Ontario EBT Data Transport Protocol

Version ~~2.2 Patch 1~~3.0

Published by:

Ontario EBT Working Group

~~October 25~~January 10, 200~~4~~5

NOTICE OF DISCLAIMER AND LIMITATION OF LIABILITY

The information provided is directed solely to professionals who have the appropriate degree of experience to understand and interpret its contents in accordance with generally accepted engineering or other professional standards and applicable regulations. No recommendation as to products or vendors is made or should be implied.

While the information contained herein has been prepared from sources deemed to be reliable, The EBT Standards Working Group ("EBT WG"), operating under the auspices of the Ontario Energy Board ("OEB") or its successor organization reserves the right to revise the information without notice, but has no obligation to do so. Use of the information is at your discretion and THE OEB EBT WG MAKES NO REPRESENTATION OR WARRANTY THAT THE INFORMATION IS TECHNICALLY ACCURATE OR SUFFICIENT OR CONFORMS TO ANY STATUTE, GOVERNMENTAL RULE OR REGULATION AND MAKES NO REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR AGAINST INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS. IN NO EVENT SHALL THE OEB EBT WG OR ITS SUCCESSOR ORGANIZATION BE LIABLE FOR LOST PROFITS OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES. ANY AND ALL USE OF OR RELIANCE UPON SUCH INFORMATION, INCLUDING ANY SELECTION OF PRODUCTS OR VENDORS IS SOLELY YOUR RESPONSIBILITY AND YOU ASSUME ALL RISKS AND LIABILITIES, IF ANY, WITH RESPECT THERETO.

Formatted: Font: 12 pt

Formatted: Font: (Default) Times New Roman, 12 pt

CONTENTS

1. Executive Summary

This document defines the Internet Data Transport protocol and rules for EBT transactions as defined by the Ontario Energy Board's EBT Work Group for the deregulated Electric marketplace in Ontario, Canada.

2. Revision History

Author	Version	Date	Description
J. Cartwright	1.0	December 28, 2000	Initially published version
D. Darnell	1.1	January 14, 2001	
J. Cartwright	1.2	January 30, 2001	
J. Cartwright	1.3	February 7, 2001	Formatting. Schema modifications
J. Stewart	1.4	May 17, 2001	Clarifications per standards meetings, added Public Key Management from D. Darnell.
J. Stewart	2.0	August 3, 2001	GMT Changes and Keep Alive
J. Stewart	2.1	December 12, 2001	Added updates due to issues #559 and #595
D. Brooks	2.2	June 3, 2002	Enhancements to address issues #664/#666; Download Complete ACK
J. Stewart	2.2	October 18, 2002	Enhancement requested by EBT Advisory Committee to support two kinds of Download, one with and one without Download Complete.
T. Stark	2.2 Patch 1	October 25, 2004	Updates for GI 658
T. Stark	3.0	January 10, 2005	Updates for GI 765, addition of Disclaimer Statement

3. Introduction

This document clarifies and outlines the necessary protocol for data transport between Market Participant EBT senders and EBT Market Participant EBT recipients.

In this document, it is assumed that the reader is familiar with the HTTP transport protocol, including its message specifications. For technical specifications, please refer to documents listed under References.

3.1 Scope

The EBT Transport Protocol defines the technical and functional standard for EBT messaging in the Ontario deregulated electric market.

The EBT Transport Protocol consists of three parts:

- The Security Protocol defines the architecture to ensure EBT message integrity.
- The Message Protocol defines an overall framework for expressing the different types of exchanged messages and structure of these messages.
- The Delivery Protocol defines the logical flow of EBT messages.

3.2 Definitions

This document uses the following definitions:

CA	Certification Authority.
EBT Message	A transport protocol object that includes HTTP information and encapsulates one PIPE Document.
EBT Document	One XML document consisting of a PIPE Document, which in turn is made up of one or more EBT transactions.
HTTP	HyperText Transfer Protocol.
HTML	HyperText Markup Language.
PIPE	Partner Interface Protocol for Energy.
XML	eXtensible Markup Language.

4. Security Protocol

Secure transmission of EBT messages requires:

- Encryption of the EBT document such that only the intended recipient may read it.
- Digital signing of the EBT document such that the sender identity and data integrity may be verified.
- A secure connection between Market Participant senders and recipients, which, together with logon identification and password for the sender, ~~authenticates~~authenticate and ~~validates~~validate the identity of the party. Note that this does not require client or browser installed public key certificates. It does require the recipient's HTTP secure server to have a valid server side X.509 public key certificate.

