CHANGE LOG: Tab 2, Introduction

Section/ Paragraph	Area to Change/Comment	Suggested Change
Header	Replace WGQ with REQ	REQ for WGQ
17	Greater gas industry	Replaced with greater gas and electric industries
[all]	Most, but not all, references to WGQ	Replaced most instances of WGQ with REQ.
17	Marketplace for natural gas	Replaced by matketplace for energy
[all]	Most references to Internet EDI/EDM and BATCH FF	Replaced with Electronic Delivery Mechanism (EDM)
18	TAB 7, TAB 8, TAB 9 and TAB 10	Replaced with TAB 7 Testing Guidelines and TAB 8 Appendix
18	Appendix C and Appendix D	Eliminated, and renamed Appendix E to Appendix C

CHANGE LOG: Tab 3, Executive Summary

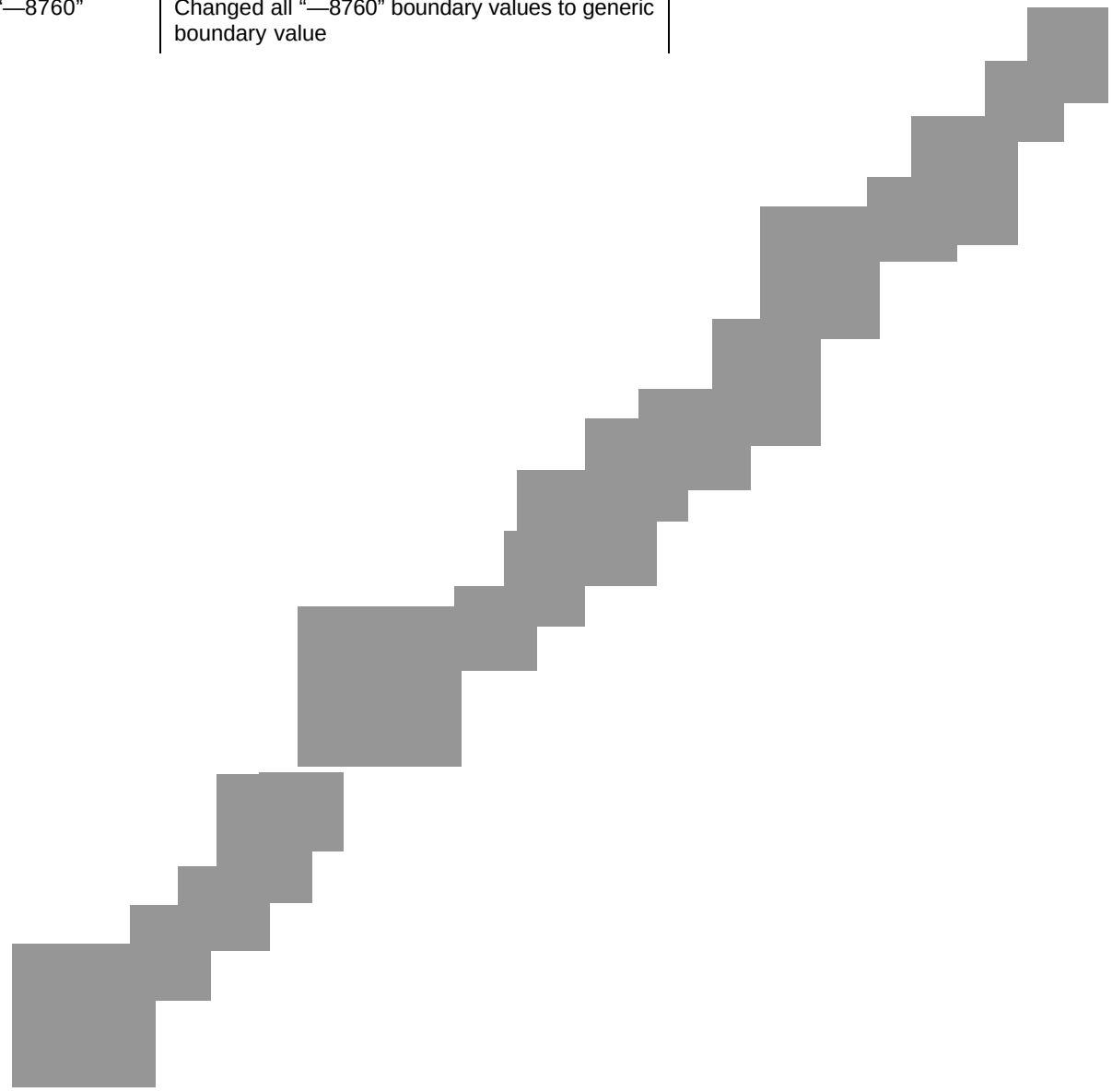
Section/ Paragraph	Area to Change/Comment	Suggested Change
Header	Replace WGQ with REQ	REQ for WGQ
19	Added sentence introducing the concept of NAESB, as should be present in all Exec Summaries.	Selected sentence off NAESB website explaining NAESB.
[all]	Most, but not all, references to WGQ	Replaced most instances of WGQ with REQ.
[all]	Numerous references to Batch FF/EDM	Removed references to Batch FF/EDM, following lead of Tab 6.
19,20	Spelled out all acronyms the first time they appeared, as should be the case with an Exec Summary	NAESB, all 4 quadrants, EDM, EDI, FTTF
19	Identified who benefits from standards	Added Gas and Electric utilities, changed the word 'bank' to 'financial institution.
20	More precise wording on who will require consensus	Added 'international community'.
20	Identify testing partners	Changed Gas to Electric
21	Concerns about reliability	Updated wording to more accurately describe the current state of affairs with the Internet

CHANGE LOG: Tab 6

Section/ Paragraph	Area to Change/Comment	Suggested Change
[all]	Numerous references to WGQ.	Replaced WGQ with REQ
[all]	Numerous references to Batch FF/EDM	Removed references to Batch FF/EDM
Page 51	Extraneous "to"	Removed
Page 52	Common Code Identifier format	Tagged as OPEN ISSUE
[all]	Numerous references to gisb-acknowledgment-receipt	Tagged as OPEN ISSUE
Page 52	Description of input-format data element incorrect for REQ use	Removed reference to FF, added XML
[all]	Numerous references to CDI/script	Removed references to CGI/script
Page 52	request-status	Removed reference to decryption process

Page 53	time-c data element lacks time-zone indicator	Added time-zone indicator
Page 53	Transaction-set data element requires enhancement to remove gas industry specific references	Changed transaction-set to 16 character free form text field
Page 54	Diagram	Added EDM Server and simplified
Page 58	Reference to pipeline under Throughput Considerations	Removed pipeline reference
Page 58	HTTP Request Data Elements	Brief description added
Page 59	Incorrect description of transaction-set for REQ purposes	Provided new description for transaction-set, and tagged 8-character names as an OPEN ISSUE
Page 59	Writing a Batch Browser	Removed reference to NAESB home page
Page 59	Description of content type line	Rephrased to indicate that this referred to the specific example on page 59
Page 60	Description of content length	Rephrased to indicate that this referred to the specific example earlier on page 59
[all]	Several references to version 1.4	Changed to 1.6
Page 65	Reference to multipart POST	Tagged implementation as an OPEN ISSUE
Page 67	References to Central time zone	Removed restriction to use Central time
Page 67	Synchronization of client	Added reference to clock accessible via the Internet
Page 67	References to gas nominations	Removed references to gas nominations
Page 67	Brief description of HTTP Request	Moved the description to page 58
Page 68	Using a Web Server	Added reference to the POST method as the only supported HTTP method
Page 68	Timestamp time-c references lack timezone indicator	Added identifier for time zone
Page 68	Text incorrectly indicates that SSL or S-HHTP are optional	Removed references to SSL being optional and all references to S-HTTP
Page 68	The CGI Process	Changed to The Receive Process
Page 68	Text specific to pipelines and gas industry	Replaced with more generic text
Page 69	Reference to C program	Removed
Page 69	Reference to BGI programming	Removed
Page 69	Writing the CGI Process	Changed to Developing the Receive Process
Page 69	Reference to GET operation	Removed
[all]	References to CGI process	Changed to receiving program
Page 69	Reference to standard input	Changed to input stream
Page 71	Reference to standard output	Changed to sending program
Page 71	URL/CGI Implementation Guidelines	Changed to Receive Process URL Implementation Guidelines
Page 71	References to WGQ standard 4.3.12	Tagged as OPEN ISSUE
Page 71	A separate URL for nominations	Replaced by a separate URL may be used for customized processing
Page 72	Text specific to gas industry regarding capacity release	Replaced with more generic text
Page 72	References to Central Time zone	Removed
Page 72	References to nomination deadline	Removed

Page 74	time-c data element has no time zone indicator	Added time zone indicator and provided description of its use
Numerous Examples	examples of "gisb-acknowledgement-receipt" containing time-c missing time zone indicator	Updated all examples of time-c to include time zone indicator
HTTP Response, Request		Capitalized all references to HTTP Response, Request
"—8760"	Changed all "—8760" boundary values to generic boundary value	



INTRODUCTION

The North American Energy Standards Board (NAESB) is a voluntary non-profit organization comprised of members from all aspects of the greater gas and electric industries. NAESB Internet Electronic Transport (ET) Standards are a product of the North American Energy Standards Board. The NAESB mission is to take the lead in developing and implementing standards across the industry to simplify and expand electronic communication, and to streamline business practices. This will lead to a seamless North American marketplace for energy, as recognized by its customers, the business community, industry participants and regulatory bodies.

The standards are written as 'minimums,' which industry participants are encouraged to exceed through provision of value-added services and customized arrangements. NAESB defines 'exceed the minimum standard' to mean surpassing the standards without negative impact on contracting and non-contracting parties.

All of the standards have been adopted in the realization that as the industry evolves and uses the standards, additional and amended NAESB standards will be necessary. Any industry participant seeking additional or amended standards (including principles, definitions, standards, data elements, process descriptions, technical implementation instructions) should submit a request detailing the change to the NAESB office so that the appropriate process may take place to amend the standards.

TAB 1 Version Notes

Contains notes about this version, and, if appropriate, a brief summary of changes from the immediately preceding version.

TAB 2 Introduction

Provides a background statement about NAESB's Mission and the underlying concepts behind the design and use of this guide.

TAB 3 Executive Summary

Provides a brief outline of the industry business situation which is the basis for development of this guide.

TAB 4 Business Process & Practices

Provides a brief overview of the business process and the NAESB approved principles, definitions and standards related to the business process covered by this guide.

TAB 5 Related Standards

Provides a reference to any related standards.

TAB 6 Technical Implementation - Electronic Transport (ET)

Provides an overview of the business process for ET.

Data Dictionary

Provides definition of the standard data elements and the usage requirements for each element.

Batch Flow Diagram

Sending Electronic Packages

Provides instructions to develop mechanisms for sending of NAESB standard format data files.

Receiving Electronic Packages

Provides instructions to develop mechanisms for receiving of NAESB standard format data files.

Security

Provides guidelines for data privacy, data integrity, authentication and non-repudiation of inbound and outbound packages.

