Implementation Guide
for
Gas Industry Electronic Commerce
Using the Internet

Prepared by the

Gas Industry Standards Board
Future Technology Task Force

3rd Draft
September 11, 1996

Assembled by Cynthia Johnson, NGPL and Susan Croley, PanEnergy
with substantial input from Terry Lehn, Enron and John Tsucalas, Enron

document standards
Document Font: Book Antiqua
Title on title page: 18 pt
Other text on title page: 14 pt and 12 pt
Footer: 10 pt, Italic
Section heading: 14 pt
Paragraph heading: 12 pt
Text: 10 pt.
all sample code and data will be in shadow boxes with 10% shading, arial 10 font

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Purpose

The Future Technology Task Force (FTTF) of the Gas Industry Standards Board (GISB) has developed standards for protocols to accomplish electronic commerce using the Internet. The FTTF determined technologies necessary to rapidly, reliably and safely move a transaction set across the Internet. Once received, the transaction set is moved through an X12 translator and to a back-end processing application. However, X12 translation and back-end processing are outside the FTTF scope. This document is concerned with the delivery from the output of one company's translator to the input of another company's translator.

This document is a high-level guide to implementing various technologies necessary to communicate transactions using the protocols identified by the FTTF. As such, this guide is not intended to be a comprehensive, in-depth manual. Wherever possible, this guide points to more in-depth material. The Reference section provides locations on the Internet to obtain more information as well as books and periodicals that proved useful to FTTF members.

1.2 Standards

The standards adopted by the FTTF, as with all GISB standards, should be adhered to by the trading parties as minimum standards. A trading party may offer additional functions or features as options but should not require their use. Such additional features or functions are termed "mutually agreed to" in that if both trading partners agree on the inclusion, the additional feature requirements will be met. However, if either trading party does not agree to the inclusion of additional features, then the partners must allow for transmission and receipt of data using the minimum standards.

The Trading Partner Agreement is a key reference in electronic commerce. It will define the "designated site" for each partner (see the Business Practices Subcommittee documentation), values used for variable parameters, and optional features that will be used by the partners.

2. EXECUTIVE SUMMARY

2.1 Roles in Electronic Commerce

In all electronic commerce, one party initiates, or sends, a transaction and the other party receives the transfer. In the Internet environment, the sender is referred to as the client and the receiver is referred to as the server. You should expect to act in both the client role and the server role during the electronic commerce process. Once a transaction set is successfully received for processing, the original receiving party switches to the client role to send a confirmation transaction back to the original sender's server. Therefore, it is essential that both the sending and receiving aspects of electronic commerce are addressed in your implementation.

2.2 Assess your capabilities

Depending on your situation, you may implement the complete solution in-house. Given the existence of in-house systems expertise, it should be possible to implement the technologies in this guide with little, if any, assistance. On the other hand, smaller organizations may want to use this guide to identify services that they will obtain from a third-party.

As much as possible, the FTTF chose technologies where most of the programs needed to implement could be acquired as "shrink-wrapped" software at low cost. Where commercial quality products that can just be "plugged in" do not exist, sample code has been identified. This sample code has the drawback of being unsupported. It is intended for companies that have technical expertise but need just some starter code from which to build their own versions.

A mixture of in-house expertise and third-party services will be the likely approach of several organizations. To determine where you may require the services of a third-party, you should assess your present capabilities. For example, a company may have experience with X12 translators, but little experience with Internet technology at this time.

2.3 In-house implementation

If you are choosing to implement most or all of the required functionality in-house, this document is particularly pertinent. It attempts to capture the "lessons learned" from those companies that participated in the pilot.

The FTTF has demonstrated through the pilot test that electronic commerce using the Internet can work. However, the FTTF strongly encourages all parties to fully investigate the ramifications of introducing electronic commerce using the Internet. This includes ensuring that all customer data, internal data, and applications are secure from intruders or other parties not authorized for access.

An investment in Internet technology must be made which will involve hardware, software, and technical expertise. Hardware investments will include a server to

receive incoming EDI files, a firewall processor to block intruder access, a proxy server to ???..... Software investments will include operating software for the servers, including the firewall and proxy servers, programming languages which support Internet technologies, and encryption/decryption software to provide security during the transfer. Technical expertise investments will include development and maintenance of server applications to process incoming files as well as applications to initiate communication with the server of your trading partner.

2.4 Using a Third-Party

It is expected that third-party providers will offer a variety of services from a full "turn key" solution to assistance only where you require it. Such assistance might include programming, system configuration and system administration as well as private communication links such as provided by VANs.

Several third-party providers have participated in the GISB process and particularly on the FTTF. Several companies in the gas industry have utilized a Value Added Network (VAN) provider to assist with the communication of X12 transactions. These providers are also likely sources of expertise and packaged solutions.

3. TECHNICAL IMPLEMENTATION

3.1 Technology Selected by the FTTF

The FTTF is a sub-task force of the GISB Electronic Delivery Mechanisms (EDM) task force. The EDM task force determined that TCP/IP should be the transport protocol for communication of future GISB transactions. In addition, they directed that, in general, standard Internet protocols should be chosen for specific tasks. The FTTF considered various Internet protocols to accomplish to delivery of a transaction at the application protocol level. They chose the Hyper-Text Transfer Protocol (HTTP). HTTP is also known as the Web technology.

There are two primary Internet software components involved in Web communications. The first is called a browser and runs as client software. The second is called a Web server, or HTTP server and usually runs on a dedicated server computer.

