follows the closing single quote.

separator One of the following:

&	an ampersand means "AND"
|	a vertical bar means "OR"
:	a colon ends the *condition* section

Example conditions:

(X12) If beg08invtype data equals IBM, or does not exist, usage of the object at ordinal 850/4/// is "must use"

condition

```
850/4///=[beg08invtype'EQ'IBM|beg08invtype'NEXIST':USAGE:2:]
```

vertical bar
means "or"

first colon separates
condition and type

(EDIFACT) If the pai0103paymethod data contains 20 or 26, usage of the object at ordinal ORDERS/12/// is "must use."

`ORDERS/12///=[`**`pai0103paymethod'EQ'20`**`|`**`pai0103paymethod'EQ'26`**`:USAGE:2:]`

USAGEs are explained below.

No *condition* **in semantic rule:** If the condition is omitted entirely, the semantic rule will start with a colon. This means that the "then" is always in effect. Example:

Always run an external routine on the element at ordinal 850/29///2:

`850/29///2=[:COMEXIT:FSVBExit.CheckDigit.ValidateCheckDigit:88,UP(UPC12)]`

Always set usage for the object at ordinal 850/4/// to not used:

`850/201///=[:USAGE:1:]`

Normally, you would not use USAGE without a condition, since you could more easily accomplish the same thing in the .SETS section by placing a period in front of the segment: `[.REF,,2]`

The Type, Setting, and Parameter Areas

The *type* , *setting*, and *parameter* areas of the semantic rule execute if the *condition* is satisfied. Mandatory parts are underlined:

<u>*location*= **[***condition***:** *type***:** *setting***:** *parameter***]**</u>

type and *setting* must be included. In some cases, it is followed by a *parameter*. These are described together since some types must have parameters, and others do not.

type	Corresponding setting	parameter
USAGE	A digit, 0-4, representing a usage (explained in **USAGE** below)	(none)
LOCALCODE	A code set ordinal (explained in LOCALCODE on page 44)	(none)
APPVALUE	The name of an application values list (explained in APPVALUE on page 44)	(none)
exit routine	The name of a COM exit (explained in Exit Routines on page 45)	(some COM exits have parameters)

USAGE

USAGE sets the object's usage if *condition* is met. In the .SETS, .SEGS, or .COMS sections, the object must be set to dependent (&). Usage *setting* can be:

0   for "used"
1   for "not used"
2   for "must use"
3   for "recommended"
4   for "not recommended"

Examples

`850/193///=[beg08invtype'EQ'IBM:`**`USAGE:2:`**`]`

The usage for the segment or loop at ordinal 850/193/// will be "must use" if the element labeled with variable beg08invtype contains IBM.

```
850/4///=[beg08invtype'EQ'IEL|beg08invtype'EQ'INR:USAGE:1:]
```

The usage for the segment or loop at ordinal 850/4/// will be "not used" if the element labeled with variable beg08invtype contains IEL or INR.

LOCALCODE

LOCALCODE specifies a code set to use if *condition* is met. The format is:

location=[*condition*:**LOCALCODE:***n*:]

where:

LOCALCODE   is a literal.

n   is the code set ordinal in the .CODES section. Note the trailing colon.

Example

```
820/6///1=[TotalAmtBPR02'GT'1000:LOCALCODE:0:]
```

In this 820 transaction set, 820/6///1 points to element 128 at REF06. If the element labeled with TotalAmtBPR02 contains data greater than 1000, then element 128's code set 0 (the first one) is to be used. To see the code set itself, look in the .CODES section under element 128, where you would see something like this:

128=*list of used dictionary codes*%[VE,VR]

　　　　　　　　　　　　　　↑____ *code set position is 0, since it is the first one after the %*

This rule could then be summarized as: If the object identified in the .OBJVARS section by the variable name TotalAmtBPR02 contains data greater than 1000, then use the code list VE,VR. for the object with ordinal 820/6///1.

Notice that code set positions start with 0, not 1.

Code sets that are used only in semantic rules, but not directly attached anywhere, will have no addressing information appended to them in the .CODES section:

Attached in ORDERS message:	128=*list of used dictionary codes*%[VE,VR]+ORDERS/3///1-3
Attached in 850 transaction set:	128=*list of used dictionary codes*%[VE,VR]+850/3///1-3
Used only in semantic rule; not directly attached anywhere:	128=*list of used dictionary codes*%[VE,VR]

APPVALUE

APPVALUE specifies an application values list to use if *condition* is met. The format is:

location=[*condition*:**APPVALUE:***applist*:]

where:

APPVALUE   is a literal.

applist   is the name of the application value list. These are set up in the .VALLISTS section. Note the trailing colon.

Example (EDIFACT)

```
ORDERS/228///2-2=[NAD0201payer'EQ'PR:APPVALUE:PURAGT:]
```

This means: If the data in the element specified by variable NAD0201payer is **PR**, use one of the values in the application value list PURAGT for the subelement at location ORDERS/228///2-2.

The contents of list PURAGT are set up in the .VALLISTS section as in the following example:

```
.VALLISTS
PURAGT="Purchasing Agent"
'Ed Wilson
'Pat Paulie
'R. Jamieson
```

There is no entry for PURAGT in the .VALREFS section for ORDERS/228///2-2, since it only applies if NAD0201payer equals PR.

Exit Routines

SEF provides for exit routines that might be used by translators or EDI data analyzers. They could validate element values, enforce dependency rules, etc. These are specific to a particular translator or analyzer application that reads the SEF, and should be ignored by others.

The SEF specifies an external routine to run at a particular location. You can optionally specify a condition that must be met before it runs. The format is:

location= [*condition* : *exit name* : *program* **.** *function* : *parameters*]

Where:

exit name	An application-specific name for the type of exit (Microsoft's component object model). Foresight uses COMEXIT for a Microsoft common object module exit.
program	Name of a COM-enabled server program that supports OLE automation, such as a Java class, Visual Basic ActiveX DLL, or C++ ActiveX DLL. This name may have multiple parts, each separated by a period. The program in the example below is FSVBExit.CheckDigit. The program should be listed in the windows registry of the PC where this guideline will be used.
function	Name of the function in the program, preceded with a period and followed by a colon.
parameters	(if the function uses parameters) One or more parameters used by the function.

Examples:

no condition

```
ORDERS/12///2-1=[:COMEXIT:FSVBExit.CheckDigit.ValidateCheckDigit:]
```

exit   *program*   *function*   *no parameters*

The example above specifies that, when the subelement at ORDERS/12///2-1 is encountered, run the ValidateCheckDigit function in server FSVBExit.CheckDigit. It should run under all conditions, since there is no condition before the first colon. COMEXIT is a local term used by the application that is to read the SEF.

```
850/1///=[:COMEXIT:CheckCTT.SetCheckCTTCount:]
```

When the element at 850/1/// is encountered, always run the SetCheckCTTCount function in server CheckCTT.

```
850/29///2=[:COMEXIT:FSVBExit.CheckDigit.ValidateCheckDigit:88,UP(UPC12)]
```

At the specified location, always run FSVBExit.CheckDigit.ValidateCheckDigit, and pass it the following parameters: 88,UP(UPC12).

