1. Introduction

Where Internet EDM/EDI Fits in Gas Industry Commerce

The scope of Internet EDM/EDI is to address the communication of GISB standard data format transactions (including X12) between one trading partner's translator/processor and another trading partner's translator/processor.

Business Reasons for Using Internet EDM/EDI

The question may be asked, what are the advantages of using Internet EDM to communicate our business transactions in GISB EDI standard data formats as opposed to using Value-added Networks (VANs). As an even broader question, why use EDI standard data formats for transactions at all? With EDI, data already existing in your own computer applications can be used to build nominations and other gas industry transactions. Information from a service provider, such as scheduling, allocation, invoicing, can be mapped to a common format. This common format eliminates the need for the following as these additional steps leave room for errors, unnecessary intervention and complications in processing:

- transfer data from a paper document to an application format input file at each trading partner site
- if electronic files are used, mapping between various application data formats for each and every trading partner

A company that relies on computerized systems to conduct business and exchanges transactions with several trading partners can communicate those transactions more efficiently with EDI standard data formats and with Internet EDM as the communications mechanism. EDI employs standard data formats for all trading partners. By using the public Internet for transmission, a single connection is required, eliminating the complexity of different connection methods for different trading partners. In a VAN environment, transmission of transactions sent to trading partners who use a different VAN may be considerably delayed because of data transfer schedules between the VANs. The Internet EDM solution eliminates this delay because the transaction is sent directly to the trading partner's designated receipt site.

Technology Selected by the GISB

The transport protocol for communication of future GISB transactions should be TCP/IP. In addition, standard Internet protocols should be chosen for specific tasks. Various Internet protocols were considered to accomplish to delivery of a transaction at the application protocol level. The Hyper-Text Transfer Protocol (HTTP) was chosen.

Practical information systems require more functionality than simple retrieval, including

search, front-end update, and annotation. HTTP allows an open-ended set of methods to be used to indicate the purpose of a request. HTTP is also used as a generic protocol for communication between user agents and proxies/gateways to other Internet protocols, allowing basic hypermedia access to resources available from diverse applications and simplifying the implementation of user agents.

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 standard data elements, each with element name and description, have been defined in the Section "Data Dictionary For Internet EDM". 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 Internet EDM process as mentioned above, selected topics that may affect your overall implementation are discussed.

Client/Server Roles in EDM/EDI

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 when using EDM/EDI. Once an EDI 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 EDM/EDI implementation.

Requirements for creating an EDM/EDI system

The GISB home page contains reference material that parties may utilize in evaluating and choosing hardware and software to create an EDM/EDI system. An organization may choose to develop an EDM/EDI solution in-house, or use outside services to use EDM/EDI. The best solution for a particular organization must be determined based on the assessment of specific needs and the resources available to that organization

Third-party providers 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 those provided by VANs.

Communication Protocols

HTTP is the standard protocol and Post is the standard method by which transactions will be transmitted over the public Internet. The content type used to package the GISB

standard format file (including X12) and its related parameters for the HTTP request is **multi part**. This provides more flexibility in the coding of the messaging components in the application because of the way it handles the delimiting of data parts passed in the body of the form.

Sending Transactions (Client)

It is possible to send transactions using widely-available interactive web browsers. This may be appropriate for shippers who do not have a significant number of transactions to send each day.

It was determined that in order to provide the level of automation required by some organizations such as a large pipeline company to handle the volume of transactions and the level of interface needed for possibly many back-end process applications, a fully automated batch browser is a required component of the application. In this form, the batch browser can be an event-driven mechanism used to push the transaction from the sender's previous processes (the back-end application, the translation, and the security process) across the Internet to the trading partner's server site where receipt of the transaction is acknowledged. The automated batch browser would also better serve the logging function of transactions being sent.