The data elements, element name, and element description have been defined by the FTTF and are outlined in the Data Dictionary section. The following two sections identify what is involved in sending and receiving transactions. After that comes a discussion regarding the securing of the transactions to be sent. The remaining sections cover considerations for other aspects of the overall process. While these were not the focus of the FTTF as mentioned above, selected topics that may affect your overall implementation are discussed.

3.2 Data Dictionary For Internet EDI

Business Name	Definition	Format	Usage*	Condition
from	**DUNS number of the party sending the transaction**	**DUNS number format**	**Request #1 Required**	**used in file transmittal; displayed in HTTP response; and, used in posting back decryption-related errors**
input-data	**the filename for the transaction data set transmitted**	**including drive letter and directory name with filename if needed**	**Request #4 Required**	**used in file transmittal of any 10 HPDRs; and, used for posting back all transaction value pairs for a transmittal that had decryption-related errors**
input-format	**descriptor of the data format used for the file transmitted**	**X12; error**	**Request #3 Required**	**"X12" used in file transmittal; "error" used in posting back any decryption-related errors**
request-status	**status describing success or failure of transmission at recipient server**	**ok; EEDM###:error description; WEDM###:warning description see HTTP Response Standard Error Codes and Messages**	**Response #2 Required**	**"ok" is returned if all is fine with the CGI processing; error messages/warnings and their related descriptions are returned if problems were encountered in CGI processing or in the decryption process**
server-id	**uniquely identifies the server and cgi processing the transaction**	**up to 20 characters long**	**Response #3 Required**	**displayed in the HTTP response and posted back for any decryption-related errors**
time-c	**the time file transfer is complete at the server**	***yyyymmddhhmmss***	**Response #1 Required**	**displayed in the HTTP response and posted back for any decryption-related errors**
to	**DUNS number of the party the transaction was sent to**	**DUNS number format**	**Request #2 Required**	**used in file transmittal and displayed in HTTP response and posted back for any decryption-related errors**
transaction-set	**descriptor of the transaction types included in the input-data.**	**The values used must be from the unique 8-character names define by the GISB _____. See_____ for the list. When a file contains mixed transaction types, use _____.**	**Request #5 Mutually-Agreed-To**	**used in file transmittal**
trans-id	**sequential number assigned to the transaction by the server cgi upon processing before being passed to the decryption process**	**integer up to 15 characters in length**	**Response #4 Required**	**displayed in the HTTP response and posted back for any decryption-related errors**

*** The Usage column defines whether the element appears in the HTTP Request (Client-generated) or the HTTP Response (Server-generated), the order in which the element appears in the data stream, and whether the field is Required or Mutually-Agreed-To.**

4. SENDING TRANSACTIONS

4.1 HTTP Post

Most people think of the Web as the process of using a browser to fetch, or download, documents, not upload them. Indeed, this capability is most prevalent. It utilizes the GET method of the HTTP protocol.

HTTP also provides for uploading of files using another method known as POST. The POST method, like the GET, returns a document from the server. It is this method which will be used to send GISB X12 transactions and receive the response from the server. It happens that the GET method is also capable of uploading data but is limited to 1024 characters which is why the POST method is necessary.

4.2 Using an interactive browser

When most of us think of Web surfing, we think of using an on-line browser such as NCSA's Mosaic, Netscape's Navigator or Microsoft's Internet Explorer. When you enter an HTTP Uniform Resource Locator (URL), the browser opens the HTML document identified by the URL. Basically, a URL is an "address" of an HTML document on a Web server.

In order to use an interactive browser to upload data, an HTML document must be created for that function. The HTML document can reside on either the server to which you are uploading or the client's system. The "form" feature of HTML allows that within an HTML document, a form can be created which allows the client to type in any necessary data elements, such as userid and input format and then specify a file to be uploaded from the PC. Some type of "Send" button would be on the form and when selected, the form would cause an HTTP POST to be issued, thereby uploading the file. Below is an example of an HTML document with a form which specifies the POST method and contains the required data elements.

An HTML form like that described here could be used with any retail browser that supports multipart POST with a file upload. Among those supporting this are:

- **Netscape Navigator in versions 2.0 or greater**
- **Microsoft Internet explorer version ???**

These extensions are planned in:

- **Mosaic**

Sample of HTML document with a form to perform a multipart post:

```
<HTML>
<HEAD>
<TITLE>GISB File Upload</TITLE>
<H1><CENTER>GISB File Upload</CENTER></H1>
</HEAD>
<BODY>
<hr>
<form ENCTYPE="multipart/form-data" ACTION="http://www.target.server/cgi-bin/upload.exe" METHOD=POST>
Enter DUNS number for From and To
From: <input TYPE="text" NAME="from" SIZE=20 VALUE=""><br>
To: <input TYPE="text" NAME="to" SIZE=20 VALUE=""><br>
Format of this file: <input TYPE="text" NAME="input-format" SIZE=6 VALUE="X12"><br>
Send this file: <INPUT NAME="input-data" TYPE="FILE"><br>
<input TYPE="submit" VALUE="Send File"><br>
</form>
</body>
</html>
```

NOTE: I have been told that the standard format for DUNS number is xx-xxx-xxxx. Should we change this to indicate a 11-character field. Also, there is no guarantee that every company has a DUNS number, is there?]

The non-bolded text in this example is the basic HTML required for a document and allows your page to show a title in the title bar. The bolded text is the form within the document and is described in more detail.

The important characteristics of the form within the HTML document are:

- **ENCTYPE= specifies the encoding type. The FTTF identified "multipart/form-data" as the standard encoding type.**
- **ACTION= specifies the URL that will receive the uploaded data. The Trading Partner Agreement identifies the URLs for both parties.**
- **METHOD= specifies the HTTP protocol method. The FTTF identified "POST" as the standard method.**
- **<input ...> Five input areas are specified on this form: from, to, file format, file name, "Send File" button.**

NOTE: This document often refers to "multipart POST" which implies the encoding type and method as described in this example.

