Etasoft
Building blocks of integration

Beginners Guide to EDI X12 (including HIPAA)

Main website http://www.etasoft.com
Products website http://www.xtranslator.com

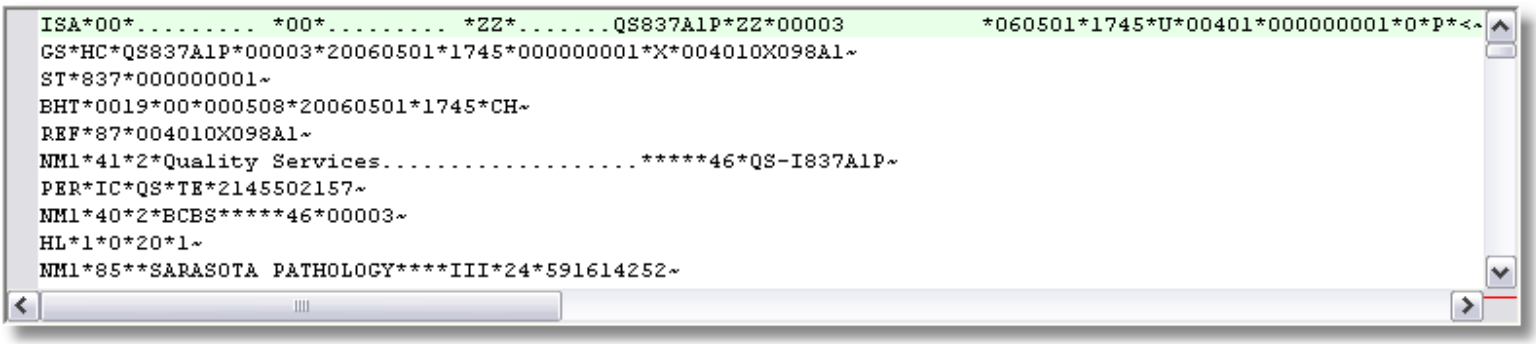
Purpose

The purpose of this document is to explain basics of EDI X12 standard format including its application for HIPAA.
Document uses a lot of common EDI X12 and HIPAA terms. Some of the terms are also specific to Etasoft Inc. products.

What is EDI X12

Just to put it simply - EDI X12 (Electronic Data Interchange) is data format based on ASC X12 standards. It is used to exchange specific data between two or more trading partners. Term ‘trading partner’ may represent organization, group of organizations or some other entity. In most cases it is just organization or company.

You will see term ‘trading partner’ used heavily in computer programs that perform actual data communication, and programs that tie data communication with specific translation.



There is an example of typical EDI X12 file. In this case it is Healthcare Claim EDI X12 837 release version 4010.

EDI X12 standards and releases

EDI X12 is governed by standards released by ASC X12 (The Accredited Standards Committee). Each release contains set of message types like invoice, purchase order, healthcare claim, etc. Each message type has specific number assigned to it instead of name. For example: an invoice is 810, purchase order is 850 and healthcare claim is 837.

Every new release contains new version number. Version number examples: 4010, 4020, 4030, 5010, 5030, 6010, etc. Major releases start with new first number. For example: 4010 is one of the major releases, so is 5010. However 4020 is minor release. Minor releases contain minor changes or improvements over major releases.
Understanding the difference between major and minor releases is important. Let say you have working translation for some messages for release 4010, and if you want to upgrade to 4020 you will notice only a few changes between the two, and if you want to upgrade to release 5010 you might need to make a lot of modifications to current translation.

At the time of this writing 4010 is most widely used release. It is the first release that is Y2K compliant. Most of HIPAA based systems know and use 4010. 5010 is next major release that is gaining popularity and will replace 4010 in the future.

Conclusion: to translate or validate EDI X12 data you need to know transaction number (message numeric name) and release version number. Both of those numbers are inside the file.