Receipt of Transactions (Server)

The receipt of transactions in the multi part HTTP Post request would require some form of Common Gateway Interface (CGI) program in order to send back a response that would notify the batch browser that it has received the transaction and whether the file in its unprocessed form and its parameters were accepted as sent or rejected. This component of the application would be able to parse out the parameters and related file and determine if the appropriate parameters had been transmitted with the file, log the appropriate statistics including a time stamp about the file and parameters, store the file and send the response back to the batch browser with the time stamp and other required response elements. After the appropriate processes have taken place in the CGI, the file would then be forwarded to the security process, any translation necessary, and finally the back-end processor.

Security

Though many decisions as to overall security measures are left to each trading partner and their environment, several security measures were established as standards to ensure a minimum level of confidence in conducting business over the Internet and to provide some uniformity in the implementation of security. Four primary security aspects were considered as vital in providing the level of protection of transactions needed for gas industry commerce: data privacy, data integrity, authentication, and non-repudiation. The FTTF found that these concerns are addressed by the use of encryption and digital signature capability of the Pretty Good Privacy (PGP) security application. Any process used for encryption and decryption compatible with PGP 2.6 (using keys generated with

the RSA algorithm) meets the minimum standard to be applied to files transmitted over the Internet. To prevent unwanted intruders from connecting to the Web sites, basic authentication is the required standard. Additional issues such as firewall security are discussed in the standards, but are considered implementation issues to be addressed by each organization.

2. Executive Summary

The Gas Industry Standards Board (GISB) has developed standards for protocols to accomplish machine-to-machine (batch) electronic commerce using the Internet. This type of data exchange will be referred to as EDM/EDI. Once received from a trading partner via the Internet, EDM/EDI data is decrypted and moved through a translator or other appropriate processor/parser for GISB standard file formats such as X12 and forwarded to a back-end processing application. However, file format translation and back-end processing are outside the Internet EDM/EDI scope. The scope of this transmission mode is limited to the delivery of the output of one company's application to the input of another company's application.

This documentation is a high-level guide to implementing various technologies necessary to communicate transactions using standard Internet protocols. As such, this material 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 have been recommended.

Open Standards

There are several major topic areas related to Internet Electronic Delivery Mechanism covered in this manual. When looking to implement Internet EDM, one should become familiar with the following components of the implementation:

- Communications Protocols
- Sending of Transactions
- Receipt of Transactions
- Security

The “open” standard technologies selected by GISB to address these areas are designed to provide flexibility and scalability. The specific implementation of the standards is dependent upon what fits the trading partner’s needs and available resources. A brief delineation of these components and their relationship to the model are covered at a high level in the Business Process and Practices (Business Process Description) section and in more detail in later sections of this manual.

Same Application Implementation For All Trading Partners

The basic assumption in designing and implementing the Internet EDM application is that it is not platform-specific. What is meant by this is that an organization's Internet EDM application serves the role of communicating with all trading partners in the gas industry no matter what hardware, operating system and programming languages they use at their site. For this reason, testing with other trading partners with a variety of platforms is very important in ensuring that your EDM application is compatible with a range of platforms used by various trading partners.

Testing With Gas Industry Internet EDM Participants

The FTTF meets on an frequent basis by scheduled teleconference or in-person meetings to discuss issues, problems, and further refinement of the GISB standards. These discussions will provide a means to benchmark results and provide feedback to each other on possible enhancements to the participants' implementations. The FTTF provides a forum for organizations to share experiences and technical information about the technology used and the business issues of using this technology.

Importance of the Trading Partner Agreement When Using EDM

The expectations of who will perform what function and how it will be accomplished in Internet EDM should, at some level, be laid out in the trading partner agreement. This clarification in the agreement would help to expedite a smoother communication between the trading partners when first setting up their Internet EDM relationship. The specifications in the trading partner agreement should be tested before production implementation to formulate a solution to any problems revealed during testing well before reliance on the implementation.

Concerns About Future Reliability of the Public Internet

Continued monitoring of the Internet's viability as an infrastructure will take place. Increased traffic and potential lack of sufficient transmission capacity on the Internet is difficult to predict and quantify at this time. Concerns may be resolved by new Internet service providers and new communications technologies to compensate for the rapid growth of the Internet.