Multiple Semantic Rules on One Object

The same ordinal can have multiple semantic rules, each enclosed in square brackets. Example:

If beg08invtype data equals IBM and beg09contracttyp data exists, the ordinal at 850/4/// must be used.

If beg08invtype data doesn't exist, the ordinal at 850/4/// is not used.

```
850/4///=[beg08invtype'EQ'IBM&beg09contracttyp'EXIST':USAGE:2:][beg08inv
type'NEXIST':USAGE:1:]
```

There are no spaces in this example.

The .TEXT Sections

The .TEXT sections store information about changes in a standard's text. This includes notes, comments, names, purposes, descriptions, titles, semantic notes, explanations, definitions, etc. Any text not provided in the SEF file for an implementation guideline is assumed to be that in the underlying X12 or EDIFACT standard.

This example, from a .TEXT,SETS section, shows the three fields in each record within TEXT sections:

```
850,0,Purchase Order for Kaver Corpn/l
850,3,Sender and receiver codes should be passed. n/l
850~1,3,Transaction Set Header for Kaver Corp POn/l
850~1,4,ST customized for Kaver Corp's PO. n/l
   ↑   ↑                          ↑
   1   2                          3
```

1. The first field is the location of the changed item. A comma marks the end of this field. In the four example lines above, this field is either **850** or **850~1.**

2. The second field tells what was changed at that location. A comma marks the end of this field.

3. The third field shows the actual text after the change. A **n/l** ends this field, which may be very long. Some notes, comments, etc., can appear as wrapped on this printed page and on your computer screen, but take care that they are not accidentally broken by a word processor or editor.

Since these fields can vary, depending on what was changed, they are described in the .TEXT sections below.

The .TEXT,SETS Section

This section contains text information local to each transaction set or message. This can be:

*   Text describing the transaction or message itself.

*   Or, text that will be used in a particular hierarchy in the set: a note on a particular element when used in a particular segment when used in a particular set, for example.

This section does not contain any text from the dictionary.

Records in this section will always start with the transaction set ID (ASC X12) or message name (EDIFACT).

Short introductory example

```
850~1,3,Transaction Set Header for Kaver Corp PO
```

This says:

Where (the first field):	In transaction set 850's segment with ordinal number 1 (usually the ST)
	See "The First Field: Where" below.
What (the second field):	The name for a segment reference. See "The Second Field: What" below.
Text (the third field):	"Transaction Set Header for Kaver Corp PO."

The First Field: Where

In .TEXT,SETS, the first field contains the transaction set or message, perhaps followed by a segment, which may be followed by a composite or element, etc. Examples:

850	The location of the change is the 850 transaction set.
850~1	The location of the change is: • 850 transaction set. • Within the 850, the segment with ordinal number 1. This is probably the 850's ST. See Ordinal Numbers in the .SETS Section on page 13.
850~1~2	The location of the change is: • 850 transaction set. • Within the 850, the segment with ordinal number 1. • Within that segment, whatever has ordinal number 2: either an element or a composite.
850~20~4~4	The location of the change is: • 850 transaction set. • The segment with ordinal number 20 within that set. • Whatever has ordinal number 4 within that segment (a composite). • Whatever has ordinal number 4 within that composite (a subelement).
850~2~2~EO~1~0	The location of the change is: • 850 transaction set. • The segment with ordinal number 2 within that set. • Whatever has ordinal number 2 within that segment. In this case, we can tell it is an element here, because the next item is a code value. • Code value EO. • Part 1 of code value EO. This will almost always be 1 except for rare instances where there is a two-part code value. See **Two-part codes** on page 65. • The code set ordinal number. 0 means there is no code set ordinal.
850~20~4~4~EO~1~0	The location of the change is: • 850 transaction set. • The segment with ordinal number 20 within that set. • Whatever has ordinal number 4 within that segment. In this case, we can tell it is a composite here, because the next item is not a code value. • The subelement with ordinal number 4 within that composite. • Code value EO. • Part 1 of code value EO. This will almost always be 1 except for rare instances where there is a two-part code value. See **Two-part codes** on page 65. • The code set ordinal number. 0 means there is no code set ordinal.

The Second Field: What

In .TEXT,SETS, the second field contains one of the following codes, which have different meanings depending on the path shown in field 1:

Path in Field 1	Number in Field 2*	Item That Gets Text in Field 3
Transaction Set: 850	0	Transaction Set or message title.
	1	Transaction Set functional group (X12).
	2	Transaction Set or message purpose.
	3	Level 1 note on transaction set or message.
	4	Level 2 note on transaction set or message.
	5	Level 3 note on transaction set or message. See * below for other levels of notes.
Transaction Set~segment ordinal number: 850~1	0	Segment reference notes that are part of the transaction set in X12.
	1	Segment reference notes documented with the segment (like in VICS/UCS).
	2	Segment reference comment documented with the transaction set.
	3	Segment name.
	4	Level 1 note on segment.
	5	Level 2 note on segment.
	6	Segment purpose.
	7	Level 3 note on segment. See * below for other levels of notes.
Transaction Set~segment ordinal number~element or composite ordinal number: 850~1~4	0	Level 1 note on element or composite.
	1	Level 2 note on element or composite.
	2	Name of element or composite.
	4	Level 3 note on element or composite. See * below for other levels of notes.

Transaction Set~segment ordinal number~composite ordinal number~subelement ordinal number: 850~20~4~1	0	Level 1 note on subelement.
	1	Level 2 note on subelement.
	2	Name of subelement.
	4	Level 3 note on subelement. See * below for other levels of notes.
Transaction Set~segment ordinal number~element or composite number [~subelement ordinal number if composite]~code value~code value part~code set: 850~1~4~AH~1~0 or 850~1~4~8~AH~1~0	0	Level 1 note for code value when in hierarchy shown.
	1	Level 2 note for code value when in hierarchy shown.
	4	Level 3 note for code value when in hierarchy shown. See * below for other levels of notes.

* A field 2 number of 14 or more is a level note greater than level 3. Please see **Level Notes** on page .

The Third Field: the Text

The third field will be the text to be used for the item in field 2 when in the location in field 1. The following is an example .TEXT,SETS section with the third field in each record shown in bold.

```
.TEXT,SETS
850,2,This Draft Standard for Trial Use contains the format and establishes
the data contents of the Purchase Order Transaction Set (850) for use
within the context of an Electronic Data Interchange (EDI) environment. The
transaction set can be used to provide for customary and established
business and industry practice relative to the placement of purchase orders
for goods and services. This transaction set should not be used to convey
purchase order changes or purchase order acknowledgment information. This
is for use by Kaver Corp's Eastern Division after 3/31/95.n/l
850~2,3,Beginning Segment for Purchase Order for Kaver Corp.n/l
850~3,3,Note/Special Instructionn/l
850~3,5,Sys Sup: If the PO is marked as expedited, this field will
contain:\r\n\r\nE1- urgent, process immediately\r\nE2- priority, process in
tonight's overnight queue\r\nE3- prompt, process with next scheduled
run\r\nE4- routinen/l
```

Record length: Records can be long. In the example above, we have marked the new-lines with **n/l**.