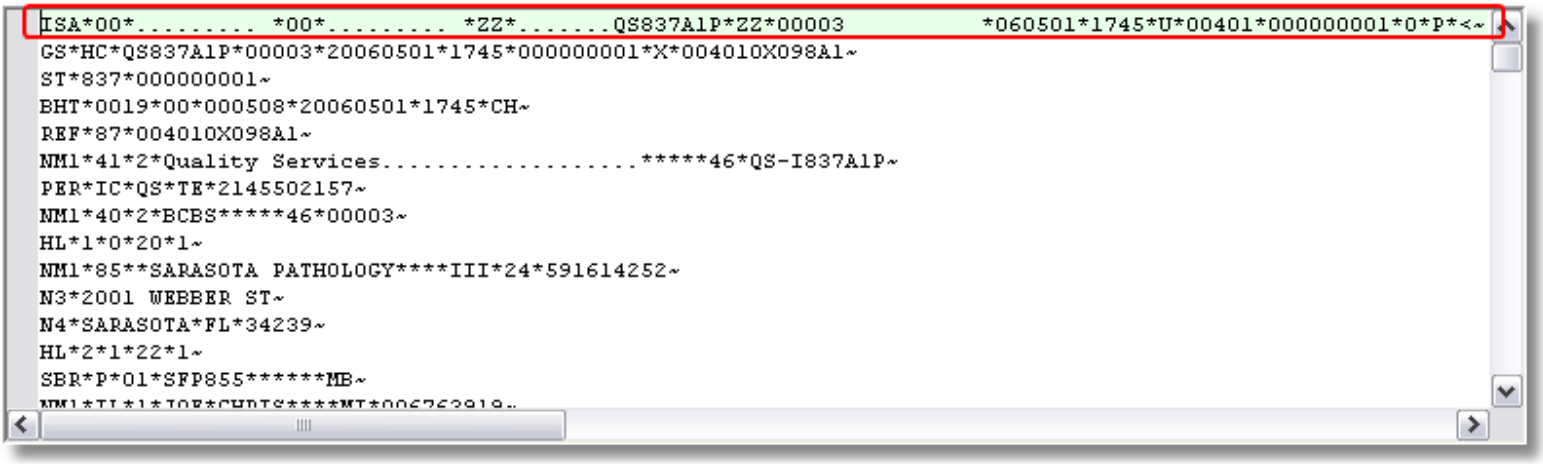
Trading Partner Requirements

EDI X12 standard covers number of requirements for data structure, separators, control numbers, etc. However many big trading partners impose they own even more strict rules and requirements. It can be everything: specific data format requirements for some elements, requirement to contain specific segments (segments that are not mandatory in EDI X12 standard being made mandatory), etc. In HIPAA those specific trading partner requirements are usually listed in separate document called Companion Guide. It is essential to follow these documents to the letter when implementing EDI systems.

Those business requirements make EDI system development highly specific to each implementation. This is one of the reasons why EDI software is expensive to build, run and maintain.

EDI X12 Dissected

Standard EDI X12 format data is text file separated by segment, element and sub-element delimiters (separators). You can open EDI X12 files using any text editor even standard Windows notepad.exe utility. Carriage return and line feed are not required characters by EDI X12 standard. If they are not present in the file after each segment separator you will see continues line of data in the typical text editor. It is very hard to read data formatted that way. In our examples we will use files with carriage return and line feed following segment separator, so each segment will be on a separate line.