Other Considerations

Provides information regarding error notification and testing. Includes a reference guide and examples for repudiation and validation.

TAB 7 Testing Guidelines

TAB 8 Appendix

Appendix A - Reference Guide
Appendix B - Repudiation and Validation Examples
Appendix C - Minimum Technical Characteristics for an EDM Server

EXECUTIVE SUMMARY

The North American Energy Standards Board (NAESB) Wholesale Gas Quadrant (WGQ), Retail Electric Quadrant (REQ), and Retail Gas Quadrant (RGQ) have developed standards for electronic commerce over the Internet. The Internet Electronic Transport (ET) standards enable the rapid, reliable, and safe transportation of electronic information between NAESB trading partners.

This document is a high-level guide to implementing various technologies necessary to communicate transactions and other electronic data using standard protocols. As such, this guide is not intended to be a comprehensive, in-depth manual. Where possible, this guide points to more in-depth material. The Reference section provides locations on the Internet to obtain more information as well as recommended books and periodicals. ??should we delete this section??

Overview of Electronic Transport Life Cycle

In the Internet ET life-cycle, the party sending data, the "Sender", creates an electronic package by encrypting the data payload and applying appropriate header 'envelope' information such as 'to' and 'from'. This electronic package is submitted to the trading partner's SSL Web server as an HTTP Request using the POST method.

The receiving party, the "Receiver", receives and decrypts the package, then forwards the payload data to back-office processes. A receipt is sent from the Receiver to the Sender with timestamps and any error notices. The Receiver back-office systems process the data according to NAESB Quadrant-specific Electronic Delivery Mechanisms (QEDM). If the Receiver decrypts in a separate process, the Receiver may send an Error Notification package to the Sender to identify errors found during decryption.

Trading partners take turns being the Sender and Receiver depending on what information and data needs to be exchanged.

The Electronic Transport standards do not care what the contents of the electronic package are. Each business process may define different contents, and the ET is designed to work with any type of contents (e.g. EDI, flat files, etc).

Open Standards

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

- Communications Protocols
- Sending of Secure Electronic Packages
- Receipt of Secure Electronic Packages
- Security
- IETF EDIINT AS2 – "HTTP Transport for Secure EDI"

The open standard technologies selected by NAESB WGQ/REQ/RGQ to address these areas are designed to provide flexibility and scalability. There are business benefits gained from adherence to "HTTP Transport for Secure EDI" such as:

- Allows potential to more readily, electronically trade with others (e.g., utilities, financial institutions, suppliers, retail customers)
- Makes it more likely that Commercial Off-The-Shelf (COTS) software packages can be purchased to replace custom written applications currently in place to support legacy GISB/NAESB EDM
- Strengthens the surety of receipt and error notification

??AS2 GISB/UCC profiles; AS2 has lost GISB references; divergence regarding PKI/PGP??; HTTP Transport for Secure EDI (AS2)??AS2 profiles?? is an emerging standard, largely based on the original NAESB WGQ EDM, that is being developed by the Internet Engineering Task Force, the Internet standards body. Adherence with a formal, international Internet standard, such as AS2 ensures that the specification will not change without due process and any changes that do occur will be the result of a broad consensus in the international community. Individual companies and entire industries are free to use as much or as little of AS2 as they see fit, providing the maximum flexibility to meet business needs. 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 is covered at a high level in the Business Process and Practices (Major functions of Internet EDM covered by the Standards) section and in more detail in later sections.

Same Internet ET Application Implementation For All Trading Partners

The Internet ET application is not platform-specific. An organization's Internet ET application serves the role of communicating with all trading partners in the energy industry no matter what hardware, operating system and programming languages their trading partners use. For this reason, testing with other trading partners on a variety of platforms is very important in ensuring that each Internet ET application is compatible with a range of platforms used by various trading partners.

Further Information

Please see the NAESB home page at http://www.naesb.org/ for additional useful information on the implementation of Internet ET.

Key Assumptions

This document makes the following assumptions:

Contents of the Electronic Package Do Not Matter

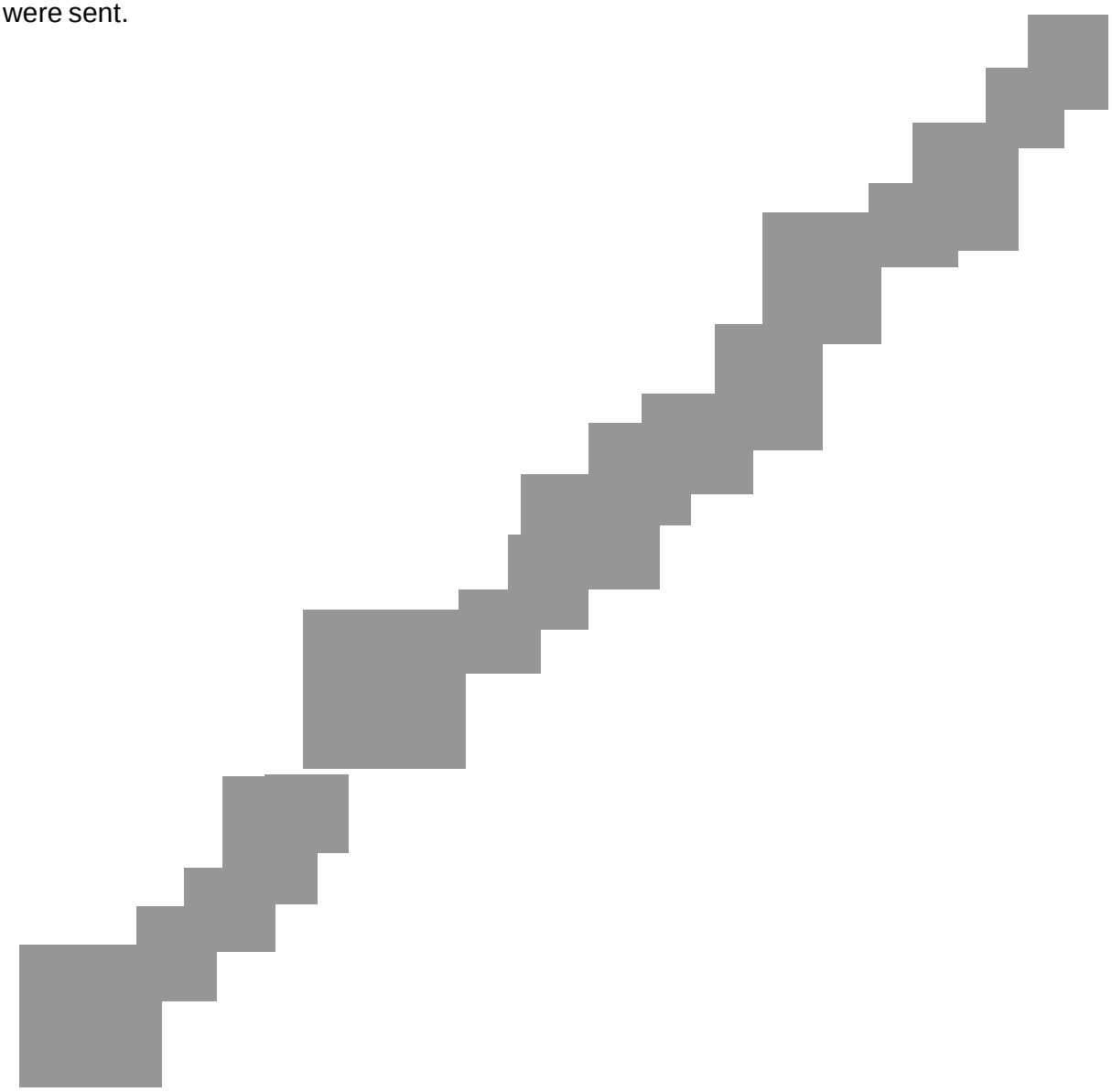
The Internet ET is does not care what type of data, ie the payload of the electronic package, is being sent. The Internet ET's main function is to get the package from point X to point Y reliably with privacy, authentication, integrity, and non-repudiation.

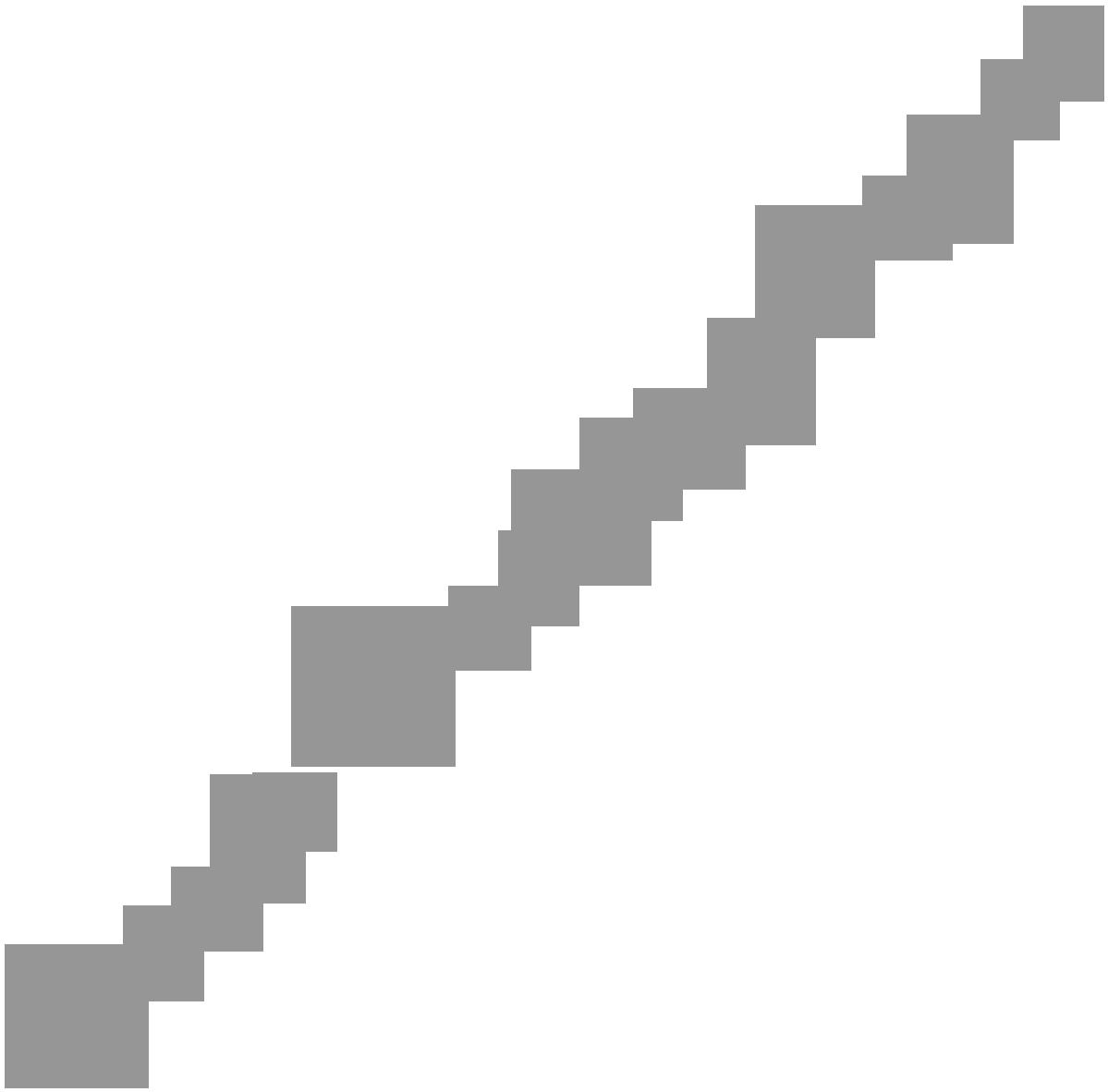
Importance of the Trading Partner Agreement (TPA)

The expectations of who will perform what function and how it will be accomplished in Internet ET should be laid out in the trading partner agreement. ??review TPA; should we add a section that details elements of the TPA that are relevant to the ET??.