Symbols that you might encounter include:

\r	carriage return
\n	line feed
\r\n	carriage return and line feed (equivalent to a new line)
\r\n\r\n	results in one blank line in the text

Example .TEXT,SETS Section

Instead of realistic text, the third field has been changed to make it clearer to you where the text goes:

```
.TEXT,SETS
850,0,New title for Purchase Order
850,1,PZ                                    ← changed functional group
850,2,New purpose on 850
850,3,Level 1 note on 850
850,4,Level 2 note on 850
850~2,3,New name for BEG which is ordinal 2 in PO
850~2,4,Level 1 note for BEG which is ordinal 2 in PO
850~2,5,Level 2 note for BEG which is ordinal 2 in PO
850~2~7,0,Level 1 note for PO, BEG-07. BEG is ordinal 2 in PO, BEG-07 is
the element that is ordinal 7 in BEG
850~2~7,1,Level 2 note for PO, BEG-07
850~2~7~AD~1~0,0,Level 1 note on PO, BEG-07, code AD
850~2~7~AD~1~0,1,Level 2 note on PO, BEG-07, code AD
850~2~7~AD~1~0,52,Level 42 note on PO, BEG-07, code AD
```

The .TEXT,SEGS Section

This section contains segment dictionary text that differs from the standard. The principals are the same as those described in "The .TEXT Sections**"** above. The records start with a segment.

Short introductory example

```
ST,2,This segment starts a transaction set for Kaver Corp.
```

This says:

Where (the first field):	ST in the dictionary.
What (the second field):	Level 1 note on dictionary segment ST (see The Second Field: What below)
Text (the third field):	"This segment starts a transaction set for Kaver Corp."

The First Field: Where

In .TEXT,SEGS, the first field will be the dictionary segment, which may be followed by a composite or element, etc. Examples:

ST	The location of the change is the dictionary segment ST.
ST~2	The location of the change is: • Dictionary segment ST. • Within the dictionary segment ST, the element with ordinal position 2 (see Ordinal Numbers in the .SEGS Section on page 20).
ST~0	The location of the change is: • Dictionary segment ST. • Within the dictionary segment ST, the element with ordinal position 0. Ordinal position 0 refers to all elements in the segment. This could go with a semantic note that is general and applies to all elements in the ST.
ST~4~2~EO~1~0	The location of the change is: • Dictionary segment ST. • Within the dictionary segment ST, the composite at ordinal position 4. • Within the composite, the subelement at ordinal position 2. • Code EO, which is a one-part code having 0 code sets.

The Second Field: What

In .TEXT,SEGS, the second field contains one of the following codes, which have different meanings depending on the path shown in field 1:

Path in Field 1	Number in Field 2 *	Item That Gets Text in Field 3
Dictionary segment: ST	0	Segment dictionary's name for segment.
	1	Segment dictionary's purpose for segment.
	2	Segment dictionary's level 1 note on segment.
	3	Segment dictionary's level 2 note on segment.
	4	Segment dictionary's level 3 note on segment.
		See **Level Notes** on page 60 for other levels of notes.
Dictionary segment ~ element or composite ordinal number: ST~1	0	(Reserved)
	1	Segment dictionary's semantic note for the element or composite.
	2	Segment dictionary's comment for the element or composite.
	3	Segment dictionary's level 1 note for the element or composite.

	4	Segment dictionary's level 2 note for the element or composite.
	5	Segment dictionary's name for the element or composite.
	7	Segment dictionary's level 3 note for the element or composite.
		See **Level Notes** on page 60 for other levels of notes.
Dict. Seg. ~composite~subelem:	3	Segment dictionary's level 1 note on subelement.
MEA~4~2	4	Segment dictionary's level 2 note on subelement.
	5	Segment dictionary's name of subelement.
	7	Segment dictionary's Level 3 note on subelement. See * below for other levels of notes.
Dict. Seg.~element~code~part~ codeset-ordinal:	1	Segment dictionary's level 1 note on code.
MEA~2~BB~1~0 or:	2	Segment dictionary's level 2 note on code.
MEA~4~2~BB~1~0 (with composite)		
	4	Segment dictionary's level 3 note on code.
		See **Level Notes** on page 60 for other levels of notes.

* A field 2 number of 14 or more is a level note greater than level 3. Please see **Level Notes** on page 60.

The Third Field: the Text

The third field will be the text to be used for the item in field 2 when in the location in field 1. The following is an example .TEXT,SEGS section with the third field in each record shown in bold.

```
.TEXT,SEGS
```

BEG,3,**Systems Support: This runs in tandem with old Accounting System 1A until June 30, 1995. This segment will have to be changed to meet the specs of the new Accounting System 95A. See R. Kiefer for details.***n/l*

PID~4,1,**PID04 should be used for industry-specific product description codes. New codes will start with two alpha characters with the remainder integers.***n/l*

Record length: As in the .TEXT,SETS section, some notes, comments, purposes, etc., can be very long. Therefore, some records in this section can be very long, although they appear as wrapped here. For the example above, we have marked the new-lines with ***n/l*** .

Symbols that you might encounter include:

\r	carriage return
\n	line feed
\r\n	carriage return and line feed (equal to a new line)
\r\n\r\n	results in one blank line in the text

Example .TEXT,SEGS Section

Instead of realistic text, the third field has been changed to make it clearer to you where the text goes:

```
.TEXT,SEGS
ST,0,Transaction Set Header for Kaver Corp
ST,1,To indicate the start of our local transaction set and to assign a
control number
ST,3,level 2 note on dictionary segment ST
ST~0,1,Semantic note on dictionary segment ST, element position 00
ST~2,1,Semantic note on dictionary segment ST, element position number
02.\r\n
ST~2,2,Comment on dictionary segment ST, element position no. 02
```

The .TEXT,COMS Section

This section contains composite dictionary text that differs from the standard. The principals are the same as those described in The .TEXT,SETS Section on page 47. The records start with a composite.

Short introductory example:

```
C058,2,Element C058-01 should contain no punctuation.
```

This says:

Where (the first field):	Composite C058 in the dictionary.
What (the second field):	Level 1 note on dictionary composite C058 (see The Second Field: What below)
Text (the third field):	"Element C058-01 should contain no punctuation."