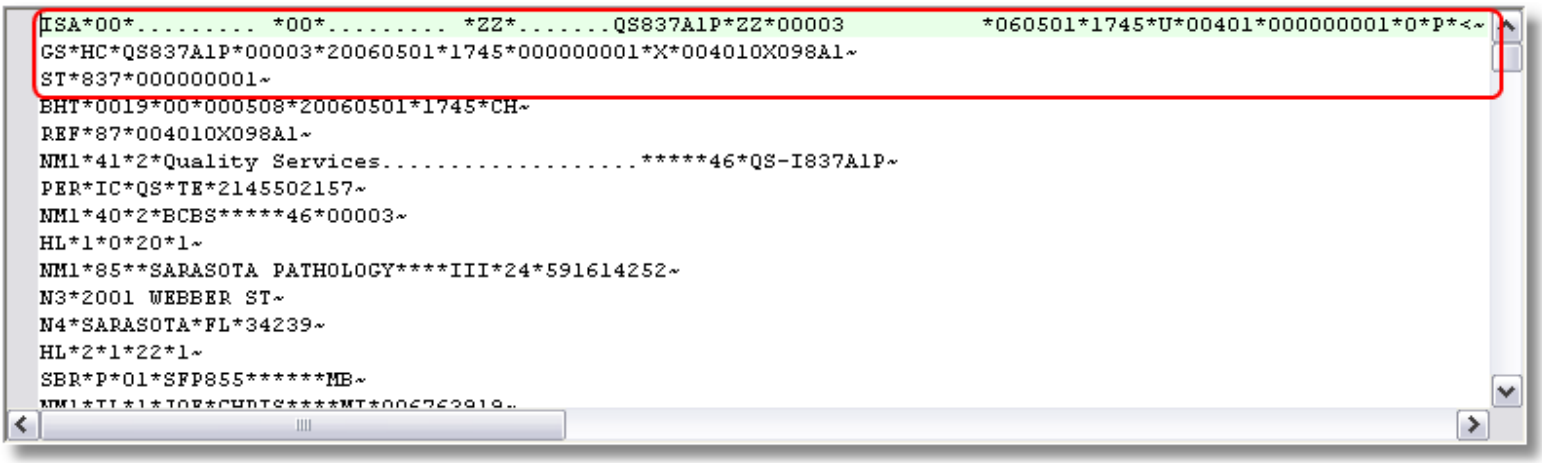


There is typical EDI X12 837 Healthcare Claim (HIPAA) release version 4010.

Each segment is displayed on the separate line. In this example each segment ends with ~ (tilde). That is so called segment separator or segment delimiter. Each segment starts with 2-3 letter code that identifies it. Example: ISA, GS, ST, BHT are all segment identifiers. Each segment contains elements separated by element separator. In our example it is * (star).

While in our case separators are printable characters like tilde and star, they do not have to be. They can be other characters like <, > (greater than or less than signs) | pipe sign, also non-printable characters. Most translators detect separators for incoming EDI X12 files.

Some segments form EDI X12 envelope. They are common to all EDI X12 files and message types. Those segments are ISA, GS, ST, SE, GE, IEA. This set contains important information about trading partners (like Sender Id, Receiver Id, etc.). It also contains interchange, transaction group and transaction control numbers, counts, transmission dates and times, and more.



These are traditional EDI X12 envelope segments at the beginning of the file.

These are traditional EDI X12 envelope segments at the end of the file.

Enveloping segments work in pairs. ISA-IEA represents an interchange. GS-GE is a group inside of interchange and ST-SE is a transaction inside the group.

```
ISA*00*.......... *00*.......... *ZZ*.......QS837AlP*ZZ*00003          *060501*1745*U*00401*000000001*0*P*<~
GS*HC*QS837AlP*00003*20060501*1745*000000001*X*004010X098Al~
ST*837*000000001~
BHT*0019*00*000508*20060501*1745*CH~
REF*87*004010X098Al~
NM1*41*2*Quality Services.....................*****46*QS-I837AlP~
PER*IC*QS*TE*2145502157~
...
... (other segments go here)
...
SE*158*000000001~
GE*1*000000001~
IEA*1*000000001~
```