Testing With Energy Industry Internet ET Participants

The Internet ET requires basic connectivity testing between trading partners. Testing should assure that the package was received, was decrypted properly, and that appropriate receipts were sent.





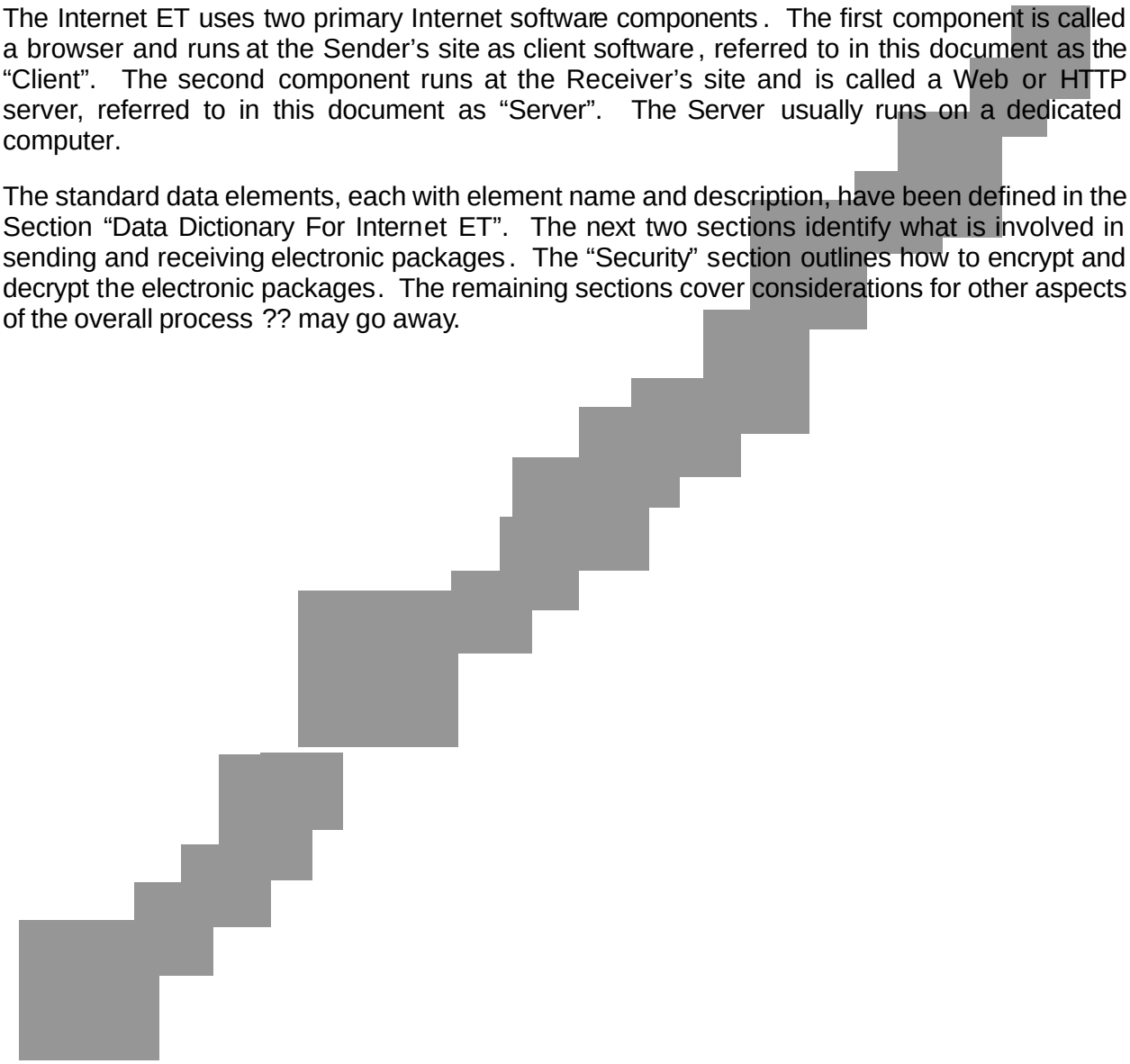
TAB 6, TECHNICAL IMPLEMENTATION - INTERNET ET

Technologies Selected by NAESB WGQ/REQ/RGQ Internet ET

The NAESB Internet ET uses TCP/IP and HTTP to securely and reliably transport electronic packages to trading partners..

The Internet ET uses two primary Internet software components. The first component is called a browser and runs at the Sender's site as client software, referred to in this document as the "Client". The second component runs at the Receiver's site and is called a Web or HTTP server, referred to in this document as "Server". The Server usually runs on a dedicated computer.

The standard data elements, each with element name and description, have been defined in the Section "Data Dictionary For Internet ET". The next two sections identify what is involved in sending and receiving electronic packages. The "Security" section outlines how to encrypt and decrypt the electronic packages. The remaining sections cover considerations for other aspects of the overall process ?? may go away.



DATA DICTIONARY FOR INTERNET ET??REVIEW IN DETAIL

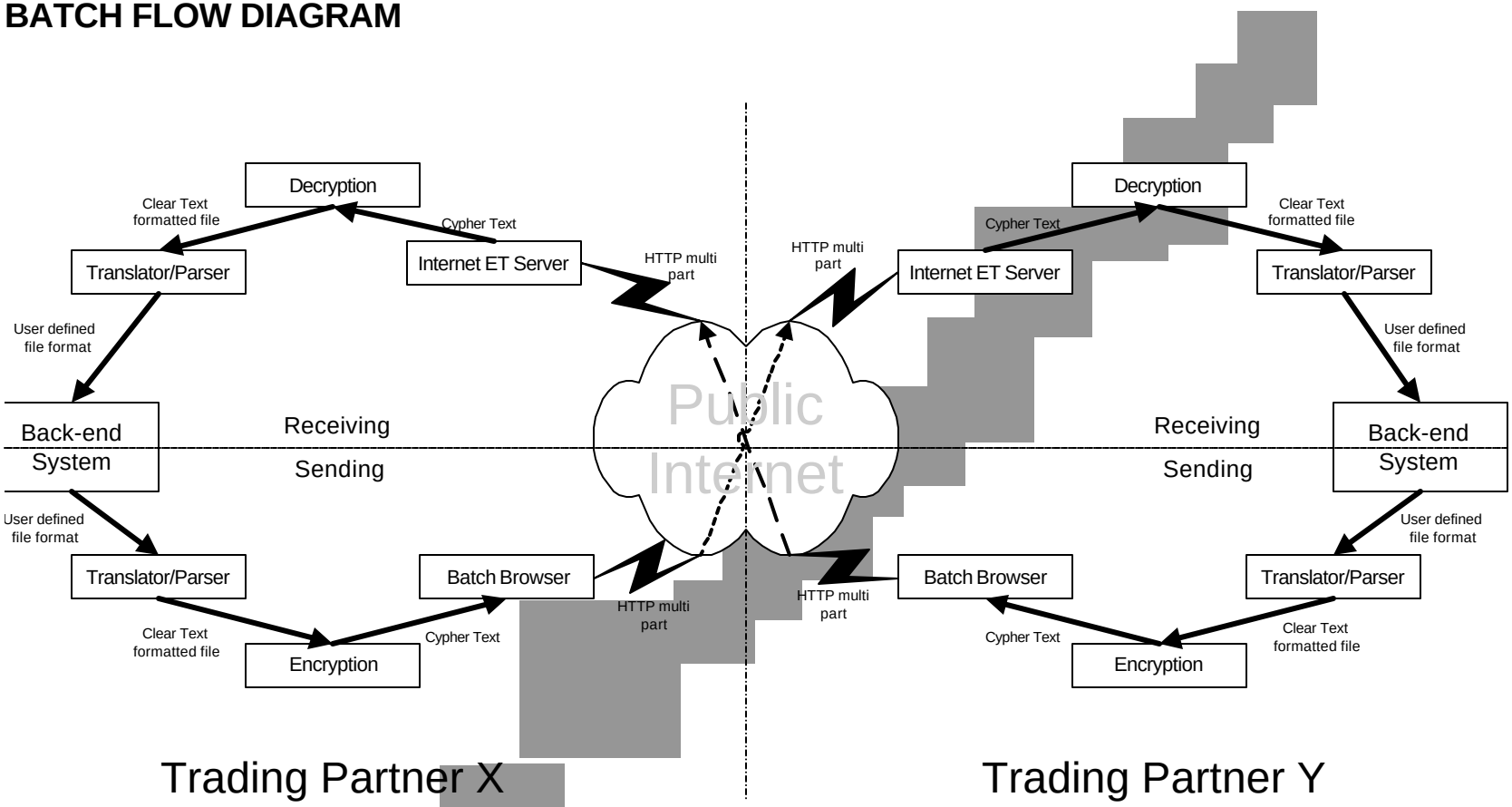
Business Name	Definition	Format	Usage*	Condition
from**	the party sending the transaction	Common Code Identifier format (OPEN ISSUE 001)	in Request; M	used in file transmittal; displayed in HTTP Response; and, used in posting back decryption-related errors
input-data	the filename for the transaction data set transmitted	including drive letter and directory name with filename if needed	in Request; M	used in file transmittal of any transaction data sets; 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 ;XML;error	in Request; M	"X12","XML", or other NAESB REQ standard format indicator used in file transmittal; "error" used in posting back any decryption-related errors
receipt-disposition-to	the party to receive receipts, the value should be the same as the "from"	Common Code Identifier format (OPEN ISSUE 001)	in Request; M	used in file transmittal and in posting error notifications
receipt-report-type	type of receipt type being requested by sender	gisb-acknowledgement-receipt (OPEN ISSUE 002)	in Request; M	used in file transmittal and in posting error notifications
receipt-security-selection	used to request signed receipts	signed-receipt-protocol=required,pgp-signature;signed-receipt-micalg=required,md5	in Request; MA	used in file transmittal and in posting error notifications
refnum	used by the party to assign a unique message identifier for tracing purposes	maximum 40 character integer value	in Request; MA	May be used by sender to send tracking information to a recipient. Use of this data element is by mutually agreed. This data element is conceptually similar to a Message-ID filed within RFC 822.
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 processing; error messages/warnings and their related descriptions are returned if problems were encountered in processing.
server-id	uniquely identifies the Server 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