The First Field: Where

In .TEXT,COMS, the first field will be the dictionary composite, which may be followed by a subelement ordinal. Examples:

C001	The location of the change is the dictionary composite C001.
C001~2	The location of the change is: • Dictionary composite C001. • Within the dictionary composite C001, the subelement with ordinal position 2 (see Ordinal Numbers in the .COMS Section on page 29).
C001~2~ABC~1~0	The location of the change is: • Dictionary composite C001. • Subelement with ordinal 2. • Code ABC, which is a 1-part code that is not part of a code set.

The Second Field: What

In .TEXT,COMS, the second field contains one of the following codes, which have different meanings depending on the path shown in field 1:

Path in Field 1	Number in Field 2*	Item That Gets Text in Field 3
Composite (example: C001)	0	Dictionary composite name.
	1	Dictionary composite purpose.
	2	Level 1 note on dictionary composite .
	3	Level 2 note on dictionary composite.
	4	Level 3 note on dictionary composite. See * below for other levels of notes.
Composite~subelement ordinal number (example: C058~2)	0	(Reserved)
	1	Dictionary composite semantic note for the subelement in the ordinal position shown.
	2	Dictionary composite comment for the subelement in the ordinal position shown.
	3	Dictionary level 1 note on the subelement when used in that ordinal position in the composite shown.
	4	Dictionary level 2 note on the subelement when used in that ordinal position in the composite shown.
	7	Dictionary level 3 note on the subelement when used in that ordinal position in the composite shown. See * below for other levels of notes.
Composite~subelement ordinal number~code~part~codeset ordinal (example: MEA~4~7~ABC~1~0)	0	Level 1 note on code in composite dictionary.
	1	Level 2 note on code in composite dictionary.
	4	Level 3 note on code in composite dictionary. See * below for other levels of notes.

* A field 2 number of 14 or more is a level note greater than level 3. Please see **Level Notes** on page 60.

The Third Field: the Text

The third field will be the text to be used for the item in field 2 when in the location in field 1. The following is an example .TEXT,COMS section with the third field in each record shown in bold.

```
.TEXT,COMS
```

C080,0,**Party name for Kaver Corp.***n/l*

C058~0,2,**In general, the subelements in C058 should include the complete address including the country code. Do not include the company name.***n/l*

C058~1,1,**Element C058-01 should contain the 1st line of the street address with no punctuation***n/l*

Record length: Some records can be very long in this section. For the example above, we have marked the new-lines with *n/l* .

Symbols that you might encounter include:

\r	carriage return
\n	line feed
\r\n	carriage return and line feed (equal to a new line)
\r\n\r\n	results in one blank line in the text

Example .TEXT,COMS Section

Instead of realistic text, the third field has been changed to make it clearer to you where the text goes:

```
.TEXT,COMS
C058,2,This is a level 1 note
C058,3,This is a level 2 note
C080,0,This is a new name for the composite
C080,1,This is the new purpose for C080
C080,2,This is a level 1 note for C080
C080,3,This is a level 2 note for C080
C080~0,1,This is a semantic note on C080 on subelement 00 (general)\r\n
C080~2,2,This is a comment for element with ordinal number 2 in C080\r\n
C080~4,1,This is a semantic note for subelement with ordinal number 4 in
C080
```

The .TEXT,ELMS Section

This section contains element dictionary text that differs from the standard. The principles are the same as those described in "The .TEXT,SETS Section" on page 47. The records start with an element number.

<div style="border:1px solid black; padding:10px;">

<div align="center">**Short introductory example:**</div>

`3036,1,Name of sending party involved in transaction.`

This says:

Where (the first field):	Element 3036 in the dictionary.
What (the second field):	Level 1 note on dictionary element 3036 (see The Second Field: What below)
Text (the third field):	"Name of sending party involved in transaction."

</div>

The First Field: Where

In .TEXT,ELMS, the first field will be the dictionary element, which may be followed by a code. Examples:

3036	The location of the change is the dictionary element 3036.
3279~DO~1~0	The location of the change is: • Dictionary element 3279. • Code value DO. • Part 1 of code value DO. This will almost always be 1 except for rare instances where there is a two-part code value. See **Two-part codes** on page 65. • The code set ordinal number. 0 means there is no code set ordinal.

The Second Field: What

In .TEXT,ELMS, the second field contains one of the following codes, which have different meanings depending on the path shown in field 1:

Path in Field 1	Number in Field 2*	Item That Gets Text in Field 3
Element (example: 3274)	0	Dictionary element name.
	1	Dictionary element description.
	2	Level 1 note on dictionary element.
	3	Level 2 note on dictionary element.
	4	Level 3 note on dictionary element. See * below for other levels of notes.

Element ~code~part~code set (example: 3279~DO~1~0)	0	Dictionary code value definition.
	1	Dictionary code value explanation.
	2	Dictionary level 1 note on code value.
	3	Dictionary level 2 note on code value.
	4	Dictionary level 3 note on code value. See * below for other levels of notes.

* A field 2 number of 14 or more is a level note greater than level 3. Please see **Level Notes** on page .

The Third Field: the Text

The third field will be the text to be used for the item in field 2 when in the location in field 1. The following is an example .TEXT,ELMS section with the third field in each record shown in bold.

`.TEXT,ELMS`

`3036,0,`**`Party name (sending)`**`n/l`

`3036,1,`**`Name of sending party involved in transaction.`**`n/l`

`3036,2,`**`This is the name of the sending party; omit all punctuation such as periods after St. and Ave.`**`n/l`

Record length: Some records can be very long in this section. For the example above, we have marked the new-lines with **n/l** .

Symbols that you might encounter include:

\r	carriage return
\n	line feed
\r\n	carriage return and line feed (equal to a new line)
\r\n\r\n	results in one blank line in the text

Example .TEXT,ELMS Section

Instead of realistic text, the third field has been changed to make it clearer to you where the text goes:

```
.TEXT,ELMS
4435,0,New name for  dictionary element 4435
4435,1,New description for dictionary element 4435: Identification of the
channel of payment.
4435,2,Level 1 note for dictionary element 4435
4435,3,Level 2 note for dictionary element 4435
4401~AD~1~0,0,New definition for AD code in 4401
4401~AD~1~0,1,New explanation for AD code in 4401
4401~AD~1~0,2,Level 1 note for dictionary element 4401's code AD
4401~AD~1~0,3,Level 2 note for dictionary element 4401's code AD
```

Complete TEXT Section Charts

functional group	level 1 note	level 2 note	level 3 note	name or title	purpose	description	definition	explanation	semantic note	comment	set note	segment note	set comment	
1	3	4	5	0	2									transaction set or message
	4	5	7	3	6						0	1	2	segment in transaction set or message
	0	1	4	2										element in transaction set or message
	0	1	4	2										composite in transaction set or message
	0	1	4	2										subelement in composite in transaction set or message
	0	1	4											code on element in transaction set or message
	0	1	4											code on subelement in transaction set or message
	2	3	4	0	1									segment in segment dictionary
	3	4	7	5					1	2				element in segment dictionary
	3	4	7	5										composite in segment dictionary
	3	4	7	5										subelement in segment dictionary
	0	1	4											code on element in segment dictionary
	0	1	4											code on subelement in segment dictionary
	2	3	4	0	1									composite in composite dictionary
	3	4	7	5					1	2				subelement in composite dictionary
	0	1	4											code in composite dictionary
	2	3	4	0		1								element in element dictionary
	2	3	4				0	1						code in element dictionary

level 3 note (*see* **Level Notes** *on page 60 for additional levels*)

Note levels beyond level 3 will have a number of 10 plus the level. Each level's number is consistent for notes on all items (example: a level 6 note will have a number of 16, regardless of whether it is on a segment in the transaction set or a code in the composite dictionary).