Those are three layers of enveloping. Each layer may repeat multiple times.

```
ISA*00*.......... *00*.......... *ZZ*.......QS837AlP*ZZ*00003          *060501*1745*U*00401*000000001*0*P*<~
GS*HC*QS837AlP*00003*20060501*1745*000000001*X*004010X098Al~
ST*837*000000001~
BHT*0019*00*000508*20060501*1745*CH~
REF*87*004010X098Al~
NM1*41*2*Quality Services.....................*****46*QS-I837AlP~
PER*IC*QS*TE*2145502157~
NM1*40*2*BCBS*****46*00003~
HL*1*0*20*1~
NM1*85**SARASOTA PATHOLOGY****III*24*591614252~
N3*2001 WEBBER ST~
N4*SARASOTA*FL*34239~
HL*2*1*22*1~
SBR*P*01*SFP855******MB~
```

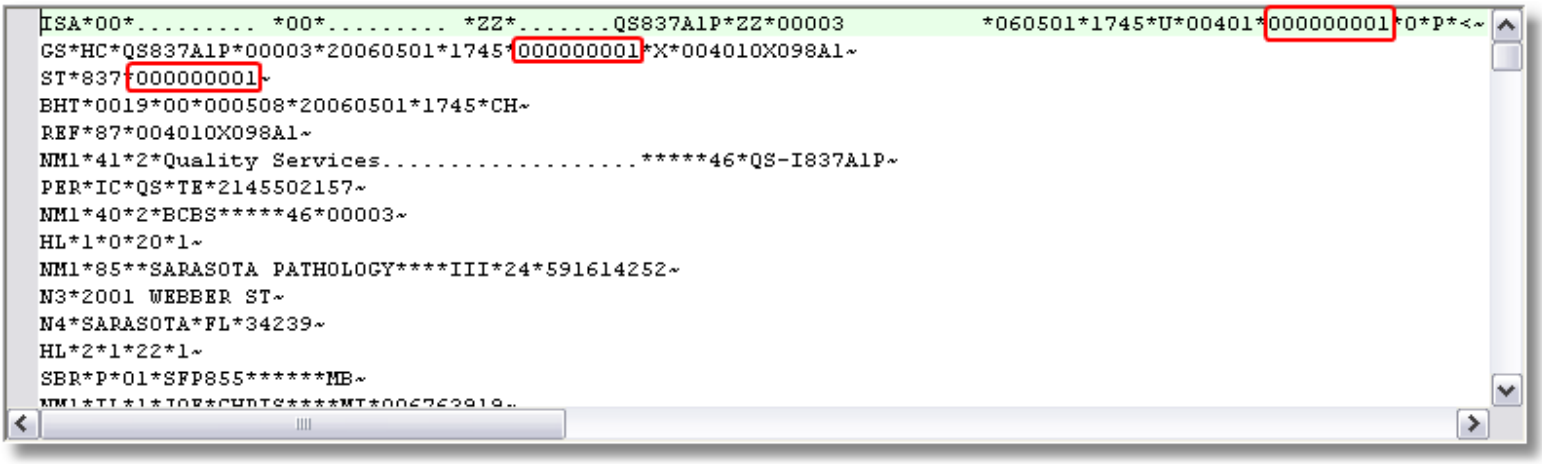
These are segment identifiers. Based on those we can have basic idea what data is contained in the segment.

```
ISA*00*.......... *00*.......... *ZZ*.......QS837AlP*ZZ*00003          *060501*1745*U*00401*000000001*0*P*<~
GS*HC*QS837AlP*00003*20060501*1745*000000001*X*004010X098Al~
ST*837*000000001~
BHT*0019*00*000508*20060501*1745*CH~
REF*87*004010X098Al~
NM1*41*2*Quality Services.....................*****46*QS-I837AlP~
PER*IC*QS*TE*2145502157~
NM1*40*2*BCBS*****46*00003~
HL*1*0*20*1~
NM1*85**SARASOTA PATHOLOGY****III*24*591614252~
N3*2001 WEBBER ST~
N4*SARASOTA*FL*34239~
HL*2*1*22*1~
SBR*P*01*SFP855******MB~
```

Some segments repeat more than once. In most cases they have different qualifiers inside in order to be identified. For example: there are number of NM1s in this file. Some of them located even at the same level (same loop). But each one has it's own qualifier as first element. So NM1*41 contains different information than NM1*40.