4.1 Encryption and Signature

Software and parameters compliant with the OpenPGP Internet draft RFC2440 shall be used to encrypt and sign the EBT document prior to sending, ensuring that the contents of the document may not be tampered with and may only be decrypted by the intended recipient.

4.1.1 Rules

The following rules are in effect for security:

- Market Participant Senders and Market Participant Recipients must generate a public/private key-pair with a 1024-bit key length.
- Market Participants must have only one public key active at any time.

4.1.2 PGP Parameters and Options

The following PGP parameters and options must be used:

- Signing private key must be DSA at 1024 bits
- Encrypting public key must be El Gamal (ELG-E) at 1024 bits
- Cipher Algorithm must be Triple DES (3DES)
- Key expiration must be set at 2 years
- User ID must be in format "name (organization) <email address>"
- Message Digest Algorithm / Hash must be SHA
- Compression must be used: ZIP parameter option is preferred, in which compressed packets are compressed with RFC1951 DEFLATE

4.1.3 Public Key Availability

The public key should be registered with the Massachusetts Institute of Technology worldwide PGP public key-server. Although market participants are required to renew

Comment [TS1]: GI 765 – v3.0

PGP keys every 2 years, it is required that legacy PGP keys must be kept for a minimum of 7 years.

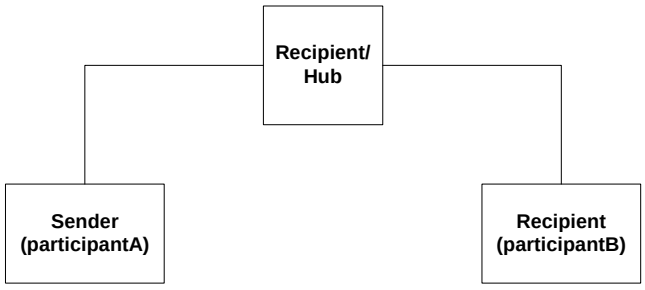
~~The recipient HTTPS server must have a readily available directory with an archive of the sender's PGP public keys. This archive will contain the Sender's public key files for a minimum period of 2 years, with a minimum of a full 7 years of archive available on offline storage media.~~

4.1.4 Usage

All EBT documents passing between a participant sender and a recipient must be encrypted using the public key of the recipient and signed with the private key of the sender.

4.1.4.1 Usage Examples

4.1.4.1.1 Participants on the same Hub



Participant A sends document intended for participant B.

1. Sender looks up Recipient/Hub public key on directory service.
2. Sender encrypts document using Recipient/Hub public key and signs document using Sender private key.
3. Recipient/Hub decrypts document using Recipient/Hub private key and verifies signature.
4. Recipient/Hub looks up Recipient public key on directory service.
5. Recipient/Hub encrypts document using Recipient public key and signs document using Recipient/Hub private key.
6. Recipient decrypts document using Recipient private key and verifies signature.

4.1.5 References

PGP homepage.	http://www.pgp.com
PGP international homepage.	http://www.pgpi.com
MIT public PGP keyserver.	http://pgp.mit.edu

OpenPGP IETF standards http://www.ietf.org/ids.by.wg/openpgp.html

4.2 Secure Transmission

The HTTPS protocol shall be used to provide a secure connection between the recipient's server and a sender's client computer, delivering an encrypted stream of data. This is necessary to protect the sender's logon ID and password.

HTTPS will utilize VeriSign Digital Ids to verify the identity of the recipient's Server and to encrypt the communication session.

4.2.1 Rules

The following rules are in effect for the HTTPS connection:

- The ~~recipientHubs~~recipient Hubs must have a valid VeriSign X.509 Digital ID installed on their server.
- The senders are not required to have a client-side Digital ID.
- EBT Documents must always be transferred to and from a recipient's server using the HTTPS protocol.
- The Recipients/hubs in a session must present a valid Digital ID.

4.2.2 HTTPS connection

- port 443

4.2.3 VeriSign Digital ID Class

- Global Server ID.

4.2.4 Session encryption type

- 128-bit

4.2.5 References

VeriSign homepage. http://www.verisign.com

5. Message Protocol

The Message Protocol defines the format of EBT messages exchanged between Market Participants and Hubs over the Internet. Using the Hypertext Transfer Protocol (HTTP), EBT messages in the form of data files will be sent over the Internet. This section outlines the HTTP request and response specifications.