Examples

Level	Corresponding Number
4	14
5	15
6	16
45	55
112	122

.TEXT,SETS

`850,0,`*`some text`*	Title of transaction set 850.
`850,1,`*`some text`*	Functional group.
`850,2,`*`some text`*	Purpose of transaction set 850.
`850,3,`*`some text`*	Level 1 note on transaction set 850.
`850,4,`*`some text`*	Level 2 note on transaction set 850.
`850,5,`*`some text`*	Level 3 note on transaction set 850.
`850,`*`n`*`,`*`some text`*	Level *n-10* note on transaction set 850. *n* is 14 and above. Example: `850,32,`*`some text`* is a level 22 note.
`850~3,3,`*`some text`*	Name of third segment in transaction set 850.
`850~3,4,`*`some text`*	Level 1 note on third segment in transaction set 850.
`850~3,5,`*`some text`*	Level 2 note on third segment in transaction set 850.
`850~3,7,`*`some text`*	Level 3 note on third segment in transaction set 850.
`850~3,`*`n`*`,`*`some text`*	Level *n-10* note on third segment in transaction set 850. *n* is 14 and above. Example: `850~3,14,`*`some text`* is a level 4 note.
`850~3,6,`*`some text`*	Purpose of third segment in transaction set 850.
`850~3~4,0,`*`some text`*	Level 1 note on 4th element in 3rd segment of transaction set 850.
`850~3~4,1,`*`some text`*	Level 2 note on 4th element in 3rd segment of transaction set 850.
`850~3~4,4,`*`some text`*	Level 3 note on 4th element in 3rd segment of transaction set 850.
`850~3~4,`*`n`*`,`*`some text`*	Level *n-10* note on 4th element in 3rd segment of transaction set 850. *n* is 14 and above. Example: `850~3~4,124,`*`some text`* is a level 114 note.
`850~3~4,2,`*`some text`*	Name of 4th element in 3rd segment of transaction set 850.

`850~21~4,0,`*`some text`*	Level 1 note on composite in 4th position of the segment in the 21st position in transaction set 850.
`850~21~4,1,`*`some text`*	Level 2 note on composite in 4th position of the segment in the 21st segment in transaction set 850.
`850~21~4,4,`*`some text`*	Level 3 note on composite in 4th position of the segment in the 21st position in transaction set 850.
`850~21~4,`*`n`*`,`*`some text`*	Level *n-10* note on composite in 4th position of the segment in the 21st segment in transaction set 850. *n* is 14 and above.
	Example: **`850~21~4,33,`*`some text`*** is a level 23 note.
`850~21~4,2,`*`some text`*	Name on composite in 4th position in the 21st segment in transaction set 850.
`850~21~4~1,0,`*`some text`*	Level 1 note on the 1st subelement in the comp. at the 4th position of the segment at the 21st position in the transaction set 850.
`850~21~4~1,1,`*`some text`*	Level 2 note on the 1st subelement in the comp. at the 4th position of the segment at the 21st position in the transaction set 850.
`850~21~4~1,4,`*`some text`*	Level 3 note on the 1st subelement in the comp. at the 4th position of the segment at the 21st position in the transaction set 850.
`850~21~4~1,`*`n`*`,`*`some text`*	Level *n-10* note on the 1st subelement in the comp. at the 4th position of the segment at the 21st position in the transaction set 850. *n* is 14 and above.
	Example: **`850~21~4~1,17,`*`some text`*** is a level 7 note.
`850~21~4~1,2,`*`some text`*	Name of 1st subelement in the comp. at the 4th position of the segment at the 21st position in the transaction set 850.
`850~3~6~LNF~1~0,0,`*`some text`*	Level 1 note on LNF code on 6th element in 3rd segment of transaction set 850 (see **Two-part codes** on page 65).
`850~3~6~LNF~1~0,1,`*`some text`*	Level 2 note on code LNF on 6th element in 3rd segment of transaction set 850 (see **Two-part codes** on page 65).
`850~3~6~LNF~1~0,4,`*`some text`*	Level 3 note on code LNF on 6th element in 3rd segment of transaction set 850 (see **Two-part codes** on page 65).
`850~3~6~LNF~1~0,`*`n`*`,`*`some text`*	Level *n-10* note on code LNF on 6th element in 3rd segment of transaction set 850 (see **Two-part codes** on page 65). *n* is 14 and above.
	Example: **`850~3~6~LNF~1~0,15,`*`some text`*** is a level 5 note.
`850~21~4~1~LNF~1~0,0,`*`some text`*	Level 1 note on the LNF code for the 1st subelement in the comp. at the 4th position of the segment at the 21st position in the transaction set 850 (see **Two-part codes** on page 65).
`850~21~4~1~LNF~1~0,1,`*`some text`*	Level 2 note on the LNF code for the 1st subelement in the comp. at the 4th position of the segment at the 21st position in the transaction set 850 (see **Two-part codes** on page 65).

`850~21~4~1~LNF~1~0,4,`*`some`* *`text`*	Level 3 note on the LNF code for the 1st subelement in the comp. at the 4th position of the segment at the 21st position in the transaction set 850 (see **Two-part codes** on page 65).
`850~21~4~1~LNF~1~0,`*`n`*`,`*`some`* *`text`*	Level *n-10* note on the LNF code for the 1st subelement in the comp. at the 4th position of the segment at the 21st position in the transaction set 850 (see **Two-part codes** on page 65). *n* is 14 and above.
	Example: **`850~21~4~1~LNF~1~0,14,`*`some text`*** is a level 4 note.