Business Name	Definition	Format	Usage*	Condition
time-c	the time file transfer is complete at the Server, where + or –ZZ indicates delta from UTC (ref ISO 8601)	yyyymmddhhmmss-ZZ; yyyymmddhhmmss+ZZ (OPEN ISSUE 010)	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 (OPEN ISSUE 001)	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; (OPEN ISSUE 003) refer to NAESB REQ Implementation Guide, Related Standards Tab, Hypertext Transfer Protocol (HTTP) section, HTTP transaction-set Code Values table.	in Request; MA	used in file transmittal
trans-id	sequential number assigned to the transaction by the Server 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
version	the NAESB REQ EDM version being used by the sender	numeric, decimal notation (e.g. 1.6)	in Request; M	used in file transmittal and in posting error notifications

*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 (OPEN ISSUE 001)

BATCH FLOW DIAGRAM



Batch Flow Diagram

Electronic Transport Life Cycle

The life cycle of an Electronic Package using Internet ET is described below:

Sender		Receiver
• Collects payload data to be sent • Encrypts payload • Prepares digital signature if necessary		
• Uses browser to create Electronic Package multipart HTTP Request with header data elements and payload. • Uses HTTP POST to send the electronic package to the Receiver	↘	
		• Receives the HTTP Request on their Web/HTTP Server • Validates Sender information from HTTP Request Header data elements and payload • **Decrypts payload file • Prepares Receipt • Checks digital signatures if necessary
	↙	• Sends Receipt with either “OK” or error message
• Updates logs • If errors, correct errors then repeat process		• **Decrypts payload file
	↙	• If errors in decryption, sends Error Notification to Sender
• Receives Error Notification • Updates logs • correct errors then repeat process		• Updates logs • If no errors, Receiver processes contents of payload

**Parties may choose to decrypt file before or after Receipt is sent to Sender.

SENDING TRANSACTIONS

General Flow

The following is an example of the steps necessary to send an Internet ET package:

1. Open HTTP connection (OPEN ISSUE 004)
2. Check connection status. If in error, re-queue package according to Internet ET standards. This check should be performed here and throughout the following processes.
3. Post, including Authentication, Send multipart form, Receive HTTP Response data
4. Check connection status. If in error re-queue package according to Internet ET standards
5. Check HTTP status code (200 is good, less than 300 may be acceptable). If status is not successful re-queue package according to Internet ET standards
6. Close connection - wait for other end to close in a reasonable time
7. Parse HTTP Response data elements
8. If request-status ok, then log success
9. If request-status error, then log error
10. If no valid request-status re-queue package according to Internet ET standards
11. Remove package from sending queue when successful or when failed completely

If trading partners agree to implement signed receipts, then the sending party must include the "receipt-security-selection" data element in the posted data. The receiving party must digitally sign the "gisb-acknowledgement-receipt" (OPEN ISSUE 002) and encapsulate the "gisb-acknowledgement-receipt" (OPEN ISSUE 002) and digital signature body parts within a MIME envelope with a "content-type" of application/pgp-signature.

HTTP Post

Internet ET uses HTTP POST to send electronic packages and responses. The POST method allows the upload of complete datasets without special encoding.

Using an Interactive Browser for Internet ET

Electronic packages can be uploaded to a trading partner using an interactive browser secured using SSL 128-bit encryption. Sending electronic packages via an interactive browser is ideal for a small volume of package transfers, or as a back-up method to any batch or automated process.

To use an interactive browser to upload data, an HTML document must be created with an HTML <FORM> element that allows the Sender to type in any necessary data elements, such as "to", "from", "input-format", and the name of the file to be uploaded. When the user submits the form, an HTTP POST is sent to the Server with the package, which includes the uploaded file and the required data elements.

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 below, could be used with any browser that supports multipart POST with a file upload. .

EXAMPLE: HTML DOCUMENT WITH A FORM FOR MULTIPART POST USING AN INTERACTIVE BROWSER:

```
<HTML>
<HEAD>
<TITLE>NAESB Internet ET Package Upload</TITLE>
<H1><CENTER>NAESB Internet ET Package 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 (OPEN ISSUE 001)
From: <INPUT TYPE="text" NAME="from" SIZE=20 VALUE=""><br>
To: <INPUT TYPE="text" NAME="to" SIZE=20 VALUE=""><br>
NAESB Internet ET Version: <INPUT TYPE="text" NAME="version" SIZE=5 VALUE="1.6"><br>
Deliver Receipt To: <INPUT TYPE="text" NAME="report-disposition-to" SIZE=20 VALUE=""><br>
Receipt Type: <INPUT TYPE="text" NAME="receipt-report-type" SIZE=30
VALUE="gisb-acknowledgement-receipt"><br>(OPEN ISSUE 002)

IF requesting signed receipts also include: Receipt Type: <INPUT TYPE="text" NAME="receipt-security-selection"
SIZE=30 VALUE="signed-receipt-protocol=required, pgp-signature; signed-receipt-micalg=required, md5"><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 (TPA) (OPEN ISSUE 008) identifies the URLs for both parties.
- METHOD= specifies the HTTP protocol method. "POST" has been defined as the Internet ET standard method.
- <INPUT ...> HTML INPUT elements include the required data elements such as "from", "to", and "input-format". Refer to the data dictionary for the complete list of required data elements.

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 interactive browser will take the values entered in the input fields and reformat them according to the encoding type into a data stream. The file identified for upload, is opened and its contents are included in the data stream. The data stream is then sent to the URL specified by "**ACTION=**", which indicates a Server Receiving script or program written to receive the package.

Using a Batch Browser for Internet ET

A batch browser is used by companies that want to automate their transport processes and/or

prefer to minimize human involvement. This browser needs to be capable of program-based or script-based initiation.

A batch browser can be created using custom programming. A batch browser will be coded to perform the same formatting that the interactive browser performed to send a data stream which conforms to the HTTP and Internet ET protocols. A batch browser must be coded as a "TCP sockets" program. See Section ??check?? "Writing a Batch Browser".

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 base64-encoded within the document header. Base64-encoding utilities are readily available on the Internet as either public domain software or commercial libraries.

??do we need this because of SSL 128?? GPD will research?? NO, what do we do now?

Server Response

The Server will send a "gisb-acknowledgement-receipt" (OPEN ISSUE 002) as an HTTP Response to the Client before dropping the Client's connection. If the transacting parties agree to use signed receipts, then the Server applies a digital signature to the "gisb-acknowledgement-receipt" (OPEN ISSUE 002) and encapsulates the entire package in a MIME envelope of "content-type: application/pgp-signature".

The "gisb-acknowledgement-receipt" returned from the Server contains the "time-c" receipt timestamp that is recorded when the final byte from the package upload is received and stored. This receipt timestamp is the official timestamp regarding transaction turnaround deadlines as defined in Internet ET and QEDM standards. This timestamp and all other pertinent package transmittal information should be logged by the receiver when the posted package is stored on the Server, and logged by the Client. Likewise, any errors or warnings should be logged at both the Client and Server.

HTTP Request Data Elements

The HTTP Request will provide all required data elements in the ORDER DEFINED BELOW. Any "mutually-agreed-upon" data elements will follow the required data elements in the data stream. Refer to the section "Data Dictionary for Internet ET" for descriptions of these data elements.

Required Data Elements, Listed in the Required Order:

1. from
2. to
3. version
4. receipt-disposition-to
5. receipt-report-type
6. input-format
7. input-data

Mutually Agreed Upon Data Elements

- transaction-set
- receipt-security-selection

??check deleted descriptions against data dictionary

Writing a Batch Browser

A batch browser Client needs to simulate the actions of an interactive browser Client. As stated earlier, the interactive browser Client will take the HTML form, reformat the information according to the HTTP protocol, then send the data stream to the 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.

EXAMPLE: A TYPICAL HEADER SENT TO THE SERVER

```
POST /cgi-bin/AS2dispatcher HTTP/1.1
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
```

POST Line - . In the example above, the first line indicates the POST method was used and identifies which Receiving program to call:

```
POST /cgi-bin/AS2dispatcher HTTP/1.1
```

Content Type - The "content-type" line indicates that the encoding method is multipart, and identifies the character string used as the boundary.

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

Boundary String - The "boundary=" identifies the string that 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. The string used CANNOT OCCUR ANYWHERE ELSE IN THE PACKAGE 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. . The LAST boundary required in the form is TWO HYPHEN CHARACTERS APPENDED TO THE BACK of the separator boundary, used to indicate to the Server program that this is the end of the data.

Content Length - 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. It tells the Server how much is going to come after this point. In the example above, the content length is:

Content-Length: 5379

Envelope / Required Data Elements – The envelope information for the package ("to", "from", etc) is included in a series of boundaries that include the content-disposition and "name=" qualifiers, followed by the data element value. The example below includes the "from" field as "123456789" and the "to" field as "234567890".

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 Internet ET Data Dictionary. 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 blank line. 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 example. In most programming libraries and commercial products the starting delimiter is "\r\n\r\n" (C notation).

The example below includes the "from" field as "123456789" and the "to" field as "234567890".

------------------------------87453838942833
content-disposition: form-data; name="from"

123456789 (OPEN ISSUE 001)
------------------------------87453838942833
content-disposition: form-data; name="to"

234567890 (OPEN ISSUE 001)
------------------------------87453838942833
content-disposition: form-data; name="version"

1.64
------------------------------87453838942833
content-disposition: form-data; name="receipt-disposition-to"

123456789 (OPEN ISSUE 001)
------------------------------87453838942833
content-disposition: form-data; name="receipt-report-type"

```
gisb-acknowledgement-receipt (OPEN ISSUE 002)
------------------------------87453838942833
content-disposition: form-data; name="input-format"

x12
```

Payload – The important electronic data (EDI, etc) that is being packaged and sent is encrypted and included in its own boundary section.

The data field containing the Internet ET payload file has two extra identifiers. The "filename=" element indicates the name of the file sent from the sender/client computer. In the example the name of the file is **"c:\temp\smallnom.bin"**.

```
content-disposition: form-data; name="input-data"; filename="c:\temp\smallnom.bin"
```

The "content-type" element indicates the type of the data being transmitted according to accepted Internet standards.

```
content-type: multipart/encrypted; boundary=--boundary2--200309090001; protocol="application/pgp-encrypted"
```

Note that encrypted files can be multipart also, which means they will have their own boundary string.