NOTE: HTTP will be used within the HTTPS secure connection protocol.

5.1 Definitions

HTTP-Version HTTP/1.1

5.2 References

HTTP/1.1 Standards Document *Hypertext Transfer Protocol – HTTP/1.1*
Fielding, et. al

5.3 Message Format

EBT message exchange between a Market Participant and a Hub must follow the HTTP/1.1 protocol.

All interactions will involve an HTTP request followed by a corresponding HTTP response.

5.3.1 General Message Format

Refer to the HTTP/1.1 Standards Document.

5.3.2 Required Header Fields

Every HTTP message exchanged must contain the following general headers:

- Date
- Connection

5.3.2.1 Date

The Date field is an HTTP Date that is created by the origin client (Market Participant) in a request or by the origin server (Hub) in a response.

The Date format is according to RFC 1123 and includes a 24-hour clock with a time zone of GMT. For example:

Date: Sun, 06, Nov 1994 08:49:37 GMT

5.3.2.2 Connection

The Connection field in HTTP/1.1 should always be of type 'Keep-Alive'
The Connection format is:

Connection: Keep-Alive

5.3.3 Entity-Header Fields

Entity-Header fields are only included if the HTTP message contains an entity-body. Every HTTP message containing an entity-body must include the following mandatory entity-header fields:

- Content-Language
- Content-Length
- Content-Type

5.3.3.1 Content-Language

The Content-Language is:

Content-Language: en, fr

5.3.3.2 Content-Length

The Content-Length is:

Content-Length: x

where x is the size of the entity-body in bytes and must be greater than 0.

5.3.3.3 Content-Type

The Content-Type for is:

Content-Type: multipart/form-data; boundary=EBTpart;

5.3.4 Message-Body / Entity-Body

The Message-Body is the Entity-Body and may be either:

- an encrypted EBT document with sender and security information
- an XML HTTP response
- an XML HTTP response with a directory listing
- empty

5.3.4.1 Rules

The following rules are in effect for the message protocol:

- If the message-body is empty, entity-headers must not be included.
- The plaintext EBT document must not be larger than 500Mb prior to encryption and compression.

5.4 HTTP Requests

Senders send HTTP requests to a recipient or Hub to upload an EBT document, to initiate a download of an EBT document, to confirm the completion of a download or to get a directory listing.

5.4.1 Request-Line

The Request-Line follows the following format:

Method `SP` Request-URI `SP` HTTP-Version `CRLF`

5.4.1.1 Method

The Method indicates the "method to be performed on the resource identified by the Request-URI" and is case-sensitive.

Hubs only allow the POST method; all other methods will be responded to by a 501 HTTP response, indicating that the method is Not Implemented.

EBT documents are uploaded to the Recipient/Hub using the POST method. The entity-body is multipart containing the sender, security information and the encrypted EBT document to be uploaded.

NOTE: This EBT document must have a unique filename (written in the POST header information).

5.4.1.2 Request-URI

The Request-URI identifies the mailbox location on the hub where the request will be delivered. Each Recipient's server must supply this URI to Market Participant sender.

5.4.2 Request-Header Fields

The following is a list of mandatory request-header fields included with every HTTP request to the Recipient's server:

- Host

5.4.2.1 Host

The Host is:

Host: www.ontarioRecipientServer.com: `x`

where www.ontarioRecipientServer.com is the domain of a particular Recipient/Hub and `x` is the port number.

5.5 HTTP Responses

A Recipient/Hub will send an HTTP response for every request it receives from a sender. These responses are not the Functional Acknowledgements to PIPE documents, but server-generated messages to acknowledge the receipt of an HTTP request and to supply any data as the entity-body.

Errors reported by the Recipient/Hub must be those from the transport response schema. These error responses being returned by the application server at the Recipient/Hub must include an XML response that can be validated against the transport response schema. This XML response must correspond to the information being returned in the HTTP error code and message.

If the application server at the Recipient/Hub wishes to return an error code that is not part of the transport response schema, it should use the base error code for the class of error that it is reporting (i.e., 400 or 500).

Errors being returned before the request makes it to the application server at the Recipient/Hub (i.e., from firewalls) are not required to include the XML response suffix and can therefore use other standard error codes as required.

5.5.1 Status-Line

The Status-Line contains the protocol version, numeric status code, and an associated textual phrase.