.TEXT,SEGS

`MEA,0,`*`some text`*	Name of MEA segment in segment dictionary.
`MEA,1,`*`some text`*	Purpose of MEA segment in segment dictionary.
`MEA,2,`*`some text`*	Level 1 note on MEA segment in segment dictionary.
`MEA,3,`*`some text`*	Level 2 note on MEA segment in segment dictionary.
`MEA,4,`*`some text`*	Level 3 note on MEA segment in segment dictionary.
`MEA,`*`n`*`,`*`some text`*	Level *n-10* note on MEA segment in segment dictionary. *n* is 14 and above.
	Example: **`MEA,20,`*`some text`*** is a level 10 note.
`MEA~0,2,`*`some text`*	Comment on MEA segment in the segment dictionary, second element.
`MEA~2,1,`*`some text`*	Semantic note on MEA segment in the segment dictionary, second element.
`MEA~2,3,`*`some text`*	Level 1 note on second element in MEA segment in segment dictionary.
`MEA~2,4,`*`some text`*	Level 2 note on second element in MEA segment in segment dictionary.
`MEA~2,7,`*`some text`*	Level 3 note on second element in MEA segment in segment dictionary.
`MEA~2,`*`n`*`,`*`some text`*	Level *n-10* note on second element in MEA segment in segment dictionary. *n* is 14 and above.
	Example: **`MEA~2,100,`*`some text`*** is a level 90 note.
`MEA~2,5,`*`some text`*	Name of second element in MEA segment in segment dictionary.
`MEA~4,3,`*`some text`*	Level 1 note on composite at MEA04 in segment dictionary.
`MEA~4,4,`*`some text`*	Level 2 note on composite at MEA04 in segment dictionary.
`MEA~4,7,`*`some text`*	Level 3 note on composite at MEA04 in segment dictionary.
`MEA~4,`*`n`*`,`*`some text`*	Level *n-10* note on composite at MEA04 in segment dictionary. *n* is 14 and above.
	Example: **`MEA~4,55,`*`some text`*** is a level 45 note.
`MEA~4,5,`*`some text`*	Name of composite at MEA04 in segment dictionary.

`MEA~2~3B~1~0,0,some text`	Level 1 note on code 3B in second element in MEA segment in segment dictionary (see **Two-part codes** on page 65).
`MEA~2~3B~1~0,1,some text`	Level 2 note on code 3B in second element in MEA segment in segment dictionary (see **Two-part codes** on page 65).
`MEA~2~3B~1~0,4,some text`	Level 3 note on code 3B in second element in MEA segment in segment dictionary (see **Two-part codes** on page 65).
`MEA~2~3B~1~0,n,some text`	Level n-10 note code 3B in second element in MEA segment in segment dictionary (see **Two-part codes** on page 65). n is 14 and above.
	Example: `MEA~2~3B~1~0,75,some text` is a level 65 note.
`MEA~4~8,3,some text`	Level 1 note on 8th subelement in the composite at MEA04 in the segment dictionary.
`MEA~4~8,4,some text`	Level 2 note on 8th subelement in the composite at MEA04 in the segment dictionary.
`MEA~4~8,7,some text`	Level 3 note on 8th subelement in the composite at MEA04 in the segment dictionary.
`MEA~4~8,n,some text`	Level n-1 note on 8th subelement in the composite at MEA04 in the segment dictionary. n is 14 and above.
	Example: `MEA~4~8,14,some text` is a level 4 note.
`MEA~4~7,5,some text`	Name of 7th subelement in the composite at MEA04 in the segment dictionary.
`MEA~4~7~ABC~1~0,0,some text`	Level 1 note on the ABC code of the 7th subelement in the composite at MEA04 in the segment dictionary (see **Two-part codes** on page 65).
`MEA~4~7~03~1~0,1,some text`	Level 2 note on code 03 of the 7th subelement in the composite at MEA04 in the segment dictionary (see **Two-part codes** on page 65).
`MEA~4~7~03~1~0,4,some text`	Level 3 note on code 03 of the 7th subelement in the composite at MEA04 in the segment dictionary (see **Two-part codes** on page 65).
`MEA~4~7~03~1~0,n,some text`	Level n-1 note on code 03 of the 7th subelement in the composite at MEA04 in the segment dictionary (see **Two-part codes** on page 65). n is 14 and above.
	Example: `MEA~4~7~03~1~0,20,some text` is a level 10 note.

.TEXT,COMS

`C001,0,some text`	Name of composite C001 in composite dictionary.
`C001,1,some text`	Purpose of composite C001 in composite dictionary.
`C001,2,some text`	Level 1 note on composite C001 in composite dictionary.
`C001,3,some text`	Level 2 note on composite C001 in composite dictionary.
`C001,4,some text`	Level 3 note on composite C001 in composite dictionary.

`C001,`*n*`,`*some text*	Level *n-1* note on composite C001 in composite dictionary. *n* is 14 and above.
	Example: `C001,18,`*some text* is a level 8 note.
`C001~0,1,`*some text*	General semantic note on composite C001 in composite dictionary.
`C001~2,2,`*some text*	Comment on second subelement in composite C001 in composite dictionary.
`C001~4,3,`*some text*	Level 1 note on 4th subelement in composite C001 in composite dictionary.
`C001~4,4,`*some text*	Level 2 note on 4th subelement in composite C001 in composite dictionary.
`C001~4,7,`*some text*	Level 3 note on 4th subelement in composite C001 in composite dictionary.
`C001~4,`*n*`,`*some text*	Level *n-1* note on 4th subelement in composite C001 in composite dictionary. *n* is 14 and above.
	Example: `C001~4,150,`*some text* is a level 140 note.
`C001~4,5,`*some text*	Name of 4th subelement in composite C001 in composite dictionary.
`C001~4~AB~1~0,0,`*some text*	Level 1 note on code AB of 4th subelement in composite C001 in composite dictionary (see **Two-part codes** on page 65).
`C001~4~ZZ~1~0,1,`*some text*	Level 2 note on code ZZ of 4th subelement in composite C001 in composite dictionary (see **Two-part codes** on page 65).
`C001~4~ZZ~1~0,4,`*some text*	Level 3 note on code ZZ of 4th subelement in composite C001 in composite dictionary (see **Two-part codes** on page 65).
`C001~4~ZZ~1~0,`*n*`,`*some text*	Level *n-1* note on code ZZ of 4th subelement in composite C001 in composite dictionary (see **Two-part codes** on page 65). *n* is 14 and above.
	Example: `C001~4~ZZ~1~0,44,`*some text* is a level 34 note.

.TEXT,ELMS

`1608,0,`*some text*	Name of element 1608 in element dictionary.
`1608,1,`*some text*	Description of element 1608 in element dictionary.
`1608,2,`*some text*	Level 1 note on element 1608 in element dictionary.
`1608,3,`*some text*	Level 2 note on element 1608 in element dictionary.
`1608,4,`*some text*	Level 3 note on element 1608 in element dictionary.
`1608,`*n*`,`*some text*	Level *n-1* note on element 1608 in element dictionary. *n* is 14 and above.
	Example: `1608,21,`*some text* is a level 11 note.
`I33~D1~1~0,0,`*some text*	Definition for code value D1 on element I33 in element dictionary (see **Two-part codes** on page 65).