When a user selects the "Send File" button, the browser will take the values entered in the input fields and reformat them according to the encoding type into a data

stream. For the file identified for upload, the file is opened and its contents are included in the data stream, rather than the file's name. The data stream is then sent to the URL specified by ACTION=. The URL will indicate an HTTP server script or program written to receive the data.

For a smaller site only performing a few transactions or file transfers this manual process would be viable as a primary transmission tool. This method could also be considered a back-up method to any batch or automated process that may be implemented. If the client provides its own form, the form can be copied for each trading partner. The only change to the HTML would be to modify the URL shown on the ACTION= line.

4.3 Using a batch browser

For companies that have automated much of their back-end process and prefer to avoid unnecessary human involvement, a so-called "batch browser" is needed. This browser needs to be capable of program-based or script-based initiation. At this time, there are few off-the-shelf batch browsers which use the POST method. Most use the GET method.

The FTTF is pursuing vendors which have GET method batch browsers to request a POST version of their product. Watch the GISB FTTF web page for updates on the availability of such products.

However, a batch browser can be created using custom programming. The batch browser will be coded to perform all of the same formatting that the interactive browser performed to send a data stream which conforms to the HTTP protocol. A batch browser must be coded as a sockets program. See the appendix for an example.

A sockets program can be written with various programming languages which offer the required library:

Language	Library
C	sockets Application Programming Interface (API)
Visual Basic	WinSock OCX (available free from Microsoft)
Java	sockets class in its standard class library
PERL	socket functions

4.4 Authentication

HTTP basic authentication includes a user id and password. Interactive browsers include a basic authentication feature which automatically prompts for user id and password. In a batch browser, the authenticiation must be specifically coded. The user id and password are to be UUEncoded within the document header. UUEncoding utilities are readily available on the Internet as either public domain software or commercial libraries.

4.5 Server Response

The receiving server will send an HTTP response to the client before dropping the client's connection. The response returned from the Web server will contain timestamps that include a timestamp recorded when the final byte from the file upload is received and stored. This timestamp is the official timestamp regarding deadlines set by GISB. This timestamp should be stored with the X12 file on the receiving server. It is also suggested that it be logged by the client as well.

4.6 Throughput Considerations

Note that the performance of the batch browser is, in most cases, critical in meeting deadlines. It is conceivable that it may be called many times for a busy site (such as a pipeline sending quick responses). It should therefore utilize whatever performance techniques that are possible. For example, it may be desirable to write a multithreaded version which can handle a certain number of requests simultaneously with a single copy of the program.

4.7 HTTP Request Data Elements

Required Data Elements (listed in the required order)

1.	**from**	**DUNS number of sending/client company**
2.	**to**	**DUNS number of receiving/server company**
3.	**input-format**	**Descriptor of the data format within the input dataset.**
4.	**input-data**	**The properly formatted file of electronic commerce data.**

Mutually Agreed Upon Data Elements

1.	**transaction-set**	**Descriptor of the transaction types included in the input-data. The values used must be from the unique 8-character names defined by the GISB _____ Committee. See _____ for the list. When data is of mixed transaction types, use value _____.**

4.8 Writing a batch browser

A batch browser needs to simulate the actions of an interactive browser. As stated earlier, the interactive browser will take the HTML form and reformat the information according to the HTTP protocol before it sends the data stream to the HTTP server. The reformatting involves adding a header and placing field delimiters around the data items. A batch browser needs to produce the same kind of data stream and therefore, writing a batch browser requires some specific knowledge of the HTTP protocol. See the Reference Guide in the Appendices section for HTTP protocol references.

First, consider the header:

Sample of a typical header sent to the HTTP server

```
POST c:\execute HTTP/1.0
Referer: http://www.get.a.life/upl.htm
Connection: Keep-Alive
User-Agent: brow v0.1 EDS Corp.
Host: localhost
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
Content-type: multipart/form-data; boundary=---------------------------87453838942833
Content-Length: 5379
```

This information is mostly documentary in purpose. The parts that are important are:

- **The first line: *POST c:\execute HTTP/1.0* indicating that the POST' method is used and which program to call.**

- **The content type line:**

  Content-type: multipart/form-data; boundary=----------------------------87453838942833

  The content-type element indicates that the encoding method is multipart. It also identifies the character string used as the boundary. The boundary will appear between each field as a delimiter.

  The boundary can be any character string that you choose except that it typically starts with one or more - (dashes) and it should be a unique string that is likely not to occur in the document. This is usually accomplished by using either the system clock or a random number so that even if by some remote chance the string appears in the document it would not apear in any re-transmission of the file. It is strongly recommended that a relatively long string be used as a boundary. The boundary when used as a separator has — appended to the front

of the string as you can note by the lines between the data fields. The last boundary also has -- appended to the back of the separator boundary, this is used to indicate to the server program that this is the end of the data.

- **The content length**:

  Content-Length: 5379

  The content-length should match the number of bytes contained in the entity body including the characters in the boundary lines, variable content, blank lines, etc. In essence, it tells the server how much is going to come after this point.

[We're going to have people coding exactly what is above... Is that what we want? e.g. Referer: http://www.get.a.life/upl.htm How should we change it?]