HTTP-Version `SP` Status-Code `SP` Reason-Phrase `CRLF`

5.5.2 Status Code and Reason Phrase

An EBT Hub must support the following Status-Codes and Reason Phrases:

- 200 OK
- 400 Bad Request
- 403 Forbidden
- 404 Not Found
- 408 Request Time-Out
- 500 Internal Server Error
- 501 Not Implemented
- 505 HTTP Version not supported

5.5.2.1 200 OK

The 200 OK status code indicates that the request has succeeded.

For Download and DownloadX requests, the entity body a separate part with the PGP encrypted EBT document.

For a DownloadComplete request, the entity body contains XML with the HTTP response. The "200 OK" status code only indicates that the document has been transferred and not that decryption has succeeded, or any other checking has taken place without error. This response includes the document id of the document that was downloaded.

For an Upload request, the entity body contains XML with the HTTP response. The "200 OK" status code only indicates that the document has been transferred and not that decryption has succeeded, or any other checking has taken place without error. This response includes the new document id and name of the document that was uploaded within the request. The new document id may, or may not, be related to the name of the document.

For a Directory request the entity body contains XML with the server response and a list of documents. The entry for each document contains the document id, the name of the document and a date stamp that is assigned by the hub to allow the spoke to download the oldest file first and preserve a first in, first out order. The document id may, or may not, be related to the name of the document.

Errors that are not reportable via a Functional Acknowledgement may be detected after successfully uploading, or downloading, a document. Typical errors of this type include problems decrypting the document and documents with headers that are mangled. In such cases, the receiving party will report the problem to the sending party via e-mail, FAX or telephone conversation. In such cases, the following types of information must be reported:

- Error type;
- File name; and
- Date and time of the error.

5.5.2.2 400 Bad Request

The 400 Bad Request status code indicates that the Recipient could not understand the request due to malformed syntax.

The Sender should make modifications to the request before re-submitting it.

All messages where the Sender has apparently made an error in the request (4xx response) that are not explicitly included in this document will be responded to with this status code.

5.5.2.3 403 Forbidden

The 403 Forbidden status code indicates that the Recipient/Hub will not fulfill the request.

5.5.2.4 404 Not Found

This status code indicates that the Recipient/Hub cannot find the Request-URI.

5.5.2.5 408 Request Time-Out

After achieving a secure connection through HTTPS the Recipient/Hub will wait a predefined amount of time for an HTTP request. If no request is received then the session will time-out. The initial suggested timeout is 90 seconds.

5.5.2.6 500 Internal Server Error

The 500 Internal Server Error indicates that the Hub encountered an unexpected condition that prevented it from fulfilling the request. No explanation need be given as to the server error.

5.5.2.7 501 Not Implemented

The 501 Not Implemented status code indicates that the method in an HTTP request was not a POST and that the requesting party has not followed the proper Message Protocol.

5.5.2.8 505 HTTP Version Not Supported

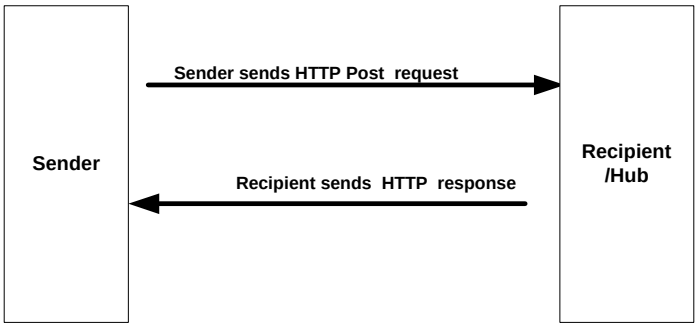
The HTTP Version must be specified as HTTP/1.1.
No other versions are supported.

6. Delivery Protocol

The Delivery Protocol defines how to exchange EBT documents using the HTTPS transport for PGP encrypted XML within the Ontario deregulated energy market. There are three possible request / response message scenarios:

- Scenario A: A Sender uploads an EBT document to a Recipient/Hub.
- Scenario B: A Sender downloads a directory (mailbox contents) from the Recipient/Hub.
- Scenario C: A Sender downloads an EBT document from their mailbox on a Recipient/Hub.

In all cases the following represents the flow of data:



6.1 General Rules

For the delivery protocol, the following rules apply:

- After sending any HTTP request, Market Participant Senders should not send another HTTP request to the same Recipient Server/Hub for **5** seconds.

6.2 Entity-Body Common Elements

Each multipart section of the entity body will begin with the boundary string:

EBTpart

6.2.1 Sender

Content-Disposition: form-data; name=”sender”
12345678

where `12345678` is the Sender's OEB License Number, or in the case of a hub the pseudo OEB license number, of the entity that issued the HTTP request.