3. Related Standards for EDI/EDM

4. Flow Diagram

5. Specifications

A. Sending Transactions

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. HTML pages, text files, and other documents can be retrieved by a browser using HTTP, FTP, or other protocols. However Web browsers allow the user to input data to a server using HTML forms. Data is entered into the fields of the form and is transmitted to the server by pressing a pushbutton or hitting the enter key.

The HTTP protocol has two methods for transmitting a request to a server. Both methods return a response to the client, which may be a document retrieved from the server. Both methods can be used to transmit form data. The GET method is the simplest and is used for requests that pass a small amount of information. Data passed with the GET method must be translated into a special format known as "URL encoding." Furthermore, the data stream transmitted by the GET method has a limit of 1024 characters. The POST method, on the other hand, allows the upload of complete datasets without special encoding. It is this method which will be used to send GISB standard format transactions and receive the response from the server.

Using an Interactive Browser

When most of us think of Web surfing, we think of using an interractive browser. 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. For purposes of GISB standards Uniform Resource Locator (URL) is as defined by the Internet Engineering Task Force (IETF).

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 to, from, 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. When choosing a packaged browser, it is mandatory that it supports multipart encoding.

<u>Sample of HTML document with a form to perform a multipart post using an interactive browser:</u>

```
<HTML>
<HEAD>
<TITLE>GISB File Upload</TITLE>
<H1><CENTER>GISB File Upload</CENTER></H1>
</HEAD>
<HR>
<BODY>
<form ENCTYPE="multipart/form-data"
ACTION="http://www.target.server/cgi-bin/upload.exe"
METHOD=POST>
Enter Common Code Identifier 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>
```

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 “multipart/form-data” encoding type is identified as the standard encoding methodology.

- 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. “POST” has been defined as the GISB 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 for the **ACTION=** attribute.

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 packaged batch browsers use the GET method.

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 Section “Writing a Batch Browser”.

A sockets program can be written with various programming languages which offer the required library to achieve this function.

Authentication

HTTP basic authentication includes a userid and password. Interactive browsers include a basic authentication feature which automatically prompts for userid and password. In a batch browser, the authentication must be specifically coded. The userid 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.

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 transaction turnaround deadlines defined in GISB standards. This timestamp and all other pertinent file transmittal information should be logged when the posted file is stored on the receiving server as well as logged by the client . Likewise, any errors or warnings should be logged at both the server and client.

Throughput Considerations

The performance of the batch browser is one component 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.

HTTP Request Data Elements

Required Data Elements (listed in the required order)

Data Element Name	Description
from	Common Code Identifier of sending/client company
to	Common Code Identifier of receiving/server company
input-format	Descriptor of the data format within the input data set.
input-data	The properly formatted file of electronic commerce data.

Mutually Agreed Upon Data Elements

Data Element Name	Description
transaction-set	Descriptor of the transaction types included in the input-data. The values used must be from the unique 8-character names defined in the Implementation Standards. See GISB Standards for the various transaction types and their corresponding 8-character names.

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 GISB home page for sources of HTTP protocol information.

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 XYZ 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 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. In this example, the boundary is comprised of 27 hyphen characters followed by a number.

The boundary can be any character string that you choose except that it is required that it will not to occur anywhere else in the form or in the transaction being sent. 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 appear 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 requires two hyphen characters appended to the front of the string as you can note by the lines between the data fields in the example. The last boundary required in the form is two hyphen characters 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 value 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.