The data portion, or body, sent to the server program would look something like:

```
-----------------------------87453838942833
Content-Disposition: form-data; name="from"

123456789
-----------------------------87453838942833
Content-Disposition: form-data; name="to"

234567890
-----------------------------87453838942833
Content-Disposition: form-data; name="input-format"

x12
-----------------------------87453838942833
Content-Disposition: form-data; name="input-data";
filename=c:\temp\smallnom.bin
Content-Type: application/octet-stream

ISA~00~        ~01~AAA6300300~14~8051611660000  ~14~1506438070000

... more data from the x12 file…

IEA~1~000003616
-----------------------------87453838942833--
```

The important characteristics of the above stream are:

- **The boundary string appears at the beginning of each data field in the body.**

- **For each body data field, two identifiers define the contents of the data field. The Content-disposition identifier defines that "form-data" is contained in the element. The name identifier defines the name of the data element. These names must match the name specified by the FTTF. The name identifier is not completely relevant since the fields should be present in the correct order but this field should be checked to verify the validity of the form content.**

- **The actual data value of the field is always preceded by a carriage return and line feed. This is typically used as a marker for the server program to indicate that a data value will follow. For example, note the blank line preceeding 'Enron' and 'x12' in the above sample. In most programming libraries and commercial products the starting delimiter is "\r\n\r\n" (c notation).**

- **The data field containing the X12 file has two extra identifiers: first the name of the file sent from the source computer "*filename=c:\temp\smallnom.bin*", and second a content type identifier on a separate line. This line should always be shown as: *"Content-Type: application/octet-stream"*. This indicates that the content of the file should be treated as binary and not converted in any manner.**

- **After the contents of the last data field, the boundary appears again as the last item of the form with an additional '--' at the end to indicate the end of the data.**

- **Although the specifications for multipart POST include several variations on this method, the GISB FTTF standards do not include implementing them at this time. The most significant of these variations is to send several files in a single post. Additionally, sending a single file split into more than one post is not expected by the HTTP server.**

The output from the browser is important to the understanding of the processing needed by the server script or program which must interpret the result. The complete data stream from the browser will look like:

```
POST c:\execute HTTP/1.0
Referer: http://www.get.a.life/upl.htm
Connection: Keep-Alive
User-Agent: brow v0.1 EDS Corp.
Host: localhost
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
Content-type: multipart/form-data; boundary=---------------------------87453838942833
Content-Length: 5379

-----------------------------87453838942833
Content-Disposition: form-data; name="from"

123456789
-----------------------------87453838942833
Content-Disposition: form-data; name="to"
```

```
234567890
-----------------------------87453838942833
Content-Disposistion: form-data; name="input-format"

x12
-----------------------------87453838942833
Content-Disposistion: form-data; name="input-data";
filename=c:\temp\smallnom.bin
Content-Type: application/octet-stream

ISA~00~          ~01~AAA6300300~14~8051611660000  ~14~1506438070000

... more data from the x12 file…

IEA~1~000003616
-----------------------------87453838942833--
```

4.9 FTTF Client Specifications

- **The HTTP client should be synchronized to the U.S. Naval Observatory Central Standard Time. Each trading party should observe the client clock over a period of time to determine the amount of "drift" occurring throughout the day. The client should be synchronized as many times per day as necessary to ensure synchronization. The most important time period to ensure synchronization is just prior to the nomination deadline. See the Appendix, "Reference Guide" for public domain software which can be used to sychronize the client.**

- **The HTTP Request will provide all required data elements in the order defined by the FTTF. Any mutually agreed to data elements will follow the required data elements in the data stream.**

5. RECEIVING TRANSACTIONS

5.1 Using a Web server

As was stated above, the FTTF has chosen HTTP using the POST method as the means to upload a transaction. On the receiving side of this HTTP request is the Web server, the second primary component in Web technology. However, the Web server does not actually save the uploaded file. Instead, it hands this responsibility over to a special program which, in effect, extends the Web server's functionality with custom programming. This special program is known as a Common Gateway Interface (CGI) program. Besides storing the file, the CGI program has the task of parsing the incoming HTTP message, noting the time so to create the timestamp, and creating an HTML response to the sending browser.

The FTTF standard places no particular unusual requirements on the Web server. Most commercially available Web servers will provide the needed functionality. However, note the comments regarding performance under "Throughput Considerations" later in this section. It should also be noted that while the current approach to security does not require a Secure Sockets Layer (SSL) or Secure HyperText Transfer Protocol (S-HTTP) capable server, one of these may be a requirement in the future. Note whether the product you are considering provides a secure version capable of either SSL or S-HTTP. (Unfortunately, it is too early to predict which of these, if either, will prevail as an emerging standard.)

Another capability you may wish to consider when choosing a Web server is whether it supports Binary Gateway Interface (BGI) capability. Specifically, this is the capability to run Dynamic Link Library (DLL) equivalents of CGI applications. Microsoft and Progress Software??? call this capability Internet Server Application Programming Interface (ISAPI) while Netscape calls it Netscape Application Programming Interface (NSAPI).

5.2 The CGI Process

A CGI program must be able to parse the multipart form. It accomplishes this by finding the boundary string in the Content-type header and scanning for its occurrences further within the uploaded stream. Upon finding these boundary strings, the program must next determine the content-disposition for each data element. This allows detection of the required text elements as well as the X12 file.

The CGI program is not concerned with the content of the X12 data. In fact, the X12 data will be encrypted (see the Security section). The CGI will merely accept the X12 data and store it as a file.

The CGI will use the Content-length to determine how much data to expect in the body.