`I33~D1~1~0,1,`*`some text`*	Explanation for code value D1 on element I33 in element dictionary (see **Two-part codes** on page 65).
`I33~D1~1~0,2,`*`some text`*	Level 1 note for code value D1 on element I33 in element dictionary (see **Two-part codes** on page 65).
`I33~D1~1~0,3,`*`some text`*	Level 2 note for code value D1 on element I33 in element dictionary (see **Two-part codes** on page 65).
`I33~D1~1~0,4,`*`some text`*	Level 3 note for code value D1 on element I33 in element dictionary (see **Two-part codes** on page 65).
`I33~D1~1~0,`*`n`*`,`*`some text`*	Level *n-1* note for code value D1 on element I33 in element dictionary (see **Two-part codes** on page 65). *n* is 14 and above.
	Example: **`I33~D1~1~0,14,`*`some text`*** is a level 4 note.

Two-part codes

You may have noticed that, when the "where" field is a code, it almost always ends with ~1~0. For example:

1608~ABC~1~0

part   *code set ordinal*

This describes code ABC on element 1608 in the element dictionary. The first item after the code itself is the code value part. A few elements have two-part code values, and this could be set to ~2 to mean the second part of a two-part code. Otherwise, it is set to ~1 for a one-part or the first part of a two-part code value. The second item after the code is the code set ordinal. It is set to ~0 if there is no code set ordinal.

Abbreviated Example SEF

```
.VER 1.6

.INI

KAVERPO,,004 030,X,X12-4030,Draft Standards Approved for Publication by ASC X12
Procedures Review Board through October 1999

.PRIVATE FORESIGHT

.CTL *

.TDVER 6382

.DATE 02/24/100:15:04:53

.PUBLIC

.STD ,RE

.SETS

850=^+100[ST,M][BEG*1,M]+200[.CUR]+100[.REF,,>1][PER,,3][.TAX,,>1][.FOB,,>1][.CTP,,>1]
+50[.PAM,,10]+150[.CSH,,5]+50[.TC2,,>1]{:25[.SAC][.CUR]}[ITD,,>1]+100[DIS,,20]+50[.INC
][.DTM,,10]+300[LIN,,5]+50[SI,,>1][PID,,200]+100[MEA,,40][.PWK,,25][.PKG,,200][TD1,,2]
[TD5,,>1][TD3,,12][TD4,,5][.MAN,,10]+60[.PCT,,>1]+40[.CTB,,5]+50[.TXI,,>1]{:>1+5[LDT]+
3[QTY,,>1]+2[MTX,,>1]+5[REF,,>1]}{:>1[AMT]+20[REF,,>1]+10[DTM]+20[.PCT,,>1]{:>1+10[.FA
1][FA2,M,>1]}}{:1000[.N9]+20[DTM,,>1]+30[MTX,,>1]+50[PWK,,>1]+30[EFI,,>1]}{:1+20[&N1]+
100[N2,,2]+50[.IN2,,>1][.N3,,2]+100[.N4,,>1]+50[.NX2,,>1][.REF,,12]+100[.PER,,>1]+50[.
SI,,>1][.FOB]+100[.TD1,,2][.TD5,,12][.TD3,,12][.TD4,,5][.PKG,,200]}{:>1[.LM][LQ,M,>1]}
{:>1[.SPI][REF,,5][DTM,,5][MTX,,>1]{:20[N1][N2,,2][N3,,2][N4][REF,,20][G61][MTX,,>1]}{
:>1[CB1][REF,,20][DTM,,5][LDT][MTX,,>1]}}{:>1[.ADV][DTM,,>1][MTX,,>1]}^{:100000+100[PO
1,M]+50[LIN,,>1]+30[SI,,>1]+20[CUR]+50[.CN1][.PO3,,25]{:>1+100[.CTP]+30[.CUR]}+20[.PAM
,,10]+40[.MEA,,40]{:1000+10[PID*1]+100[MEA,,10]}[PWK,,25]+200[.PO4,,>1]+100[.REF,,>1][
.PER,,3]{:25+200[.SAC]+50[.CUR]+20[.CTP]}+30[.IT8]+20[.CSH,,>1]+80[.ITD,,2]+100[.DIS,,
20]+50[.INC][.TAX,,>1]+100[.FOB,,>1][.SDQ,,500][.IT3,,5][.DTM,,10]+250[.TC2,,>1]+50[.T
D1]+100[.TD5,,12][.TD3,,12][.TD4,,5]+60[.PCT,,>1]+40[.MAN,,10]+90[.MTX,,>1]+10[.SPI,,>
1][.TXI,,>1][.CTB,,>1]{:>1[.QTY][.SI,,>1]}{:200[.SCH][.TD1,,2][.TD5,,12][.TD3,,12][.TD
4,,5][.REF,,>1]}{:200+50[.PKG][.MEA,,>1]}+100[.LS]{:>1+10[.LDT][.QTY,,>1][.MTX,,>1][.R
EF,,3]{:>1[.LM][LQ,M,>1]}}[.LE]{:1000+30[.N9]+20[.DTM,,>1]+30[.MEA,,40]+50[.MTX,,>1][.
PWK,,>1]+30[.EFI,,>1]}{:200+20[.N1]+100[.N2,,2]+50[.IN2,,>1][.N3,,2]+100[.N4]+30[.QTY,
,>1]+20[.NX2,,>1]+50[.REF,,12]+100[.PER,,3]+50[.SI,,>1]+10[.DTM]+40[.FOB]+50[.SCH,,200
][.TD1,,2]+100[.TD5,,12][.TD3,,12][.TD4,,5][.PKG,,200]{:>1+20[.LDT][.MAN,,10][.QTY,,5]
[.MTX,,>1]+10[.REF,,3]}}{:1000[.SLN]+50[.MTX,,>1][.SI,,>1]+100[.PID,,1000][.PO3,,104]+
50[.TC2,,>1]+80[.ADV,,>1]+20[.DTM,,10]+10[.CTP,25][.PAM,,10][.PO4][.TAX,,3]{:>1+40[.N
9]+10[.DTM,,>1][.MTX,,>1]}{:25[.SAC][.CUR][.CTP]}{:>1[.QTY][.SI,,>1]}{:10+50[.N1][.N2,
,2][.IN2,,>1][.N3,,2]+100[.N4][.NX2,,>1][.REF,,12][.PER,,3]+50[.SI,,>1]}}{:>1[.AMT]+10
0[.REF]+20[.PCT,,>1]}{:>1+80[.LM]+100[LQ,M,>1]}}^{:1+100[!CTT][AMT*1]}[SE,M]

.SEGS

AAA=[1073,M][559][901][889]

ACD=[1636][650][1262]

ACK=[668,M][380][355][374][373][326]{10[235][234]}[559][822][1271]+P0203C0405P0708P091
0P1112P1314P1516P1718P1920P2122P2324P2526P2728C282729

ACS=[610,M][150,M][352][146]
.
.(most segments omitted to save space)
.
ZR=[762,M][206,M][207,M][186][373][3][140][373]{2[1127]}{2[127]}[202]

ZT=[214,M][206,M][207,M][186][373]+P0405
```

```
.COMS
C001=[355,M][1018][649][355][1018][649][355][1018][649][355][1018][649][355][1018][649
],...------------,..&...........
C002=[704,M][704][704][704][704]
C003=[235,M][234,M]{4[1339]}[352]
C004=[1328,M]{3[1328]}
C005=[1369,M]{4[1369]}
C006=[1361,M]{4[1361]}
C007=[522,M][522][1638][935][344][1637][935][352]
C022=[1270,M][1271,M][1250][1251][782][380][799]+P0304
.
.(most composites omitted to save space)
.
C040=[128,M][127,M][128][127][128][127]+P0304P0506,..,......,..&.
C041=[373,M][624]
C042=[426,M][127]
C043={2[1271,M]}[98]
C045=[1321,M]{4[1321]}
C046=[122,M]{4[122]}
C050=[1675,M][1570,M][799,M][1565,M]{2[1675][1570][799][1565]}+P05060708P09101112
.ELMS
1=AN,1,13
2=N0,1,6
3=AN,1,60
4=ID,3,3
5=ID,3,5
7=AN,6,17
.
..(most elements omitted to save space)
.
I61=ID,1,1
I62=ID,1,1
I63=ID,1,1
I64=AN,2,6
.CODES
8=E,R
9=C1,D1:D6,E1:E6,L1:L9,LA
11=A:H,J,M,P:Y
16=A:E,P
23=2,A:L,N,P:Q,S:V,Z
33=A,D:E,O:P,W
39=A:E,I,S,Z
40=20,2B,2D:2G,40,4B,AC,AF,AL,AP,AT,BC,BE:BH,BJ:BK,BO,BR,BX,CA:CD,CG:CN,CP:CX,CZ,DD,DF
,DT,DX,ET,FH,FN,FP,FR:FT,FX,GS,HB:HC,HO:HP,HT,HV,HY,ID,IX,LO,LS,LU,NX,OB,OT,OV,PL,PP,P
```

```
T:PU,RA,RC:RG,RI,RO,RR:RT,SA,SC:SD,SK:SL,SR:ST,SV,TA:TC,TF:TR,TT:TW,UA:UE,UL,UP,VA,VE,
VL,VR:VT,WR,WY

47=A,M

48=1:4

51=C,P

52=1:9,10:17,G,V

54=BY,FE,PP,SE,ZZ

56=BB,CS,CY,DD,DR,HH,HL,HP,MD,NC,PH,PP,RD:RE,RR

66=1:9,10:50,53:59,61:63,71:78,81:82,90:99,A,A1:A6,AA,AC:AE,AL,AP,BC:BE,BG,BP,BS,C,C1:
C2,CA:CF,CI,CL:CM,CP,CR:CT,D,DG,DL,DN,DP,DS,EC,EH:EI,EP:ES,FA:FD,FI:FJ,FN,GA,GC,HC,HN,
K,LC:LE,LI,LN,M3:M6,MA:MD,MI,MK:ML,MN,MP,MR,N,NA,NI,NO,OC,OP,PA:PC,PI,PP,PR,RA:RE,RT,S
A,SD,SF,SI:SJ,SL,SP,ST,SV:SW,TC,TZ,UC,UL,UP,US,WR,XV,XX,ZC,ZN,ZY:ZZ%[9,92]+850/180///3
+850/41///3

72=D,I,N

.

...(most elements omitted to save space)

.

I62=1,2,Z

I63=1,2,3,4,5,6

.VALREFS

//ISA//6=DUNS

850/182///1=RECVDOC

850/185///2=REFID

850/2///3=PO-NUM

///310/=LOC-CODE

.VALLISTS

DUNS="Kaver DUNS number"

'78-250-12250001

LOC-CODE="Kaver Corp. regions"

'Eastern division

'Great Lakes N.

'Great Lakes S.

'Midwest 1

'Midwest 2

'Northeast N

PO-NUM="Purchase Order Number"

&^P98-[0-9][0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]$

RECVDOC="Receiving Dock"

'Loading Dock 3

'Loading Dock 437

REFID="Reference ID"

&^REF[A-Z][A-Z][0-9][0-9][0-9]$

.SEMREFS

850/180///4=[N103DUNS'EQ'9:APPVALUE:DUNS:]

850/1///=[:COMEXIT:CheckCTT.SetCheckCTTCount:]

///1251/=[:COMEXIT:DateTime.ValidateDateTimeX12:]
```