In this example, the data portion, or body, sent to the server program is as follows and assumes only required data elements are sent (not mutually agreed data elements):

```
----------------------------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~AAA6300300~14~1234567890000
~14~2345678900000

... 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 data element names must match the name specified by GISB. 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 line termination. 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 preceding “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 the required two hyphen characters following the boundary at the end of the form to indicate the end of the data.

Although the specifications for multipart POST include several variations on this method, the GISB 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 XYZ 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-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~AAA6300300~14~1234567890000
~14~2345678900000

... more data from the X12 file…

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

Client Specifications

Each client should be synchronized to Central Time (Central Standard / Central Daylight) available at any of the sites on a synchronized network of atomic clocks. 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. Please refer to the GISB home page for information on time synchronization.

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

B. Receiving Transactions

Using a Web Server

As was stated above, the protocol HTTP using the POST method as the means to upload a transaction is the standard. 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 GISB standard places no particular requirements on the vendor for the Web server. Most commercially available Web servers will provide the needed functionality. However, please refer to comments regarding performance under "Throughput Considerations" later in this section. While the current approach to security does not require a Secure Sockets Layer (SSL) or Secure Hyper Text Transfer Protocol (S-HTTP) capable server, one of these may be a requirement in the future. Determine 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. Some vendors call this capability Internet Server Application Programming Interface (ISAPI) while others call it Netscape Application Programming Interface (NSAPI).

The CGI Process

A CGI (or BGI) 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 GISB standard format file.

The CGI program is not concerned with the content of the GISB standard format data. In fact, the standard format file will be encrypted (see the Security section). The CGI will merely accept the standard format data and store it as a file. The CGI will use the Content-Length to determine how much data to expect in the body.

Throughput Considerations

It is critical that the Web server and the associated CGI programs perform efficiently. 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 server 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.

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 XYZ 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 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 implementation 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 environment variables.

```
-----------------------------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~AAA6300300~14~1234567890000
~14~2345678900000

... 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, which is unique in that it cannot appear anywhere in the transaction being sent (see above stream for an example).

The input stream should be in binary mode to accommodate encrypted files.

- Each data element is be preceded by the boundary with the required two hyphen characters appearing before 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.
    - In the receiving program, all tag values in the HTTP header should be evaluated in a case insensitive manner.

Finding the end of the stream using both content length and the boundary end mark (the boundary with two required hyphen characters 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.

URL/CGI Implementation Guidelines

GISB standard 4.3.12 states

"As a minimum, with a trading partner agreement, one designated site for receipt should be identified for each trading partner. That site should be identified by a specific Uniform Resource Locator (URL). This does not preclude multiple designated sites being mutually agreed to between trading partners."

This standard specifies that each company must offer at least one URL (URL is a one-to-one association with CGI) to accept EDI files. However, a maximum number of URLs per company is *not* included so that companies that wish to offer additional URLs will not be held back from doing so. Though companies are free to construct an EDI Web site with multiple "single-purpose" URLs, GISB recommends the use of one "general-purpose" URL.

Error notifications include errors that occur some time after the HTTP response is sent (such as a file decryption error) as well as errors on the X12 transactions. A general-purpose URL would handle all error notifications.

Companies that wish to offer multiple URLs must negotiate additional URLs with their trading partners. All URLs that will be required for use in the EDI process must be agreed to and defined in the Trading Partner Agreement (TPA) signed by both companies. An example of a company that would define multiple URLs in the TPA is a company that comes to agreement with its partners that all nominations-related transactions are sent to a URL offered by an out-sourcing vendor. All other transactions are sent to a URL offered on its own Web server.

A company can also offer additional URLs which have a special purpose without defining the URL in a TPA. Such additional URLs would be a way of offering additional

customer service. The trading partners would have the option of using the additional URL. An example of a company that offers a URL for additional customer service is a company that offers a URL to accept capacity release information requests with immediate turnaround while the general-purpose URL is set up to postpone all capacity release information requests until 4 p.m. that day. This company wishes to keep its primary Web server available for nominations requests while other information requests are handled on a secondary Web server.

To those companies who wish to offer multiple URLs, GISB strongly recommends that you divide URL usage along transactional grouping lines, such as nominations or capacity release. Create groupings that are likely to correlate to business functions in a company within the gas industry. Do not divide URL usage along an arbitrary internally-understood group such as region of the country. Remember that the intent of not specifying a maximum number of URLs is to allow companies the freedom to offer services, not to further complicate the EDI process.
Some companies have raised a question of offering a "default" URL. The default URL would be used when the trading partner was not able to determine the proper URL from the trading partner agreement. GISB does not recommend that any company offer a default URL. When situations arise where the TPA does not fully define the appropriate URL, the partners should communicate the situation, agree to the appropriate URL usage, and revise the TPA.