```
------------------------------87453838942833
content-disposition: form-data; name="input-data"; filename="c:\temp\smallnom.bin"
content-type: multipart/encrypted; boundary=--boundary2--200309090001; protocol="application/pgp-encrypted"

----boundary2--200309090001
content-type: application/pgp-encrypted

Version: 1

----boundary2--200309090001
content-type: application/octet-stream

-----BEGIN PGP MESSAGE-----
Version: PGP 6.5

hQCMAzRG1pEOIOvdAQP+JMr0m/9+8yOL60Z9Vr6fFV81FCExB/o0xmwiMkiwYsHsz0e8sb7ErC340MrNA/dw3taGMjmI
+CXYRF/PLEdg1NZE1ZCtNeL4YdIHAMLWwODGlQxhSucz8rMSgQ5mZzcOJwBdWLW70efgsu/9UljuJjYc1uZ6C03eFQv/4
3fkB+alATtgydxX4g8QK664ad+Jo/XUICSmWBL66fqJR1KLeLf4wTaqGy174Aq48Wpwvg1Eh785zC03UAw0qg0ugMt86dPe
yd91e2JigqwDYEf/DYEKD0J9BGiGpS/uAupNKj8Ocp2IWClxKOGUbxpVNOnNTqWHS/GntegvDE/7/ewCxDxsnmQS95pOl1
41QZ1RqbeNaqx2Dq/ra9g65HNchOCzjul5Vi8HHf6Yhg2WnROe+npByyCue6rihqgNVOJwj0Cvzpb4JE+gMDf3q4ISUb1Fv7
/+SSFHDdnhdC5YTpqf1Bc3B07hiLmtTXqNit31EbX9UVElObzSa9ZhxbC6/eSl7Nuf5ZTDsh9nrk+QQJ6FeC9W4cqXLj7IZyS
aRO8Vtff+4ktqeuhYusT4kSpnk027aw4O/5jomUkfb22CAe4=
=Oiuo
-----END PGP MESSAGE-----
----boundary2--200309090001--
```

Boundary String Terminators - Each multipart stream must be terminated with the boundary string terminator. After the contents of the last data field, the boundary string and the required two-hyphen terminator indicate the end of the multipart encrypted payload:

```
----boundary2--200309090001--
```

A second boundary terminator string indicates the end of the package:

```
------------------------------87453838942833--
```

See section "??" for an outline of "content-type" values.

HTTP Features Not Supported by Internet ET

Internet ET DOES NOT SUPPORT:

- multiple files in a single POST
- a single file split into multiple POSTs

EXAMPLE: AN X12 EDI DATA STREAM BEFORE ENCRYPTION:

Content-type: application/EDI-X12

ISA~00~ ~01~AAA6300300~14~1234567890000 ~14~2345678900000
... more data from the X12 file…
IEA~1~000003616

EXAMPLE: THE SAME X12 EDI ENCRYPTED WITH PGP

content-type: multipart/encrypted; boundary=--boundary2--200309090001; protocol="application/pgp-encrypted"
----boundary2--200309090001
content-type: application/pgp-encrypted
Version: 1
----boundary2--200309090001
content-type: application/octet-stream
-----BEGIN PGP MESSAGE-----
Version: PGP 6.5
hQCMAzRG1pEOIOvdAQP+JMr0m/9+8yOL60Z9Vr6fFV81FCExB/o0xmwiMkiwYsHsz0e8sb7Er340MrNA/dw3taGMjmI+
CXYRF/PLEdg1NZE1ZCtNeL4YdIHAMLWwODGlQxhSucz8rMSgQ5mZzcOJwBdWLW70efgsu/9UljuJjYc1uZ6C03eFQv/43f
kB+alATtgydxX4g8QK664ad+Jo/XUICSmWBL66fqJR1KLeLf4wTaqGy174Aq48Wpwvg1Eh785zC03UAw0qg0ugMt86dPeyd
91e2JigqwDYEf/DYEKD0J9BGiGpS/uAupNKj8Ocp2IWClxKOGUbxpVNOnNTqWHS/GntegvDE/7/ewCxDxsnmQS95pOl141
QZ1RQbeNaqx2Dq/ra9g65HNchOCzjul5Vi8HHf6Yhg2WnROe+npByyCue6rihqgNVOJwj0cVzpb4JE+gMDf3q4ISUb1Fv7/
+SSFHDdnhdC5YTpqf1Bc3B07hiLmtTXqNit31EbX9.UVElObzSa9ZhxbC6/eSl7Nuf5ZTDsh9nrk+QQJ6FeC9W4cqXLj7IZyS
aRO8Vtff+4ktqeuhYusT4kSpnk027aw4O/5jomUkfb22CAe4=
=Oiuo
-----END PGP MESSAGE-----
----boundary2--200309090001--

EXAMPLE: A COMPLETE ELECTRONIC PACKAGE DATA STREAM

```
POST /cgi-bin/AS2dispatcher HTTP/1.1
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  (OPEN ISSUE 001)
-----------------------------87453838942833
content-disposition: form-data; name="to"

234567890  (OPEN ISSUE 001)
-----------------------------87453838942833
content-disposition: form-data; name="version"

1.46
-----------------------------87453838942833
content-disposition: form-data; name="receipt-disposition-to"

123456789  (OPEN ISSUE 001)
-----------------------------87453838942833
content-disposition: form-data; name="receipt-report-type"

gisb-acknowledgement-receipt (OPEN ISSUE 002)
-----------------------------87453838942833
content-disposition: form-data; name="input-format"

X12
-----------------------------87453838942833
content-disposition: form-data; name="input-data"; filename="c:\temp\smallnom.bin"
content-type: multipart/encrypted; boundary=--boundary2--200309090001; protocol="application/pgp-encrypted"

----boundary2--200309090001
content-type: application/pgp-encrypted

Version: 1

----boundary2--200309090001
content-type: application/octet-stream

-----BEGIN PGP MESSAGE-----
Version: PGP 6.5

hQCMAzRG1pEOIOvdAQP+JMr0m/9+8yOL60Z9Vr6fFV81FCExB/o0xmwiMkiwYsHsz0e8sb7ErC340MrNA/dw3taGMjmI
+CXYRF/PLEdg1NZE1ZCtNeL4YdIHAMLWwODGlQxhSucz8rMSgQ5mZzcOJwBdWLW70efgsu/9UljuJjYc1uZ6C03eFQv/4
3fkB+alATtgydxX4g8QK664ad+Jo/XUICSmWBL66fqJR1KLeLf4wTaqGy174Aq48Wpwvg1Eh785zC03UAw0qg0ugMt86dPe
yd91e2JigqwDYEf/DYEKD0J9BGiGpS/uAupNKj8Ocp2IWClxKOGUbxpVNOnNTqWHS/GntegvDE/7/ewCxDxsnmQS95pOl1
41QZ1RqbeNaqx2Dq/ra9g65HNchOCzjul5Vi8HHf6Yhg2WnROe+npByyCue6rihqgNVOJwj0cVzpb4JE+gMDf3q4ISUb1Fv7
/+SSFHDdnhdC5YTpqf1Bc3B07hiLmtTXqNit31EbX9UVElObzSa9ZhxbC6/eSl7Nuf5ZTDsh9nrk+QQJ6FeC9W4cqXLj7IZyS
aRO8Vtff+4ktqeuhYusT4kSpnk027aw4O/5jomUkfb22CAe4=
=Oiuo
-----END PGP MESSAGE-----
----boundary2--200309090001--
-----------------------------87453838942833--
```

RECEIVING TRANSACTIONS

General Flow

The following is an example of the steps necessary to receive an Internet ET package:

1. Parse multi-part form
2. Validate HTTP Request data elements
3. If HTTP Request data elements in error, return appropriate Internet ET standard error code in the HTTP Response data elements
4. Save data
5. Create "gisb-acknowledgement-receipt" (OPEN ISSUE 002)
6. If using signed receipts, Produce a digital signature over ??over what?? the "gisb-acknowledgement-receipt" (OPEN ISSUE 002) created in step 5.
7. Encapsulate the "gisb-acknowledgement-receipt" (OPEN ISSUE 002) and digital signature body parts in a "Content-Type" of "multipart/signed envelope"
8. Return HTTP Response with the "gisb-acknowledgement-receipt" (OPEN ISSUE 002) object back to Client
9. Close connection
10. Log final results
11. Route data file to the next process based upon "input-format"

Overview of Web Server Receiving Programs

The Web Server receives the POST and calls the appropriate Receiving script or program to:

- parse the incoming HTTP Request
- create the receipt timestamp using the date, time and time zone indicator
- create an HTML Response to the Client.

An Internet ET Receiving program may be implemented using a variety of technologies and techniques, including . Active Server Pages (ASP), Common Gateway Interface (CGI), Java Server Pages (JSP), Java Servlets, and Personal Home Pages (PHP).

The Internet ET is supported by most commercially available Web/HTTP servers.

The Receiving Program and Process

The Receiving 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 Internet ET payload file.

The Receiving program only stores the payload file and is not concerned with the content of the payload file, which is encrypted. It will use the "content-length" to determine how much data to expect in the body of the package.

A Receiving process requires an executable program or module that is called by the Server when it is identified by a POST operation.

When the Server receives a POST it will first read the header and populate environment variables before calling the Receiving program. Most HTTP servers read header variables and

populate environment variables. Check your HTTP server documentation for more information.

EXAMPLE: A SAMPLE HTTP POST HEADER

```
POST /cgi-bin/AS2dispatcher HTTP/1.1
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
```

After reading the HTTP header information, the Server will buffer the remaining data transmitted and call the Receiving program specified in the POST statement. Do not assume that the Receiving program is called as soon as the header is read, which can impact your receipt timestamp. The more common implementations buffer the entire transmission before calling the program. Check your server implementation if this characteristic is important to you.

The Receiving program will have the following data stream available, and will have most of the header data available in environment variables.

EXAMPLE: DATA STREAM AVAILABLE TO RECEIVING PROGRAM