850/41///=[BEG08Invoice'EQ'IBM:USAGE:2:]

.OBJVARS

850/180///3=N103DUNS

850/2///8=BEG08Invoice

.TEXT,SETS

850,0,Purchase Order for Andr\233 Leblanc

850,3,This guideline goes into effect in June, 1999.

850,4,Will the Eastern Division be ready for this by June? Coordinate with Pat.

850~41,3,Name - Bill To

850~72,4,Customer must use the assigned code for element 559. The sales and service division requires this information. \r\nCustomer must use the assigned code for element 235. The warehouse tracking software requires this information.

850~180,3,Name - Ship To

850~72~5,0,This element is optional for ASC X12 but mandatory for our company.

850~72~5,1,Have we updated the old purchasing system for these codes? Check with Systems Support.

850~72~5~PC~1~1,0,This is not an ASC X12 code but we prefer it since orders process faster if it is used.

.TEXT,ELMS

639~PC~1~1,0,Price Cap

Index

Symbol Index

position numbers 14
PRIVATE section 7

R

RE in STD section 8
record length 8
regular expressions 39
repeat counts for elements 21
repeating elements 8, 24, 26
repeating patterns of elements 21
required 22
responsible agency 6

S

sections
  CODES 4
  COMS 4
  duplicate 5
  ELMS 4
  INI 3, 5, 6
  LS 13
  OBJVARS 4, 40
  omitted 5
  order 5
  OVERHEAD 4
  overview 3
  PUBLIC and PRIVATE 3, 7
  SEGS 4, 18
  SEMREFS 4, 41
  SETS 3, 8
  STD 3
  TEXT,COM 4
  TEXT,ELMS 4
  TEXT,SEGS 4
  TEXT,SETS 4
  VALLISTS 4, 38
  VALREFS 4, 38
  VER 3, 5
segment dictionary 18
SEGS section 18
semantic rules 41
SEMREFS section 41
sequence numbers 14
sequence numbers for groups 8
SETS section 8
SETS symbol summary 17
standard name 6

STD section 8, 13
symbols 27
symbols in CODES 37
symbols in SEGS 26
syntax rules
  segments 21

T

tables 13
TEXT,COMS section 54
TEXT,ELMS section 57
transaction sets 9

U

USAGE 43
user requirements 20

V

VALLISTS 38
VALLISTS section 38
VALREFS 38
VALREFS section 38
variable names 40
VER section 5
version of SEF 5
vertical bar 35
VRI 6

X

X 27, 30