Server Specifications

The HTTP server should be synchronized to Central Time (Central Standard / Central Daylight) available at any of the sites on a synchronized network of atomic clocks. 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. Please refer to the GISB home page for references on public sites for synchronization.

The HTTP server will provide an HTTP response to the client according to GISB standards.

- All data element names of the HTTP request and response fields will be in lower case. Note that the GISB standard format file contained in the request and response may follow a different standard.

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

A field delimiter of "*" will be used in the HTTP response. Please refrain from displaying a "*" anywhere else in the response so as not to confuse programs that need to parse on this basis.

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 opening and closing HTML tags at a minimum.

The HTTP response must be no more than 2048 characters.

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

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.

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

Time Synchronization

Testing has shown that the clocks on all computer systems drift. It has also been surprising to see just how much they do. Time synchronization is required to assure that all trading partners transaction times are accurate. Time accuracy is dependent on how much a system's clock drifts, how frequently it is resynchronized and the accuracy of the source used for synchronization.

Authoritative time synchronization is now being provided by governmental agencies around the world based on a synchronized network of atomic clocks. In the United States this includes the U. S. Naval Observatory and the National Institute of Standards and Technology.

A easy way to obtain the current time is from the U. S. Naval Observatory's Web site at http://tycho.usno.navy.mil/cgi-bin/timer.pl. The output from this page can easily be edited and reformatted to set a local system's time. Commercial, shareware and public domain packages are also available to synchronize system times. Among them are NTP (which is an internet standard), internet daytime, nisttime / usnotime.

Required Data Elements (listed in the required order)

Data Element Name	Description
time-c	the time of transfer completion at the server. The format will be *yyyymmddhhmmss*.
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 Table A, “Internet EDM Standard Error Codes and Messages”.
server-id	a *domainname* or *hostname.domainname* uniquely identifying the server associated with the CGI that received and processed the file.
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:

```
<html>
time-c=19960123203618*
request-status=ok*
server-id=coolhost*
trans-id=232323897*
</html>
```

or

error, plain text format:

```
<html>
time-c=19960619082855*
request-status=EEDM106:  Invalid To Common Code Identifier
server-id=coolhost*
trans-id=234423897*
</html>
```

or

warning, plain text format:

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

or, as a more elaborate response to a successful transmittal,

HTML format (this example is for a successful transmittal):

```
<html>
<head>
<title>Upload OK</title>
</head>
<!-- time-c=19960123203618*-->_
<!-- request-status=ok* -->
<!-- server-id=coolhost* -->
<!-- trans-id=232323897*-->
<h1>Upload OK </h1><br>
<body>
<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>
```

Using a 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 GISB standards for HTTP response
- accommodation for CGI to meet GISB standards for validation and processing

C. SENDING ERROR NOTIFICATION TRANSACTIONS

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 Internet EDM 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. This error notification has the potential of occurring only after the original HTTP Response is returned with an "ok" or a warning (WEDM999 format) for the request-status value, not an error (EEDM999).

Error Notification Data Elements

The data elements for the error notification are the same as those described in Section "Sending Transactions", with the exception of the "input-format" and "input-data" elements. The file containing the data elements for error notification should not be encrypted.

Required Data Elements for Error Notification (listed in the required order)

Data Element Name	Description
from	Common Code Identifier of sending/client company, the server company which detected the error
to	Common Code Identifier of receiving/server company, the client company which sent the data set in error
input-format	"error"

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 based on some process beyond CGI such as decryption; see Table A, “Internet EDM Standard Error Codes and Messages”.
		comments	Any comments the original receiving server wishes to include.

Mutually Agreed Upon Data Elements for Error Notification

none defined at this time

Error Notification “input-data” Element Specifications:

The file containing the data elements for error notification should not be encrypted.

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. Please refrain from displaying a “*” anywhere else in the error notification so as not to confuse programs that need to parse on this basis.

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 entire error notification must be no more than 2048 characters.

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

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.