```
------------------------------87453838942833
content-disposition: form-data; name="from"
123456789  (OPEN ISSUE 001)
------------------------------87453838942833
content-disposition: form-data; name="to"
234567890  (OPEN ISSUE 001)
------------------------------87453838942833
content-disposition: form-data; name="version"
1.64
------------------------------87453838942833
content-disposition: form-data; name="receipt-disposition-to"
123456789  (OPEN ISSUE 001)
------------------------------87453838942833
content-disposition: form-data; name="receipt-report-type"
gisb-acknowledgement-receipt (OPEN ISSUE 002)
------------------------------87453838942833
content-disposition: form-data; name="input-format"
X12
------------------------------87453838942833
content-disposition: form-data; name="input-data"; filename="c:\temp\smallnom.bin"
content-type: multipart/encrypted; boundary=--boundary2--200309090001; protocol="application/pgp-encrypted"
----boundary2--200309090001
content-type: application/pgp-encrypted
Version: 1
----boundary2--200309090001
content-type: application/octet-stream
-----BEGIN PGP MESSAGE-----
Version: PGP 6.5
hQCMAzRG1pEOIOvdAQP+JMr0m/9+8yOL60Z9Vr6fFV81FCExB/o0xmwiMkiwYsHsz0e8sb7ErC340MrNA/dw3taGMjmI
+CXYRF/PLEdg1NZE1ZCtNeL4YdIHAMLWwODGlQxhSucz8rMSgQ5mZzcOJwBdWLW70efgsu/9UljuJjYc1uZ6C03eFQv/4
3fkB+alATtgydxX4g8QK664ad+Jo/XUICSmWBL66fqJR1KLeLf4wTaqGy174Aq48Wpwvg1Eh785zC03UAw0qg0ugMt86dPe
yd91e2JigqwDYEf/DYEKD0J9BGiGpS/uAupNKj8Ocp2IWClxKOGUbxpVNOnNTqWHS/GntegvDE/7/ewCxDxsnmQS95pOl1
41QZ1RQbeN.aqx2Dq/ra9g65HNchOCzjul5Vi8HHf6Yhg2WnROe+npByyCue6rihqgNVOJwj0cVzpb4JE+gMDf3q4ISUb1Fv
7/+SSFHDdnhdC5YTpqf1Bc3B07hiLmtTXqNit31EbX9UVElObzSa9ZhxbC6/eSl7Nuf5ZTDsh9nrk+QQJ6FeC9W4cqXLj7IZy
SaRO8Vtff+4ktqeuhYusT4kSpnk027aw4O/5jomUkfb22CAe4=
=Oiuo
-----END PGP MESSAGE-----
```

----boundary2--200309090001--
----------87453838942833--

This Receiving program should check for basic validity in the environment variables and the data stream, then . parse the variables/data from the format. Data validations should include:

- The "REQUEST_METHOD" environment variable is "POST"
- The "CONTENT_TYPE" environment variable should be "multipart/form-data" and the boundary, which cannot appear anywhere in the transaction being sent
- The input stream should support binary mode to accommodate encrypted files
- Each data element should 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 a blank line ("\r\n\r\n" in C+ notation) before the start of the data
- All tag values in the HTTP header should be evaluated in a case insensitive manner
- Improperly formatted input. Finding the end of the stream using both "content-length" and the boundary string terminator end mark is a good method to detect improperly formatted input.

Acknowledgement Receipt: "gisb-acknowledgement-receipt"

The Acknowledgement Receipt is critical to non-repudiation and business process timing. Immediately after the Receiving program validates, parses, and saves the data, the Receiving program should record the time and construct a "gisb-acknowledgement-receipt" (OPEN ISSUE 002) described ??below or in section ??.

If using signed receipts, the Receiving program must also produce a digital signature of the "gisb-acknowledgement-receipt" (OPEN ISSUE 002) and send both the "gisb-acknowledgement-receipt" (OPEN ISSUE 002) and the digital signature body parts within a multipart/signed MIME envelope.

This receipt is sent from the Receiving program to the Client prior to closing the HTTP connection.

Additional Receiving Program Functions

- All data element names of the HTTP Request and Response fields will be in lower case. Note that the Internet ET standard format file contained in the Request and Response may follow a different standard. ??clean up examples; references to these??
- Carriage returns and line feeds will be ignored in all files.
- ??really?? 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 HTTP Response and in the order specified. Additional information can be included after the required elements at the server's discretion.
- The "gisb-acknowledgement-receipt" (OPEN ISSUE 002) must be enveloped in a "multipart/report", as specified in EDIINT AS2 (OPEN ISSUE 005) following the rules for Generalized Receipts.
- If signed receipts are used, the "gisb-acknowledgement-receipt" (OPEN ISSUE 002)

including the multipart/report envelope is digitally signed, producing an application/pgp-encrypted body part. Both the multipart/report ("gisb-acknowledgement-receipt" (OPEN ISSUE 002)) and the "application/pgp-signature" body parts are placed in a multipart/signed envelope and the entire package is returned to the sender.

- 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.

Receiving Process URL Implementation Guidelines

??got some work here?? Internet ET standard 4.3.12 (OPEN ISSUE 007) 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." (OPEN ISSUE 008)*

Each company must offer at least one URL to accept files using Internet ET. Companies can offer multiple URLs. Though companies are free to construct a Web site with multiple "single-purpose" URLs (e.g. nominations.xyzcorp.com; enrollments.xyzcorp.com) NAESB recommends the use of one "general-purpose" URL.

The Receiving program may initiate error notifications after the "gisb-acknowledgement-receipt" (OPEN ISSUE 002) is sent (e.g. file decryption errors). Error notifications posted to the Sender would be directed to the Sender's general-purpose URL.

All URLs that will be required for use in the Internet ET process must be agreed to and defined in the Trading Partner Agreement (TPA) (OPEN ISSUE 008). . .

HTTP Response "gisb-acknowledgement-receipt" Data Elements

Required Data Elements listed in the required order:

- time-c
- request-status
- server-id
- trans-id

Refer to your QEDM for additional required HTTP Response data elements (e.g. "time-c-qualifier" in REQ).

Examples of HTTP Response Required Data Elements:

EXAMPLE: RESPONSE, SUCCESSFUL, MULTIPART FORMAT:

content-type: multipart/report; report-type="gisb-acknowledgement-receipt"; boundary="GISB7867" (OPEN ISSUE 002)

--GISB7867
Content-type: text/html

<HTML><HEAD><TITLE>Acknowledgement Receipt Success</TITLE></HEAD> <BODY><P>
time-c=19960619082855 (OPEN ISSUE 010)*
*request-status=ok**
*server-id=coolhost**

*trans-id=234423897**
</P> </BODY></HTML>
--GISB7867
Content-type: text/plain

time-c=19960619082855 (OPEN ISSUE 010)*
*request-status=ok**
*server-id=coolhost**
*trans-id=234423897**
--GISB7867--

EXAMPLE: RESPONSE, SUCCESSFUL, MULTIPART FORMAT, TIME-QUALIFER FOR TIME ZONE:

content-type: multipart/report; report-type="gisb-acknowledgement-receipt"; boundary="GISB7867" (OPEN ISSUE 002)

--GISB7867
Content-type: text/html

<HTML><HEAD><TITLE>Acknowledgement Receipt Success</TITLE></HEAD> <BODY><P>
time-c=19960619082855 (OPEN ISSUE 010)*
*request-status=ok**
*server-id=coolhost**
*trans-id=234423897**
time-c-qualifer=-0400
</P> </BODY></HTML>
--GISB7867
Content-type: text/plain

time-c=19960619082855 (OPEN ISSUE 010)*
*request-status=ok**
*server-id=coolhost**
*trans-id=234423897**
time-c-qualifer=-0400
--GISB7867--

EXAMPLE: RESPONSE, ERROR, MULTIPART FORMAT:

content-type: multipart/report; report-type="gisb-acknowledgement-receipt"; boundary="GISB7866" (OPEN ISSUE 002)

--GISB7866
Content-type: text/html

<HTML><HEAD><TITLE>Acknowledgement Receipt Error</TITLE></HEAD> <BODY><P>
time-c=19960619082855 (OPEN ISSUE 010)*
*request-status=EEDM106: Invalid To Common Code Identifier**
*server-id=coolhost**
*trans-id=234423897**
</P> </BODY></HTML>
--GISB7866
Content-type: text/plain

time-c=19960619082855 (OPEN ISSUE 010)*
*request-status=EEDM106: Invalid To Common Code Identifier**
*server-id=coolhost**
*trans-id=234423897**
--GISB7866--

EXAMPLE: RESPONSE, WARNING, MULTIPART FORMAT:

content-type: multipart/report; report-type="gisb-acknowledgement-receipt"; boundary="GISB7866" (OPEN ISSUE 002)

--GISB7866
Content-type: text/html

<HTML><HEAD><TITLE>Acknowledgement Receipt Warning</TITLE></HEAD> <BODY><P>
time-c=19960619082855 (OPEN ISSUE 010)*
*request-status=WEDM100: Transaction Set Sent, Not Mutually Agreed**
*server-id=coolhost**
*trans-id=234423897**
</P> </BODY></HTML>
--GISB7866
Content-type: text/plain

time-c=19960619082855 (OPEN ISSUE 010)*
*request-status= WEDM100: Transaction Set Sent, Not Mutually Agreed **
*server-id=coolhost**
*trans-id=234423897**
--GISB7866--

EXAMPLE: RESPONSE, SUCCESSFUL , SIGNED RECEIPT:

content-type:multipart/signed; micalg=pgp-md5; protocol="application/pgp-signature";
boundary=--boundary2--200309090001

----boundary2--200309090001

content-type: multipart/report; report-type="gisb-acknowledgement-receipt"; boundary="GISBB7867" (OPEN ISSUE 002)

--GISB7867
Content-type: text/html

<HTML><HEAD><TITLE>Acknowledgement Receipt Success</TITLE></HEAD> <BODY><P>

time-c=19960619082855 (OPEN ISSUE 010)*
*request-status=ok**
*server-id=coolhost**
*trans-id=234423897**

</P> </BODY></HTML>

--GISB7867
Content-type: text/plain.
time-c=19960619082855 (OPEN ISSUE 010)*
*request-status=ok**
*server-id=coolhost**
*trans-id=234423897**
--GISB7867--
----boundary2--200309090001
content-type: application/pgp-signature

-----BEGIN PGP MESSAGE-----

Version: 2.6.26.5

iQCVAwUBMJrRF2N9oWBghPDJAQE9UQQAtl7LuRVndBjrk4EqYBIb3h5QXIX/LC//JV5bNvkZIGPIcEmI5iFd9boEgvpirHt
IREEqLQRkYNoBActFBZmh9GC3C041WGquMbrbxc+nIs1TIKlA08rVi9ig/2Yh7LFrK5Ein57U/W72vgSxLhe/zhdfolT9BrnH
OxEa44b+EI=
=ndaj

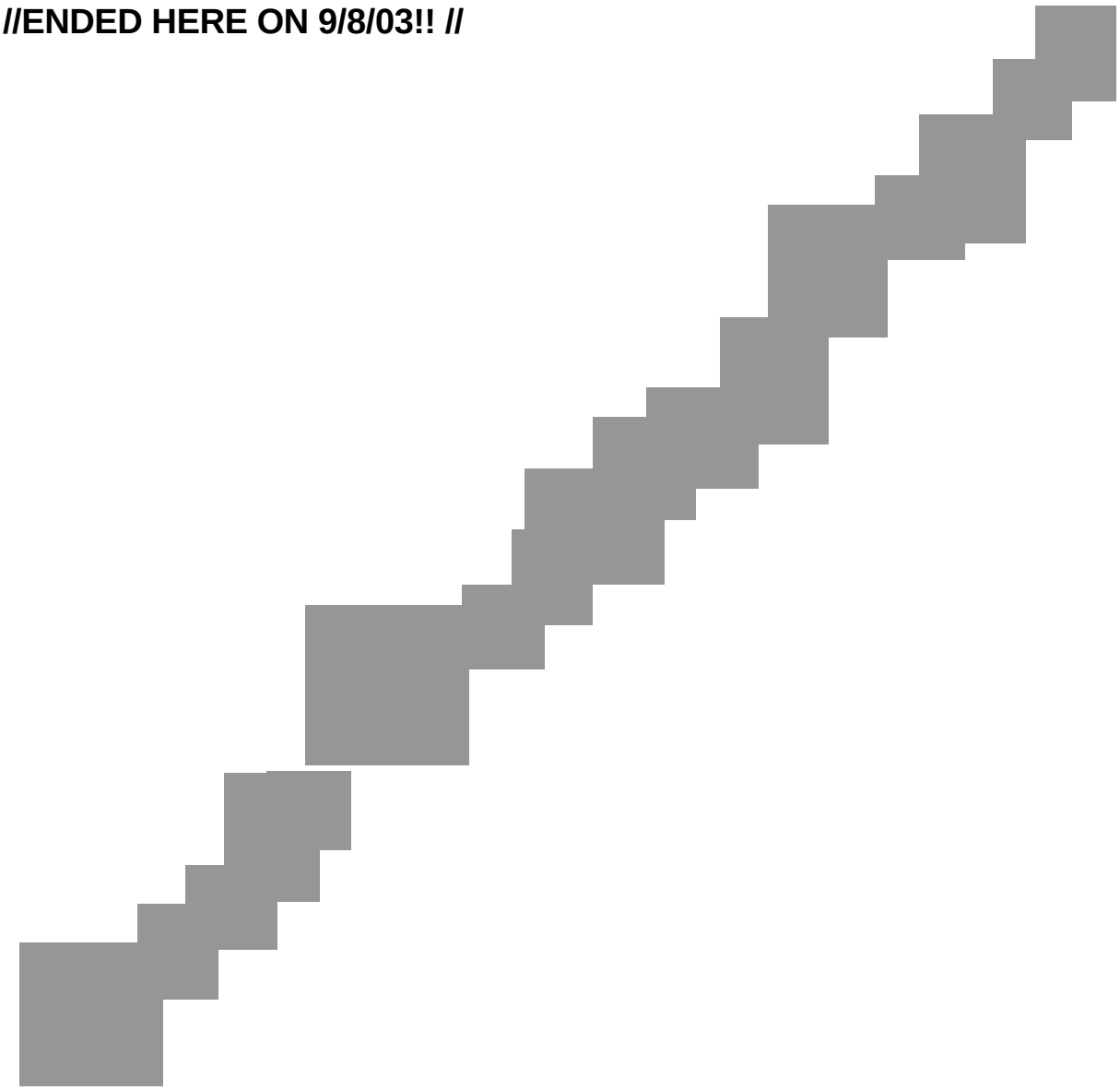
-----END PGP MESSAGE-----

----boundary2--200309090001—

EXAMPLE: RESPONSE, SUCCESSFUL:

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> (OPEN ISSUE 010)*

//ENDED HERE ON 9/8/03!! //



SECURITY

Security Concepts

The security requirements include four primary security aspects: data Privacy, data Integrity, Authentication, and Non-repudiation (PAIN).

- Data privacy: unauthorized parties cannot decipher the content of the data.
- Authentication: the receiver is certain of the identity of the sender.
- Data integrity: unauthorized parties cannot modify or corrupt the data.
- 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 and OpenPGP security application for securing transactions.

Understanding PGP and OpenPGP

Pretty Good Privacy (PGP) is the name of the chosen security application. OpenPGP is the Internet Engineering Task Force standard version of PGP which excludes all patented algorithms, allowing free commercial use of the standard. See the NAESB Web site for information on software packages to implement the PGP or OpenPGP security application. Both OpenPGP and PGP use 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. Implementers of OpenPGP should choose DSA and El Gamal when creating their key pair. The public keys will be distributed using a secure method (e.g., courier mail) to the company's trading partners.

You must use the utmost care in protecting your private key. If 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.

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 NAESB REQ standard data format, and before the data is sent to the batch browser. (Use of internal encryption such as X12.58 encryption is outside the scope of NAESB REQ encryption standards but does not conflict with PGP.)

Encryption and signatures can be accomplished manually for each file using the on-line PGP or,

on a mutually-agreed-upon basis, OpenPGP 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.

Digital signatures may also be applied, on a mutually-agreed-upon basis, to the HTTP Response by the Receiver of the original package.

Decryption / Signature Verification

After a transaction is received and processed by the Receiving program, it is ready to be decrypted and have its signature verified. PGP and OpenPGP software use 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 and OpenPGP software will return a human-readable descriptive text. (OPEN ISSUE 011)

It is recommended that all implementers create a process where the descriptive text is used to look up the ID of the trading partner in a database table. If the ID is passed along with the decrypted file, a process could be created to verify that the trading partner which sent the transaction corresponds to the trading partner identified within the file, once the data has been translated. (OPEN ISSUE 011)

When digital signatures are applied, on a mutually-agreed-upon basis, the HTTP Response received by the sender of the transaction may be verified to ensure non-repudiation of receipt of the transaction.

Throughput Considerations

Encryption, digital signing, decryption and signature verification are all very CPU intensive. Companies anticipating large volumes of Internet ET traffic should research state-of-the-art techniques for scalability, including but not limited to:

- separating decryption and signature verification processing from web server receiving and processing;
- passing . . secured or to-be-secured packages to a separate computer for security processing.
- optimizing CPU and memory on security processing computers .
- real-time or near real-time monitoring of website performance

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 TPA must identify the userid and password for this security as well as procedures for changing the password, if applicable. (OPEN ISSUE 008 and OPEN ISSUE 012)

PGP or OpenPGP 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 6.5 or greater (using keys generated with the RSA algorithm) or the OpenPGP standard, specified in IETF RFC 2440, on a mutually-agreed-upon basis. There are freely available software implementations of the OpenPGP standard available at http://www.gnupg.org/.

General 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 the 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.

CLIENT AND SERVER SPECIFICATIONS

Each Client and Server should be synchronized to a clock in the network of atomic clocks that is

accessible via the Internet. The Client and Server should be synchronized as many times per day as necessary to ensure synchronization with an atomic clock. Please refer to Appendix A, "Time Synchronization" ??check this appendix?? for references on public sites for synchronization.

HTTP Servers should be configured to use one of the allowable TCP ports listed in Appendix E (OPEN ISSUE 009).

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. Criteria for selecting an outsourced Internet ET service provider should consider their ability and experience with Internet ET standards for HTTP Request and Response validation and processing.

SENDING ERROR NOTIFICATIONS

Error Notification

When a Client sends an electronic package to a Server, the Server responds with a receipt. . Further back-office processing (e.g. decryption) may be required, and additional errors may be found.

Transport errors found by the Receiver after the initial receipt is sent to the Sender are communicated using the Error Notification transaction. . . .

Errors from translation and other back-office processing are outside the scope of the Internet ET.

When a file passes the decryption step, no notifying communication is sent back to the Client. However, if the decryption step fails, an error notification must be sent to the Client.

The Error Notification format applies to the posting of an error message after the Sender's session has been disconnected. This error notification is used only if the original HTTP Response is returned with an "ok" or a warning (WEDM999 format) for the request-status value, not an error (EEDM999).

Additionally, trading partners are permitted to use digitally signed error notifications, if both parties mutually agree to do so.

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 (OPEN ISSUE 001)

to	Common Code Identifier of receiving/server company, the client company which sent the data set in error (OPEN ISSUE 001)
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 the receiving process 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 first occurrence of the field name within the response will contain the value.
- An error notification contains two body parts nested within a multipart/report outer envelope with the content-type of "gisb-error-notification". (OPEN ISSUE 014)

- The first body part contains human readable content in HTML. The second body part contains machine readable content in plain text. Additionally, consenting trading partners can mutually agree to digitally sign error notifications.

- If digital signatures are used, the multipart/report containing the Error Notification is used to create a digital signature body part, identified by a "content-type" of application/pgp-signature. Both the multipart/report Error Notification and application/pgp-encrypted digital signature body parts are combined in a multipart/signed envelope.

EXAMPLE: ERROR NOTIFICATION ELECTRONIC PACKAGE:

```
POST /cgi-bin/AS2dispatcher HTTP/1.1
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  (OPEN ISSUE 001)
------------------------------87453838942833
content-disposition: form-data; name="to"

123456789  (OPEN ISSUE 001)
------------------------------87453838942833
content-disposition: form-data; name="version"

1.6
------------------------------87453838942833
content-disposition: form-data; name="receipt-disposition-to"

123456789  (OPEN ISSUE 001)
------------------------------87453838942833
content-disposition: form-data; name="receipt-report-type"

gisb-acknowledgement-receipt   (OPEN ISSUE 002)
------------------------------87453838942833
content-disposition: form-data; name="input-format"

error
------------------------------87453838942833
content-disposition: form-data; name="input-data"; filename="c:\temp\error.not"
content-type: multipart/report; report-type="gisb-error-notification"; boundary="GISB7868"  (OPEN ISSUE 014)

--GISB7868
Content-type: text/html

<HTML><HEAD><TITLE>Error Notification</TITLE></HEAD> <BODY><P>
orig-from=123456789*  (OPEN ISSUE 001)
orig-to=234567890*  (OPEN ISSUE 001)
orig-input-format=X12*
resp-time-c=19960619102855*  (OPEN ISSUE 010)
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*
</P> </BODY></HTML>
```

--GISB7868
content-type: text/plain

orig-from=123456789 (OPEN ISSUE 001)*
orig-to=234567890 (OPEN ISSUE 001)*
*orig-input-format=X12**
resp-time-c=19960619102855 (OPEN ISSUE 010)*
*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**
--GISB7868--
----------87453838942833—

Signed Error Notification

content-type:multipart/signed; micalg=pgp-md5; protocol="application/pgp-signature";
boundary=--boundary2--200309090001

----boundary2--200309090001

content-type: multipart/report; report-type="gisb-error-notification"; boundary="GISB7868" (OPEN ISSUE 014)

--GISB7868
Content-type: text/html

<HTML><HEAD><TITLE>Error Notification</TITLE></HEAD> <BODY><P>
orig-from=123456789 (OPEN ISSUE 001)*
orig-to=234567890 (OPEN ISSUE 001)*
*orig-input-format=X12**
resp-time-c=19960619102855 (OPEN ISSUE 010)*
*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**

</P> </BODY></HTML>

--GISB7868
content-type: text/plain

orig-from=123456789 (OPEN ISSUE 001)*
orig-to=234567890 (OPEN ISSUE 001)*
*orig-input-format=X12**
resp-time-c=19960619102855 (OPEN ISSUE 010)*
*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**

--GISB7868--
----boundary2--200309090001

content-type: application/pgp-signature
-----BEGIN PGP MESSAGE-----

Version: 6.5

iQCVAwUBMJrRF2N9oWBghPDJAQE9UQQAtl7LuRVndBjrk4EqYBIb3h5QXIX/LC//JV5bNvkZIGPIcEmI5iFd9boEgvpirHt
IREEqLQRkYNoBActFBZmh9GC3C041WGquMbrbxc+nIs1TIKlA08rVi9ig/2Yh7LFrK5Ein57U/W72vgSxLhe/zhdfolT9BrnH
OxEa44b+EI=
=ndaj

-----END PGP MESSAGE-----

----boundary2--200309090001--

Pre-validation before Decryption

Proper trapping of the range of decryption process errors listed in Table A (Internet EDM Standard Error Messages and Codes) may require program code which is external to the decryption algorithm. Some versions of the PGP software do not explicitly discriminate between EEDM601, EEDM602, EEDM603, and EEDM699 type errors. Under such a circumstance, files inbound to the decryption process should be preprocessed to trap the errors not identified by the PGP version being used. For example, searching the file for the text strings “BEGIN PGP MESSAGE” and “END PGP MESSAGE” can quickly identify “EEDM602 File not encrypted” and “EEDM603 Encrypted file truncated” type errors when the implemented PGP version only identifies decryption success, invalid public key (EEDM601), and decryption failure (EEDM699).

TAB 7 - TESTING

NAESB ELECTRONIC TRANSPORT (ET) TEST PLAN

Implementation of Internet ET requires testing to assure all parties are prepared to operate according to the Internet ET. This document focuses on testing standards for establishing Electronic Transport (ET) connectivity with a trading partner. Testing for transaction and other quadrant-specific testing standards can be found in each quadrant's QEDM.

ET Connectivity testing standards include:

- Connectivity test scripts – These scripts define the steps needed to adequately test connectivity.
- Testing Connectivity Worksheet – This worksheet defines key operations parameters for a trading partner. The parameters include ET URL's, contacts and other information.
- Testing Signoff Worksheet – This worksheet defines critical operations environment characteristics of the trading partner. The characteristics include testing environments.

GENERAL TESTING ASSUMPTIONS

The following assumptions apply to ET testing:

- This Test Plan covers ET testing. Transactions and business process test plans can be found in the QEDM.
- Testing may uncover problems. Problems found during testing should be expected.
- This Test Plan is a basic demonstration of competency, and may not uncover all problems that may eventually require correction.
- The primary goals of this test are to establish connectivity including encryption, to test transfer of a large file, and to exercise the exchange failure process.
- This Test Plan assumes that automated processes will be used when testing. Any solutions that involve manual interaction or data manipulation are documented in advance in the Testing Signoff Worksheet, and are communicated to testing partners at the beginning of the testing cycle.
- Testing is done on dedicated test systems. Production systems should not be used for testing.
- ET connectivity testing should last no longer than two weeks once begun.

TESTING GOALS

The specific testing goals of this ET Test Plan are:

- Establish ET connectivity between CRs and TDSPs, including Internet connections and encryption compatibility.
- Validate that normal production transaction files can be sent.
- Validate that ET timestamps are being delivered.
- Validate that protocol failures are handled properly.
- Validate that exchange failures are handled properly.
- Validate that encryption/decryption and digital signature failures are handled properly.

TEST EXECUTION

Scripts and Frames

The Test Scripts defined in this document detail each step of the testing process for Connectivity. There may be several test scripts for a defined business process scenario. For example, a number of scenarios are tested for both positive (accept) and negative (reject) results.

Each Test Script involves an exchange (Request and Response) of data between trading partners. Each step in the process is referred to as a 'Frame'.

Testing Scripts

Each TP will confirm receipt of test file exchange via normal ET standards and via e-mail.

- Each TP should confirm that received files were not corrupted.
- Each TP shall exercise fail-over mechanisms by simulating a protocol failure and an exchange failure, triggering the appropriate notices to the identified Market Participant contacts.
- Each TP shall exercise encryption failure processes by simulate an encryption/decryption failure, triggering the appropriate notices to the identified Market Participant contacts.
- Each TP shall send a formal notice of successful certification completion to their TP

[??more detailed scripts?]

Recommended Internal Tests

In addition to tests executed with trading partners, the following tests are recommended as internal tests of ET systems.

- Stress Test – Ability to send and receive large production files (e.g. 10MB minimum uncompressed).
- Fail-over test – Test any processes triggered by a protocol or exchange failure by your

trading partner.

Testing Responsibilities

The 'Testing Responsibilities' section details the responsibilities each party has in the testing process. This Test Plan is focused on testing connectivity. Each party has certain obligations prior to, during and after testing.

Prior to Testing

- Parties should provide daily and emergency contact information for the test lead, and the test lead alternate.
- Complete applications for certification, licensing, etc. required for the target marketplace.
- Implement a dedicated test system.
- Provide TSW and TCW to target trading partner.

During Testing

- Participate in scheduled testing conference calls with your trading partners.
- Adhere to the established test schedule.
- Execute test frames as defined.

After Testing

- Communicate formally success or failure of testing with trading partner.

CHECKLIST OF TESTING STEPS (OPEN ISSUE 015)

[??from original EDM??]

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.

- Install an interactive browser. Identify an existing Web server from among NAESB WGQ compliant servers offering interactive upload for test. See the NAESB WGQ 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 NAESB WGQ standard format to

accomplish this step in testing.

- 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.
- Add Realm One security to your file transfer, and change the URL to the secure URL. Continue transfer tests with your batch browser.
- Acquire and install PGP or OpenPGP compliant software. Generate your public and private key pair. Make sure to choose the RSA key generation algorithm for PGP or DSA and El Gamal for OpenPGP. 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.

Server and CGI

- 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.
- 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.
- Acquire or develop a CGI program to receive file transfers and process according to NAESB WGQ standards. Test transfers to your CGI using your batch browser.
- Transfer a X12 or other NAESB WGQ standard format dataset to your server and process it through your translator or other appropriate processes.
- 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.
- 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.
- Once your site security is established, contact a trading partner to test transfers against your server.
- Test with various file sizes to ensure that your CGI can process small and large files.
- 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.
- Test application with various simulated errors in both file transfers and in PGP or OpenPGP decryption.

TAB 8 – TABLES

TABLE A – INTERNET EDM STANDARD ERROR CODES AND MESSAGES

These errors and warnings are strictly related to problems found in the Receiving program 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 Receiving program or decryption. Numbering schemes and descriptions should aid in this distinction.

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	Data Element Required or. Mutually Agreed
EEDM100	Missing "from" Common Code Identifier code	from	required
EEDM101	Missing "to" Common Code Identifier	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" Common Code Identifier	from	required
EEDM106	Invalid "to" Common Code Identifier	to	required
EEDM107	Invalid input format	input-format	required
EEDM108	Invalid transaction set	transaction-set	mutually agreed
EEDM109 (OPEN ISSUE 016)	No parameters supplied	parameter string	required
EEDM110	Invalid "version"	version	required
EEDM111	Missing "version"	version	required
EEDM112	"receipt-security-selection" not mutually agreed	receipt-security-selection	mutually agreed
EEDM113	Invalid "receipt-security-selection"	receipt-security-selection	mutually agreed

Validation Code	Description	Data Element	Data Element Required or. Mutually Agreed
EEDM114	Missing "receipt-disposition-to"	receipt-disposition-to	required
EEDM115	Invalid "receipt-disposition-to"	receipt-disposition-to	required
EEDM116	Missing "receipt-report-type"	receipt-report-type	required
EEDM117	Invalid "receipt-report-type"	receipt-report-type	required
EEDM118	Missing "receipt-security-selection"	receipt-security-selection	mutually agreed
EEDM119	Mutually agreed element, refnum, not present	refnum	mutually agreed
EEDM601	Public key invalid	file itself	required - security
EEDM602	File not encrypted	file itself	required - security
EEDM603	Encrypted file truncated	file itself	required - security
EEDM604	Encrypted file not signed or signature not matched	file itself	required - security
EEDM699	Decryption Error		required for general decryption errors not specifically identified by PGP or OpenPGP messages or exit codes
EEDM701 (OPEN ISSUE 017)	EDM party not associated with EDI party		required
EEDM702 (OPEN ISSUE 017)	Data Structure Error		required if the translator does not handle this exception
EEDM703 (OPEN ISSUE 017)	Data set exchange not established for Trading Partner		required if the translator does not handle this exception
EEDM999	System error		required for general system errors to indicate severe errors in processing at the receiving site
WEDM100	Transaction set sent not mutually agreed	transaction-set	mutually agreed
WEDM102	"receipt-security-selection" not mutually agreed	receipt-security-selection	mutually agreed
WEDM103	Missing "receipt-security-selection"	receipt-security-selection	mutually agreed
WEDM104	Element refnum received, not mutually agreed; ignored	refnum	mutually agreed

TABLE B – FREQUENTLY ASKED QUESTIONS

As an end user, do I need a continuously connected internet Web server to participate in the Internet EDM in the energy industry, or can I just use a dial-up connection to my ISP and my favorite shrink-wrapped browser software?

An interactive browser connection is not enough to actively participate in the system. It is not necessary to have a private Web server, you can use a service, however the system requires that you have access to a permanent internet connection which is capable of both sending and receiving files without operator intervention.

If we use ANSI X12.58 encryption do we still need to use PGP or OpenPGP encryption?

The use of internal encryption such as X12.58 is outside the scope of the NAESB REQ encryption standards.

TABLE C – GLOSSARY OF INTERNET ELECTRONIC TRANSPORT TERMS

The following terms and conventions are used throughout this document.

Term	Definition or Convention
Batch Browser	A Browser that can be run with little or no manual operation or intervention. See "Browser"
Browser	A software program capable of generating HTTP Requests, including HTTP POST requests.
Client	The computer hardware and software used by the Sender to transmit an Electronic Package to the Receiver's Server. A Client can be fully-automated or manual.
COTS	Commercial Off-The-Shelf; software that can be purchased that requires little or no customization.
Electronic Package	A data stream sent via HTTP POST that contains header information and Payload File(s). The Payload Files are encrypted using defined Internet ET encryption techniques.
Error Notification	Errors that are trapped after the IET receipt is sent are communicated via an Error Notification package from the Receiver of the original data to the Sender.
HTTP Request	The stream of data sent from the Client to the Server that includes header information and payload data.
HTTP Response	The stream of data sent from the Server to the Client in response to an HTTP Request.
HTTP Server	A computer capable of receiving HTTP Requests and responding with HTTP Responses.
IETF	Internet Engineering Task Force; a body of technical experts that set standards, known as Requests for Comments (RFC) for the Internet.
Interactive Browser	A Browser that requires manual operation or intervention. See "Browser".
Internet EDM	The GISB and NAESB WGQ standards up to and including Version 1.7. The "Internet ET" and "QEDM" standards were derived from these EDM standards.
Internet Electronic Transport, ET	The NAESB standards for the secure transport of electronic information between trading partners.

Term	Definition or Convention
Package	See "Electronic Package."
Payload Files	The data contents inside of an electronic package. NAESB Internet ET does not care what the contents of ET Electronic Packages contain.
QEDM	Quadrant-specific Electronic Delivery Mechanism; the set of standards for each NAESB quadrant that define the business policies, practices and processes for that quadrant. The QEDM excludes electronic transport practices and standards. The QEDMs were derived from the Internet EDM from GISB and NAESB WGQ.
Quadrant-Specific Electronic Delivery Mechanism	See "QEDM".
Receipt	The HTTP Response sent from the Receiver to the Sender that includes a timestamp and OK/error status.
Receiver	The party that receives an electronic package.
Secure Electronic Package	See "Electronic Package"
Sender	The party that sends an Electronic Package.
Server	The computer hardware and software used by the Receiver to receive an Electronic Package from the Sender's Client. The Server is an HTTP/Web Server.
Web Browser	See "Browser"
Web Server	See "HTTP Server".