Control Numbers

Each layer in the envelope contains specific control number. The processing software to identify successful and failed interchanges, groups and transactions uses control numbers. Acknowledgements (997) use control numbers to report back processing results.



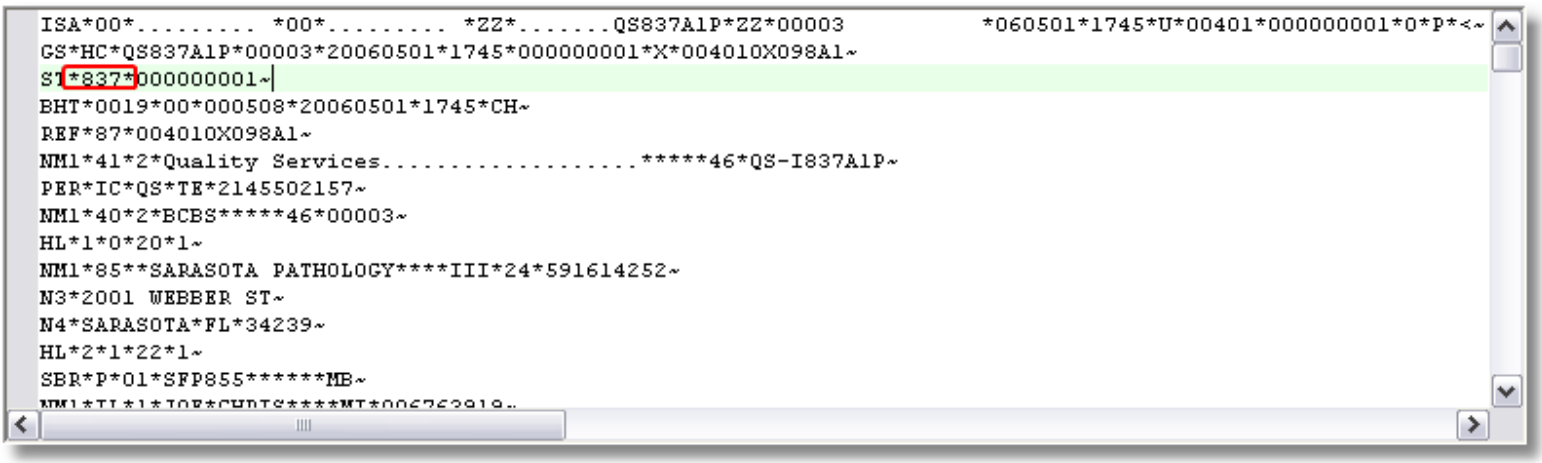
These are control numbers for interchange, group and transaction.

Control numbers at the end of EDI X12 file have to match numbers at the beginning of the file.

Control numbers have specific format and length requirements. They must also be unique. They are not required to be sequential by the EDI X12 standard however some trading partners impose more strict rules in order to track data exchange.