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 XYZ 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"

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

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

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

orig-from=123456789*
orig-to=234567890*
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--
```

TABLE A - Internet EDM 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 the GISB home page for information sources.

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

The string for the error or warning should appear in the following format:

Validation Code:Description;supplemental message to be defined by the issuing site up to 80 characters

Internet EDM Standard Error Codes and Messages

Validation Code	Description	Data Element	Required vs. Mutually Agreed
EEDM 100	Missing from Common Code Identifier	from	required
EEDM 101	Missing to Common Code Identifier	to	required
EEDM 102	Missing input format	input-format	required
EEDM 103	Missing data file	input-data	required
EEDM 104	Missing transaction set	transaction-set	mutually agreed
EEDM 105	Invalid from Common Code Identifier	from	required
EEDM 106	Invalid to Common Code Identifier	to	required
EEDM 107	Invalid input format	input-format	required
EEDM 108	Invalid transaction set	transaction-set	mutually agreed

Validation Code	Description	Data Element	Required vs. Mutually Agreed
EEDM 109	No parameters supplied	parameter string	required
EEDM 601	Public key invalid	file itself	required - security
EEDM 602	File not encrypted	file itself	required - security
EEDM 603	Encrypted file truncated	file itself	required - security
EEDM 604	Encrypted file not signed or signature not matched		
EEDM 699	Decryption Error		required for general decryption errors not specifically identified by PGP messages or exit codes
EEDM 999	System error		required for general system errors to indicate severe errors in processing at the receiving site
WEDM 100	Transaction set sent not mutually agreed	transaction-set	mutually agreed

Error Processing; Error codes table; Example, Sending Error Notification Transactions

SECURITY

Security Concepts

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 cannot deny ownership of the transaction if it was sent

with his/her digital signature.

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 application for securing transactions.

Understanding PGP

Pretty Good Privacy (PGP) is the name of the chosen security application. See the GISB home page for information on software packages to implement the PGP security application. PGP 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 RSA key generation algorithm should be chosen for versions of PGP which offer alternatives. 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 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.

Encryption / Digital Signature

Encryption and signatures are applied to files already translated to a GISB standard data format. (Use of internal encryption such as X12.58 encryption is outside the scope of GISB encryption standards 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 key 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.

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. PGP will utilize the appropriate key pair when

encrypting, signing, and decrypting if given the correct userID in the key ring identifying the trading partner. 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. If the ID is passed along with the decrypted file, a process could be created to verify that the company which sent the transaction corresponds to the company identified within the file, once the data has been translated.

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.

Implementers of Internet EDM sites should review and evaluate Domain Name Server (DNS) cache refresh intervals so as to ensure trading partner address changes are recognized on a timely basis. A refresh interval of 24 hours or less is common.

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. Guidelines for communicating the status of the decryption step have been developed. See Section "Sending Error Notification Transactions" and Table A, "Internet EDM Standard Error Codes and Messages".

Security Requirements

Basic Authentication

Basic authentication, also known as realm one security, has been defined as one of the security standards 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 for this security as well as procedures for changing the password, if applicable.

PGP File Encryption

File encryption of the EDI file is also selected as a security standard for transmission on the Internet. The encryption software employed is required to be compatible with

PGP 2.6 or greater (using keys generated with the RSA algorithm). Those companies who wish to conduct business across the Internet in an unsecure fashion may do so by mutual agreement.

Security Recommendations

Firewall

A firewall is one or more computers running special software which is designed to provide control of communications between two networks. Its purpose is to limit the types of services between these two networks. Often, a company's connection to the Internet is intended to provide several other services to its employees who are connected by an internal network such as a Local Area Network or Wide Area Network (LAN or WAN). Examples of these services include access to the World Wide Web, use of e-mail, use of file transfer capabilities and publishing content intended for viewing by the external world on a Web server. In addition, the internal network 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 unwelcome intruders from the Internet, there should exist a protective mechanism between the internal network and the public Internet, the firewall.