5.3 Throughput Considerations

It may be even more critical that the Web server and the associated CGI programs perform well. This is particularly true for pipelines which may expect to see a large number of nomination transactions come in close to the deadline. For the greatest possible throughput, the Web serer should be multithreaded. The CGI program should be multithreaded as well or be small and efficient as is possible with a C program. BGI programming may provide even better performance.
It is also suggested that a Web server and operating system be chosen that allow for scaling to a more powerful computer (possibly multi-CPU). Transaction volumes are likely to be light at first but may become heavy rather quickly.

5.4 Writing the CGI Process

A CGI process is the executable program or module that is called by the HTTP server when it is identified by a POST or GET operation. (In this case we are only concerned with POST method operations.)

When the HTTP server receives a POST it will first read the header and populate environment variables before calling the CGI. A sample header is shown below.

```
POST c:\execute HTTP/1.0
Referer: http://www.get.a.life/upl.htm
Connection: Keep-Alive
User-Agent: brow v0.1 EDS Corp.
Host: localhost
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
Content-type: multipart/form-data; boundary=---------------------------87453838942833
Content-Length: 5379
```

The important point to note is that you will not specifically code the step of reading the header and populating the environment variables, the HTTP server performs it for you. The variables populated are usually listed with the HTTP server documentation.

After reading this header the server will frequently buffer the remaining data transmitted and then call the CGI process specified in the POST statement. Do not assume that the CGI process is called as soon as the header is read. The more common implementations will buffer the entire transmission before calling the CGI. You may want to check your server implemtation if this characteristic is important to you.

The called CGI process will have the following stream available in the standard input (stdin) and most of the header data available in enviroment variables.

```
-----------------------------87453838942833
Content-Disposition: form-data; name="from"
```

```
123456789
-----------------------------87453838942833
Content-Disposition: form-data; name="to"

234567890
-----------------------------87453838942833
Content-Disposistion: form-data; name="input-format"

x12
-----------------------------87453838942833
Content-Disposistion: form-data; name="input-data";
filename=c:\temp\smallnom.bin
Content-Type: application/octet-stream

ISA~00~          ~01~AAA6300300~14~8051611660000  ~14~1506438070000

... more data from the x12 file…

IEA~1~000003616
-----------------------------87453838942833--
```

This process should check for basic validity in the environment variables and the data stream. It will parse the variables/data from the format. The data validations should include:

- **The "REQUEST_METHOD" environment variable is "POST"**
- **The "CONTENT_TYPE" enviroment variable should be "multipart/form-data" and a boundary (see above stream for an example).**
- **Typically the input stream should be in binary mode to accommodate encrypted files.**
- **Each data element should be preceded by the boundary with — on the front of it.**
- **Each data element should contain the correct name on the *Content-Disposition* line.**
- **Each data element should have `\r\n\r\n` (c notation) before the start of the data.**
- **Finding the end of the stream using both content length and the boundary end mark (the boundary with — in front and behind) is usually the best method to detect improperly formatted input.**

Immediately after the CGI validates (as above), parses, and saves the data, the CGI should record the time and construct a response described in the following section.

This response is usually sent from the CGI by writing to the standard output (stdout) of the CGI process.

5.5 FTTF Server Specifications

- **The HTTP server should be synchronized to the U.S. Naval Observatory Central Standard Time. Each trading party should observe the server clock over a period of time to determine the amount of "drift" occurring throughout the day. The server should be synchronized as many times per day as necessary to ensure synchronization. The most important time period to ensure synchronization is just prior to the nomination deadline. See the Appendix, "Reference Guide" for public domain software which can be used to sychronize the server.**

- **The HTTP server will provide an HTTP response to the client according to FTTF standards.**

- **All data element names will be in lower case in both the request and the response.**

- **Carriage returns and line feeds will be ignored in all files.**

- **A field delimiter of "*" will be used in the HTTP response.**

- **No spaces should surround the equal sign or the field delimiter.**

- **The required data elements must appear first in the response.**

- **Additional information can be included after the required elements at the server's discretion.**

  The HTTP response must be enveloped by HTML tags. ***(Is this still true??)***

- **The HTTP response must be no more than 2048 characters.**

- **The first occurrence of the field name within the response will contain the value.**

  (I think this needs to be re-worded.) **If an HTML response is given, all data must be presented in a user-readable fashion. For example, if the required machine-readable fields are embedded in comments, another representation of these fields must be presented to the user. (see the examples below)**

- **The HTTP Server should be configured as port 80. If port 80 is not available, use one of the five FTTF recommended alternate ports: 5713, 6112, 6304, 6874, 7403.**

5.6 HTTP Response Data Elements

Required Data Elements (listed in the required order)

1.	**time-c**	**the time of transfer completion at the server. The format will be yyyymmddhhmmss.**
2.	**request-status**	**a text status indicator by the server. The only defined value at this time is "ok" for a successful transfer. The server should supply a descriptive indication of the error detected following the standards for error codes and messages presented in the Appendix, "FTTF Standard Error Codes and Messages".**
3.	**server-id**	**a text string up to 20 characters in length uniquely identifying the server and CGI that received and processed the file.**
4.	**trans-id**	**A number (integer) up to 15 characters in length uniquely identifying the received transaction file at the server. The trans-id will uniquely identify the file only at the receiving server. A client may receive non-unique trans-ids across multiple servers.**