Transaction number

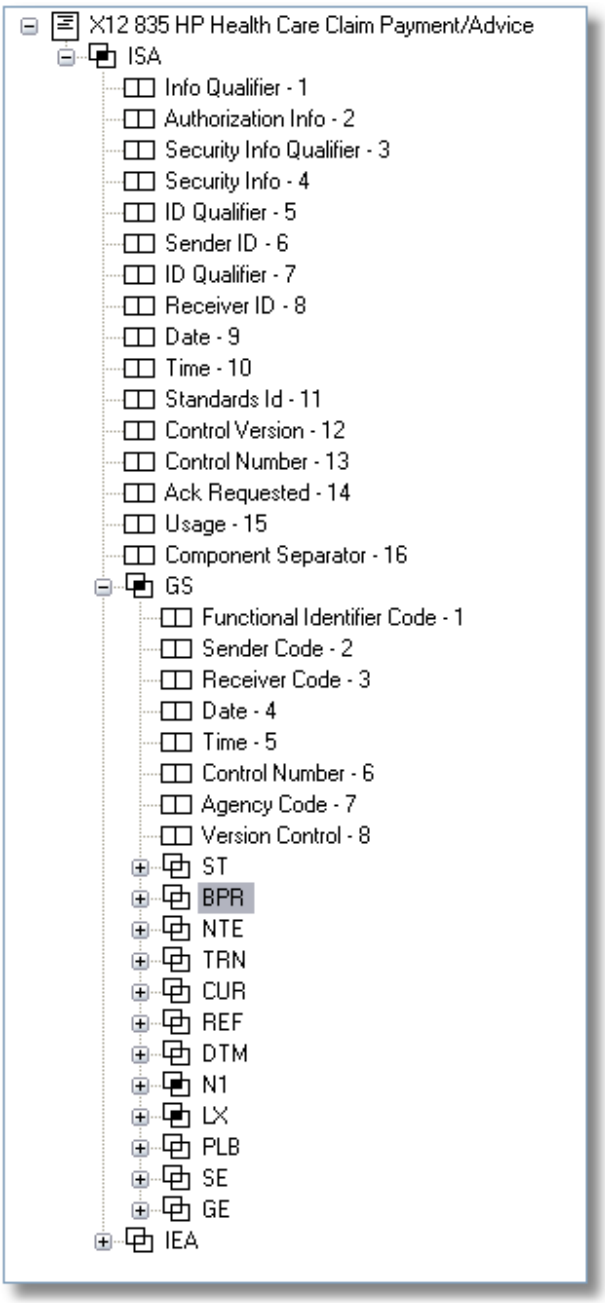
Transaction number or message numeric name is always first element inside ST segment. Sometimes one EDI X12 file may contain number of transactions and those transaction numbers' might also be different. For example: it may contain invoices (810) or purchase orders (850) with acknowledgements (997) in one and the same file. Usually files with combined mix of transactions in them are much harder to process especially if each interchange comes with separate set of separators (delimiters).



There is transaction number (message numeric name).

EDI Loops

If you look at the typical EDI X12 file, it is very hard to see that there are loops (blocks of repeating data) in it. Unlike XML format EDI X12 does not have such concept as “closing tag”. So it is not obvious where one block ends and another begins. Only by looking at EDI X12 standard documentation you can be sure and see how loops are defined. Many software packages come with templates for EDI X12 transactions. Those templates indicate looping by showing blocks of segments nested in one another in some kind of tree structure.



There is an example of EDI X12 835 looping structure imported into Etasoft XTranslator product via Template Wizard. Loops indicated via nested tree structure.

Most loops are simply blocks of repeating segments. For example: claim lines in Healthcare Claim 837 transaction or detail lines in Purchase Order 850. However some transactions contain hierarchical looping structure where loops can deepen number of times with child loops containing some numeric ID of the parent. Typical example could be Healthcare Claim 837 transaction. It is much harder to extract data from or produce data into hierarchical looping structure.

Rules and Imperfections

While EDI X12 format rules are strict real world implementations sometimes try to bend them. EDI X12 was designed to make various systems talk to each other. While most software vendors try to follow rules to the letter some let them slip away. We call

those cases 'imperfections' as most of them are not impossible to fix or workaround. This short chapter will try to list some of the typical implementation imperfections. Some of the cases listed here are not clear breakings of the rules but you may find them annoying.

Example 1:
EDI X12 data that is coming from mainframe or some other old systems is broken in lines of 80 characters. This is easily fixable by removing carriage return and line feed before data is being fed into translator. Removal of carriage return and line feeds will make EDI data as continues stream.

Example 2:
EDI files have no predefined size limits. Typical files range from 5Kbt to 5Mbt. Sending 1Gbt is not a rule breaker but having multiple smaller files instead can speed up the process. Many software packages allocate memory based on incoming data size, huge files can simply make processing system run out of memory or make it run much slower and cut memory for other services and programs on the same machine.