There are two general mechanisms employed by firewalls to provide this control: 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 computers. 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 potential attackers to focus on specific internal hosts.

Because firewalls are designed to deal with a broad set of security issues, which may vary at each organization, and are not specific to the use of HTTP, this guide does not attempt to provide specific implementation information. Deciding on a specific firewall architecture, organizational security policies, and choosing between numerous products may require outside resources to address these issues.

E. Data Dictionary For Internet EDM

Business Name	Definition	Format	Usage*	Condition
from**	the party sending the transaction	Common Code Identifier format up to 15 characters in length	in Request; M	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	in Request; M	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	in Request; M	"X12" or other GISB standard format indicator 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 Table A, "Internet EDM Standard Error Codes and Messages"	in Response; M	"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	*domainname* or *hostname.domainname*; no embedded spaces allowed	in Response; M	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*	in Response; M	displayed in the HTTP response and posted back for any decryption-related errors

to**	the party the transaction was sent to	Common Code Identifier format	in Request; M	used in file transmittal and displayed in HTTP response and posted back for any decryption-related errors
transaction-set	name of the document type being sent	8 character code; examples are: G811TSIN, G820PYRM, G860PDAL, G811IMBL, G865ALLC, etc.; please refer to GISB Implementation Standards	in Request; MA	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	in Response; M	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 Mandatory (M) or Mutually-Agreed-To (MA).

** Common Code Identifier

Index of Examples

6. Scenarios

CHECKLIST OF TESTING STEPS

Purpose

Preliminary steps in testing are helpful before the full batch browser and server applications are completed. This checklist is intended to provide a series of small achievements leading up to the complete solution.

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 compliant servers offering interactive upload for test. See the GISB home page for a list of organizations willing to act as testing partners. These organizations should have a URL complete with the CGI program name to which a tester may send test files. File content does not need to be X12 or other GISB standard format to accomplish this step in testing.

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 One security.

3. Add Realm One 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 and private key pair. Make sure to choose the RSA key generation algorithm. 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.

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. As an optional preliminary step, 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 GISB standards. Test transfers to your CGI using your batch browser.

4. Transfer a X12 or other GISB standard format dataset to your server and process it through your translator or other appropriate processes.

5. Copy the CGI to a "secure" directory where Realm One security, or basic authentication, is enabled. Using your batch browser, transfer to both URLs, with and without authentication. Thoroughly test using the incorrect userid and

password against the secure directory.

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 trading partner 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 trading partners and/or several clients within your own company transfer concurrently to ensure that your server can withstand the load.

10. Test application with various simulated errors in both file transfers and in PGP decryption.

7. APPENDIX A - Reference Guide

CGI

An excellent source on CGI is a book entitled “Special Edition Using CGI” by Jeffrey Dwight and Michael Erwin.

Firewall Security

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.

GISB

GISB Web Site: (http://www.gisb.org) Primary reference for natural gas industry standards

General GISB FTTF Reference Page: (http://www.gisb.org/fttf.htm). This location provides pointers to samples and further documentation.

HTTP

As of this printing (July 1998), there are two versions of HTTP (1.0 and 1.1) that are recognized as standards. The GISB EDM architecture is based on HTTP 1.0, and all implementations should be compatible with this version. All of the HTTP functions required by GISB EDM are expected to be fully compatible with HTTP 1.1 servers and should work without changes.

W3C WorldWide 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

Syntax information for multipart can be found here:

http://unix1.sncc.lsu.edu/internet/guides/www-docs/WWW/Protocols/rfc1341/7_2_Multipart.html

HTML

Before April 24, 1998, the recommended standard from the WorldWide Web Consortium was HTML 3.2. The specification for this standard can be found at:
http://www.w3.org/pub/WWW/TR/REC-html32.html

Effective April 24, 1998, the WorldWide Web Consortium has made a recommendation for HTML 4.0. Information on HTML 4.0 may be found at http://www.w3.org/TR/REC-html40/.

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.

PGP Software

Network Associates (http://www.nai.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)

Time Synchronization

Further information on time synchronization may be found at the following Web sites:

http://www.eecis.udel.edu/~mills/ntp/test.html

http://tycho.usno.navy.mil/ntp.html

http://www.ccd.bnl.gov/xntp

http://www.txdirect.net/users/sfisher/clock.html

http://www.is.co.za/resources/ftpsite/tucows/softsync.html