Samples of HTTP Response Required Data Elements:

successful, plain text format:

```
time-c=19960123203618*
request-status=ok*
server-id=coolhost*
trans-id=232323897*
```

successful, HTML format:

```
<!-- time-c=19960123203618*→
<!-- request-status=ok*→
<!-- server-id=coolhost*→
<!-- trans-id=232323897*→
<html>
<head>
<title>Upload OK</title>
</head>
<h1>Upload OK </h1><br>
<B>File Saved at (time-c): </B>19960123203618<br>
<B>Status (request-status): </B>ok<br>
<B>Server (server-id): </B>coolhost<br>
<B>Transaction ID (trans-id): </B>232323897<br>
</body>
</html>
```

error, plain text format:

```
time-c=19960619082855*
request-status=EEDM106:  Invalid To DUNS Number*
server-id=coolhost*
trans-id=234423897*
```

warning, plain text format:

```
time-c=19960123203618*
request-status=WEDM100:  Transaction Set Sent, Not Mutually Agreed*
server-id=coolhost*
trans-id=532323897*
```

5.7 Using an Internet Service Provider for Web Hosting

If you do not wish to install and maintain a Web server, you may wish to contact an Internet Service Provider (ISP) to provide the hosting service for you. Consider the following when selecting an ISP for Web hosting:

- **limit on storage space for receiving files**
- **ability to meet FTTF standards for HTTP response**
- **accomodation for CGI to meet FTTF standards for validation and processing**

6. SECURITY

6.1 Security Concepts

The FTTF has determined that the security requirements include the current four primary security aspects: data privacy, data integrity, authentication, and non-repudiation.

- **Data privacy: unauthorized parties cannot decipher the content of the data**
- **Data integrity: unauthorized parties cannot modify or corrupt the data**
- **Authentication: The receiver is certain of the identity of the sender.**
- **Non-repudiation: The sender is certain that the receiver received the file.???**

In general, these needs are met by using the Basic Authentication capability of the Web server and the encryption and digital signature capability of the PGP security protocol for securing transactions.

6.2 Understanding PGP

Pretty Good Privacy (PGP) is the name of the chosen security protocol. See the Reference section for software companies which sell software packages to implement the PGP security protocol. This protocol utilizes a public key/private key pair to accomplish secure file transfers. The private key must be known only to the company which generated it. The public key counterpart is shared with trading partners.

Each company must generate its public key and private key pair. The public keys will be distributed using a secure method (eg., Courier mail) to the company's trading partners. You must use the utmost care in protecting your private key. If it is compromised, the security is broken. It is recommended (required???) that a key size of 1024 be chosen when generating the key pair. This provides a significantly secure transaction.

When a company wishes to send transactions to its trading partner, it will use the partner's public key to encrypt the file. Encryption provides data privacy. Only the private key counterpart can decrypt this file. Hence, the need to guard your private key.

When the sending party encrypts the file, it also uses its own private key to "sign" the transaction. The receiving party can use the sender's public key to verify the signature. The digital signature provides non-repudiation. (is this correct??)

6.3 Encryption / Digital Signature

Encryption and signatures are applied to files already encoded with the X12 standards. The X12 encryption feature is an additional layer of encryption within the X12 file. Use of X12 encryption is outside the FTTF scope but does not conflict with PGP.

Encryption and signatures can be accomplished manually for each file using the on-line PGP software, or in an automated (or "batch") fashion using programs to encrypt and sign. Whether encrypting in a manual or automated fashion, it is essential that the correct public of the trading partner be used to encrypt and just as essential that the correct sender's own private key be used to digitally sign the file.

6.4 Decryption/ Signature Verification

After a transaction is received and processed by the CGI program, it is ready to be decrypted and have its signature verified. The receiver's private key is used for decryption. Upon request for signature verification, the PGP software will return a human-readable company name.

It is recommended that all implementors create a process where the name is used to look up the ID of the company in a database table. This ID should match the ID found within the X12 transaction. If the ID is passed along with the decrypted X12 file, a process could be created to verify that the company which sent the transaction is the same as the company identified within the X12 data, once the data has been translated.

6.5 Throughput Considerations

Encryption, digital signing, decryption and signature verification are all very CPU intensive. It is not recommended that decryption or signature verification be performed within the CGI that receives and processes the file. In fact, it would not be a good idea to have these steps performed on the same computer that is attempting to receive transactions at a time close to a deadline. Therefore, it is recommended that the secured or to-be-secured transaction be passed to a separate computer for security processing. This "passing" would likely be accomplished by using the File Transfer Protocol (FTP). The security processing computer should be optimized for CPU and memory.

Because decryption and signature verification are not handled at the time the file is received, the sender will get an HTTP response of successful transfer but doesn't know if the file can be decrypted by the receiver. The FTTF has developed some guidelines for communicating the status of the decryption step. See section 7, Sending Error Notification Transactions.

6.6 Security Requirements

- **Basic Authentication**

  The FTTF has set basic authentication, also known as realm security, as a standard for transmission on the Internet. The userid and password will be assigned by the server party according to site standards. The trading party agreement must identify the userid and password as well as procedures for changing the password, if applicable.

- **PGP File Encryption**

  The FTTF has set file encryption as a standard for transmission on the Internet. For those companies employing encryption, PGP is the standard defined by the FTTF. Those companies who wish to conduct business across the Internet in an unsecure fashion may do so by mutual agreement.

6.7 Security Recommendations

- **Firewall**

  [*the following paragraphs were submitted but I haven't had a chance to edit/merge them.}*

  from David Calvin: A firewall is a 'smart' router that sits between two unique networks. These two networks are usually an internal company network and the Internet. It's purpose is to limit the types of services between these two networks. Services, which are based on 'port numbers' [like ftp, telnet, etc.], are either allowed through the firewall or blocked. Additionally the firewall can let some services be initialized from the internal network, but not initialized from the external network. For instance, http, which usually uses port 80, can be allow internal client machines to 'contact' web servers on the outside of the firewall, but client machines on the external network cannot 'contact' web servers on the internal network. This allows a company the ability to utilize the internet, but not allow unwanted users to gain access to their internal company network.

  from Terry Lehn: The FTTF has defined security mechanisms which aim to provide for the privacy and integrity of the transaction file while it traverses the Internet. It also has described a mechanism which can authenticate the sender to the receiving host (Web server).