Example 3:
EDI files should have enveloping segments ISA, GS, ST, SE, GE, IEA. You may encounter EDI documentation from your trading partner that does not list or mention them at all. It is because trading partner mutually assumes that you know about those segments and will not even mention them in they documentation.
Only EDI files used internally by the organization may not contain ISA, GS, ST, SE, GE, IEA segments because VAN or some other third party gateway performs all the file preprocessing and routing and might be stripping them off. This is not an error but many shrink-wrapped EDI software packages will not be able to deal with files missing the envelope. They will consider EDI data as invalid.

What Are Invalid EDI X12 Files?

Once you open EDI file in some text editor like Notepad, it looks cryptic. However all valid EDI X12 files have same properties:

1. They all start with three letters: "ISA"
2. After 106 characters there is word "GS". Sometimes "GS" is on the second line. If "GS" is on the second line then there might be 108 characters between "ISA" and "GS" (there are two non-printable carriage return and line feed characters between lines). There are also few exceptions to this rule. Example: if your segment separator is carriage return then you have 105 characters in ISA and one non-printable carriage return character at the end of line with GS following it.
3. ISA segment is 106 characters long and sometimes longer. Even 108 characters if you have ~ tilde segment separator and carriage return and line feed characters at the end of segment (after the tilde).

Most translators read and break down EDI files using these two points above. Most common scenarios why translator might not be able to process the file:

1. That is not an EDI X12 file. Period. It is flat file, XML or scanned print image form or something else but EDI file.

2. If you copy and paste EDI file from Internet Browser window in most cases resulting EDI file will not be valid anymore. This is due to how spaces are displayed in the Internet Browsers. Instead of 103 characters between ISA and GS you might end up having much less.

3. Usually EDI files are sent over secure connections using encryption/decryption communications. Sometimes not just communications but EDI files are encrypted themselves. Encryption leaves ISA/GS part valid but all segments in the file unreadable. Unless EDI file is decrypted translator will not be able to read it correctly.

4. Some EDI files come from mainframe computer systems. Occasionally they are formatted into columns of 80 characters long. Extra carriage return and line feed characters break segments into multiple lines making EDI file invalid. Those files can not be processed by some translators but Etasoft XTranslator can process them.

5. Rarely EDI software communication packages combine multiple EDI messages of various types into one file. There are two variations of this combination:

a) Example: 810 invoices and 997 acknowledgements with the same separators in one file. This is valid EDI file however output of translator might not look right. Example: in translations from EDI to flat files translator will try to flatten EDI loops and encounter few loops repeating independent of each other (as per example above: 810 invoices will repeat separately

from 997 loops). It is like trying to combine two spreadsheets that have no common fields; at the end combined spreadsheet will have mix of lines that do not tie up.

b) Example: 810 invoices and 997 acknowledgements with the different separators in one file. This is not possible to process by many translators. Usually translators detect separators at the start of the EDI file. If separators change somewhere in the middle of the file, translator will continue processing the file using old separators found at the file start. Scenario when various EDI messages are mixed in one file with different separators is very rare but we mention it here to complete possible list of files that can not be processed.

HIPAA

The Health Insurance Portability and Accountability Act (HIPAA) was enacted by the U.S. Congress in 1996.

Key EDI transactions are:

* 837: Medical claims with subtypes for Professional, Institutional, and Dental varieties.
* 820: Payroll Deducted and Other Group Premium Payment for Insurance Products
* 834: Benefits enrollment and maintenance
* 835: Electronic remittances
* 270/271: Eligibility inquiry and response
* 276/277: Claim status inquiry and response
* 278: Health Services Review request and reply

These standards are X12 compliant, and are grouped under the label X12N.

Essentially they are EDI X12 4010 or 5010 release transactions. All the basic standard EDI X12 rules listed in this document apply to the HIPAA as well because HIPAA are EDI X12 transactions.

Deviations Are Standard?