  Often, a company's connection to the Internet is intended to provide several other services to its employees who are connected by a Local Area Network. Examples of these services include access to the World Wide Web, use of email, use of file transfer capabilities and publishing content intended for the external world on a Web server. In addition, the LAN will likely have connections to host computers which provide internal services such as file and print sharing, fax and database capabilities. So that availability of these services and confidential internal data

are not compromised by Internet hackers, there should exist a protective mechanism between the internal company network and the Internet. This mechanism is known as a firewall.

Just what is a firewall? It is one or more computers running special software which is designed to provide network security. There are two general mechanisms employed by firewalls: packet filtering and proxy services.

Packet filtering examines important components of the messages such as the address of the sending and target computers and the designator (port number) for a specific application running on the target computer. By doing this, it can prevent access to specific computers or programs on those computer. It can also reject messages from certain computers.

Proxy servers have various capabilities. They can act as relay agents that can examine attempted use of certain features within an application thus limiting access to these features. They can also hide (by substituting its own address) the internal addresses of clients communicating with external hosts. This hiding makes it difficult for would-be attackers to focus on specific internal hosts. Proxy servers can also authenticate and authorize access to other internal hosts using a logon (user id and password) mechanism.

Because firewalls are designed to deal with a broad set of security issues and are not specific to the use of HTTP, neither the FTTF nor this guide attempt to provide specific implementation information. Deciding on a specific firewall architecture, company security policies and choosing between numerous products may require the assistance of consultants. An excellent source which covers this topic in detail is a book entitled "Firewalls and Internet Security: Repelling the Wily Hacker" by William Cheswick and Steven Bellovin.

- **Proxy Server**

7. SENDING ERROR NOTIFICATION TRANSACTIONS

7.1 Error Notification

When a client sends a file to a server, the server responds to the receipt of the file. Though the file may be received correctly, some further processing must be done, such as decryption and X12 translation. The decryption step which will have a pass/fail status and then the X12 general translation step which will have a pass/fail status. The X12 general translation is merely the check that the file meets the X12 standards and has not been corrupted. Further translation and processing of specific transactions and elements is outside the FTTF scope.

When a file passes the decryption step and passes the general translation step, no notifying communication is sent back to the client. However, if either the decryption step or the general translation step fails, an error notification must be sent to the client.

In general, this standard format for error notification applies to the posting of an error message after sender's session has been disconnected; i.e., any further error notification that takes place after the original HTTP Response is returned with an "ok" or WEDM999 for the request-status value.

7.2 Error Notification Data Elements

The data elements for the error notification are the same as those described in Section 4, Sending Transactions, with the exception of the "input-format" and "input-data" elements.

<u>Required Data Elements (listed in the required order)</u>

1.	**from**	**DUNS number of sending/client company, the server company which detected the error**
2.	**to**	**DUNS number of receiving/server company, the client company which sent the data set in error**
3.	**input-format**	**"error"**
4.	**input-data**	**A text block containing the following items:**

orig-from	**The 'from' value from the original transmission**
orig-to	**The 'to' value from the original transmission.**
orig-input-format	**The 'input-format' value from the original transmission.**
resp-time-c	**The 'time-c' value from the original response.**

resp-server-id	**The 'server-id' value from the original response.**
resp-trans-id	**The 'trans-id' value from the original response.**
request-status	**The new status of the transaction, using FTTF standard error codes and messages.**
comments	**Any comments the original receiving server wishes to include.**

Mutually Agreed Upon Data Elements

none defined at this time

Error Notification "input-data" Element Specifications:

- **All data element names will be in lower case in the Error Notification.**
- **Carriage returns and line feeds will be ignored in all files.**
- **A field delimiter of "*" will be used in the Error Notification.**
- **No spaces should surround the equal sign or the field delimiter.**
- **The required data elements must appear first in the response.**
- **Additional information can be included after the required elements at the server's discretion.**
- **The Error Notification (input-data element only ??) must be no more than 2048 characters.**
- **The first occurrence of the field name within the response will contain the value.**

  (I think this needs to be re-worded.) **If an HTML response is given, all data must be presented in a user-readable fashion. For example, if the required machine-readable fields are embedded in comments, another representation of these fields must be presented to the user. (see the examples below)**

Error Notification Example:

```
POST c:\execute HTTP/1.0
Referer: http://www.get.a.life/upl.htm
Connection: Keep-Alive
User-Agent: brow v0.1 EDS Corp.
Host: localhost
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
Content-type: multipart/form-data; boundary=---------------------------87453838942833
```

```
Content-Length: 1958

-----------------------------87453838942833
Content-Disposition: form-data; name="from"

123456789
-----------------------------87453838942833
Content-Disposition: form-data; name="to"

234567890
-----------------------------87453838942833
Content-Disposistion: form-data; name="input-format"

error
-----------------------------87453838942833
Content-Disposistion: form-data; name="input-data"; filename=c:\temp\error.not
Content-Type: application/octet-stream

orig-from=234567890*
orig-to=123456789*
orig-input-format=x12*
resp-time-c=19960619102855*
resp-server-id=coolhost*
resp-trans-id=234423897*
request-status=EEDM601:  Public Key Invalid*
comments=Please contact 1-800-555-1212 for correct public key*
-----------------------------87453838942833--
```

8. CHECKLIST OF TESTING STEPS

8.1 Purpose

FTTF members followed many preliminary steps in testing before the full batch browser and server applications were completed for pilot testing. This checklist was compiled from members' experiences and is intended to provide a series of small achievements leading up to the complete solution.

8.2 Client/Browser

NOTE: Throughout all transfer tests, compare files stored on the server against the source file to ensure that the file transferred intact. While transferring to another company's server, you may have to contact that company to send the file back to you so that you can perform the compare.

1. **Install an interactive browser. Identify an existing Web server from among GISB servers offerring interactive upload for test (see appendix). Transfer test files to that server. File content does not need to be X12 at this point.**

2. **Develop or acquire a batch browser that uses multipart for the encoding methodology. Transfer the same test file as in step 1 to the URL not requiring Realm 1 security.**

3. **Add Realm 1 security to your file transfer, and change the URL to the secure' URL. Continue transfer tests with your batch browser.**

4. **Acquire and install PGP software. Generate your public & private key pair. Download the test server's test public key. Encrypt your data file using this key. Modify your file transfer to send the encrypted file. Continue transfer tests. Request that the test server contact decrypt your file.**

8.3 HTTP Server and CGI

1. **Install Web server. Establish an Internet connection to your server. Ensure that you have ample storage space for transferred files. Ensure that permissions are granted to the directories.**

2. **Acquire or develop an HTML page for interactive file upload (sample code is earlier in this document). Test interactive file upload to your own server using an interactive browser.**

3. **Acquire or develop a CGI program to receive file transfers and process according to FTTF standards. Test transfers to your CGI using your batch browser.**

4. **Transfer an X12 dataset to your server and process it through your translator.**

5. **Copy the CGI to a 'secure' directory where Realm 1 security, or basic authentication, is enabled. Using your batch browser, transfer to both URLs, with and without authentication.**

6. **Generate a second public/private key pair. Use the second key to encrypt a file and transfer the file to your server. Decrypt the file.**

7. **Once your site security is established, contact a GISB company to test transfers against your server.**

8. **Test with various file sizes to ensure that your CGI can process small and large files.**

9. **Request that several other companies and/or several clients within your own company transfer concurrently to ensure that your server can withstand the load.**

10. **Test all success and error conditions of both file transfers and PGP decryption.**

9. APPENDICES

9.1 FTTF STANDARD ERROR CODES AND MESSAGES

These errors and warnings are strictly related to problems found in the recipient CGI or decryption levels of processing before translation. Errors and warnings generated by the client batch browser are assumed to be documented at the client site to distinguish them from problems occurring in the recipient CGI or decryption. Numbering schemes and descriptions should aid in this distinction.

Note: For HTTP error codes see Technical References - HTTP

EEDM###	**standard error format with ### representing a numeric value further processing will not take place**
WEDM###	**standard warning format with ### representing a numeric value further processing will take place**

Validation Code	Description	Data Element	Required vs. Mutually Agreed
EEDM100	**Missing from DUNS number**	**from**	**required**
EEDM101	**Missing to DUNS number**	**to**	**required**
EEDM102	**Missing input format**	**input-format**	**required**
EEDM103	**Missing data file**	**input-data**	**required**
EEDM104	**Missing transaction set**	**transaction-set**	**mutually agreed**
EEDM105	**Invalid from DUNS number**	**from**	**required**
EEDM106	**Invalid to DUNS number**	**to**	**required**
EEDM107	**Invalid input format**	**input-format**	**required**
EEDM108	**Invalid transaction set**	**transaction-set**	**mutually agreed**
EEDM109	**No parameters supplied**	**parameter string**	**required**
EEDM601	**Public key invalid**	**file itself**	**required - security**
EEDM602	**File not encrypted**	**file itself**	**required - security**
EEDM603	**Encrypted file truncated**	**file itself**	**required - security**
WEDM100	**transaction set sent not mutually agreed**	**transaction-set**	**mutually agreed**

9.2 Reference Guide

9.2.1 GISB

- **General GISB FTTF Reference Page: (http://www.neosoft.com/~gisb/fttref.htm). This location provides pointers to samples and further documentation.**

9.2.2 HTTP

- **W3C World Wide Web Consortium. All aspects of HTTP, HTML, and other Web-related topics: http://www.w3.org/pub/WWW/**

- **General information regarding HTTP with basic terminology included: http://www.w3.org/pub/WWW/Protocols/HTTP/1.0/spec.html**

9.2.3 HTML

- **http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html**

- **http://www.interlink-2000.com/guide-to-publishing-html.html**

- **Special Edition Using HTML, Second Edition, Mark Brown, John Jung, and Tom Savola, Que Corporation, 1996**

9.2.4 Time Synchronization Software

- **Public Domain Software for Synchronization to NIST:**

9.2.5 PGP Software

- **Viacrypt**

  Internet: info@viacrypt.com
  WWW: http://www.viacrypt.com
  available for the following operating systems:
  Windows
  Macintosh
  MS DOS

UNIX (platforms):

SunOS 4.1.x (SPARC)
Solaris 2.3, 2.4
IBM RS/6000 AIX
HP 9000 Series 700/800 UX
SCO 386/486 UNIX
SGI IRIX
BSD/OS
DEC Alpha OSF/1
VAX/VMS
VMS Alpha
DG UX AviiON (88/OPEN)

9.3 GISB Servers Offering Interactive Upload for Test

9.4 Frequently Asked Questions

- **Do I have to be a member of GISB to participate in Internet EDI?**

  No, you do not have to be a member of GISB to participate in Internet EDI. However, membership in GISB provides the benefit of notification of GISB-sponsored meetings.