While EDI X12 formats are based on the standards in real world there are many deviations from the standard. Some deviations are so prevalent they become de facto sub standards in they own right. One of the best examples is EDI X12 837 Healthcare Claim. Officially there is only one EDI X12 837. But due to very different requirements for institutional, dental and professional medical claims there are subsets for 837I, 837D and 837P. There is also probably a few hundred if not thousand variations of these. Many trading partners you might need to deal with will provide you with they own document called “Companion Guide”. These additional documents list specific requirements to they business needs.

In order to combat these deviations many independent software vendors release they products with Map Editors, Mappers, Mapping tools, etc. Tools allow flexible customization and mapping of translation and validation rules to your requirements. Other messages that have number of deviations are EDI X12 835, 834, 810, 850, 855.
As a rule of thumb more complex the message structure is – more possible deviations of it are in real world implementations.

What does it mean for your implementation? Why those deviations matter?
Let’s take 837 as an example. It is important to understand that every EDI X12 837 is unique within context of trading partner. Yes, they are all based on standard 837. Yes, they are all targeted for specific sectors: 837P, 837I, 837D, and more etc. So you logically assume they are all identical. If you take two EDI X12 837s from two different trading partners chances are you will find at least few different segments in each. That means in most cases you need your implementation to include specifics.

Etasoft EDI Products

Each Etasoft product has been designed to cover specific set of functionality. Example: XTranslator – translates, EDI Validator – performs validation. All products have graphical user interface: Map Editors, Map Runners, Script Runners, Script Editors, File Editors, etc.

Almost all products come with batch processing command line utilities to help automate the processing. While batch utilities are harder to work with and require some initial getting used to, they also easy to integrate into other automated systems.

XTranslator product comes with Developer SDK. You can execute translation functions from your programs and embed functionality into your software packages. Product also supports extensibility via external plug-ins or scripts. You can add your own functions that are specific to your business. However do not go too far with it. Example: while translator can be extended to support some EDI validation, there is another product designed for EDI validation. Another example: you can write translator plug-ins to perform secure FTP transfer, but there is already XT Server product that does it.

All products come as fully functional time limited trail versions. They do not have any other limitations but evaluation period time limit. Only after evaluation period runs out, functions are restricted. It is never a good idea to use evaluation versions of the products in production systems.

EDI System Components

Basic EDI system components are:

1. Products: XTranslator. EDI translator – stand-alone or build-in part of almost every EDI system. Performs translations for incoming and outgoing EDI messages. We provide separate document that explains the differences in various translators, they strengths and weaknesses.

2. Product: EDI Validator. EDI validation – stand-alone or build-in part of high volume EDI systems. Performs EDI validation. It checks control numbers, segment counts, absence of mandatory segments, validates mandatory values, counts loops.

3. Product: XT Server. EDI processing, scheduling and communication package. Performs data transmission, allows execution at certain times or time intervals, reaction to file system events. Usually works as stand-alone server process or service. It is used in automated EDI systems.

4. Product: Alpha Forms OCR. Prepares scanned paper forms for EDI translator processing. Paper scanned forms tend to be in image file format that needs to be converted into electronic text in order to be fed into EDI translators.

5. Product: Mini Translator. EDI reporting tool – extracts and shows EDI data in human readable form. This is little brother to a typical EDI translator. While translators are designed to translate EDI data for other computer systems to understand, EDI reporting tools are like advanced EDI viewers with basic translation capabilities designed to produce simple human readable formats out of EDI files.

6. Basic database loader. Many EDI translators are able to load data directly into the relational databases. However companies choose not to use this feature and build they own database loaders in-house. This is mostly due to the higher security requirements and general non trust towards external processes feeding data into the backend production databases. In-house database loaders also perform business checks and sanity checks on incoming data from the EDI translator. Example of checks: since loader has access to the database it can query existing records and make sure that duplicates are not loaded based on some identifiers. It can also make an intelligent decision should new record be inserted or updated.