Comment [TS2]: GI 658

6.2.2 User_id

Content-Disposition: form-data; name="user_id"
`User`

where `User` is the User's allocated User_id.

The User_id should be a string with a minimum length 4 characters, maximum length 16 characters, consisting only of alphanumeric characters.

The User_id and its associated password is used to authenticate that the sender of the request is a valid trading partner known to the hub and is allowed to use the requested service. This information is passed with every request.

6.2.3 user_password

Content-Disposition: form-data; name="user_password"
`Password`

where `Password` is the User's password.

The password should be a string, minimum length 8 characters, maximum length 16 characters, consisting only of alphanumeric characters.

6.2.3.1 request_type

Content-Disposition: form-data; name="request_type"
`Request`

Where request is a string and may only be:

- Upload
- Download
- DownloadX
- DownloadComplete
- Directory
- Unknown

The 'Upload' request is used when uploading an EBT document file to the hub.

The 'Download' request is used to download an EBT document file corresponding to a given DocId from the hub. No ~~explicite~~explicit 'DownloadComplete' is required, so the

hub can remove the downloaded DocId from the directory listing upon a normal closure of the socket.

The 'DownloadX' request is used to download an EBT document file corresponding to a given DocId from the hub where an ~~explicite~~explicit 'DownloadComplete' is required. The hub will not remove the downloaded DocId from the directory listing until it receives a 'DownloadComplete' ~~requrest~~request.

The 'DownloadComplete' request is used to indicate to the hub that it should remove the specified DocId from the directory listing.

The 'Directory' request is used to obtain a list of DocIds in the sender's mailbox from the hub.

The 'Unknown' request is not sent as part of a request. It is to be used within a response where the original request does not equal one of the valid request types.

6.3 Scenario A—Upload Request

6.3.1 Overview

A Market Participant uploads an EBT document to a Recipient Server/Hub. See Appendix A for a sample message exchange for Scenario A.

6.3.2 Description

A Market Participant sends an EBT document within an HTTP request using the HTTP POST method to a Recipient Server/Hub.

The Recipient Server/ Hub replies with an appropriate HTTP response, verifying receipt of the request and indicating whether it was successfully received or whether there were any errors during the transfer.

6.3.3 Rules

The following rules apply for scenario A:

- Market Participants must only send EBT documents with themselves as the Sender in the TradingPartnerDirectory node of the EBT document.

NOTE: this will be different for hub to hub protocol.

6.3.4 Entity-body - ebt_Document

The EBT document part of the entity body has the following format:

Content-Disposition: form-data; name="ebt_document";
filename="`transaction.xml`"
Content-Type: application/octet-stream

MIME-Version: 1.0
Content-Type: multipart/encrypted; boundary="=--";
protocol="application/pgp-encrypted"

--=--
Content-Type: pgp-encrypted

Version: 1

--=--
Content-Type: application/octet-stream

-----BEGIN PGP MESSAGE-----
Version: 2.6.2

```
hIwDY32hYGCE8MkBA/wOu7d45aUxF4Q0RKJprD3v5Z9K1YcRJ2fve8
7lMlDlx4OjeW4GDdBfLbJE7VUpp13N19GL8e/AqbyyjHH4aS0YoTk1
0QQ9nnRvjY8nZL3MPXSZg9VGQxFeGqzykzmykU6A26MSMexR4ApeeO
N6xzZWfo+0yOqAq6lb46wsvldZ96YA
AABH78hyX7YX4uT1tNCWEIIBoqqvCeIMpp7UQ2IzBrXg6GtukS8Nxb
ukLeamqVW31yt21DYOjuLzcMNe/JNsD9vDVCvOOG3OCi8=
=zzaA
```

-----END PGP MESSAGE-----
--=----
--EBTpart--

where `transaction.xml` is the Sender allocated name of the file.
This filename should be a string, maximum length 255 characters.

6.4 Scenario B—Directory Request

6.4.1 Overview

A Sender requests a directory (mailbox contents) from the Recipient/Hub. See Appendix B for a sample message exchange for Scenario B.

6.4.2 Description

A Sender sends an HTTP request using the HTTP POST method to a Recipient/Hub. The Recipient/Hub replies with an appropriate HTTP response, verifying receipt of the request and indicating whether it was successfully received or whether there were any errors during the transfer.

A response to a successful request includes an XML entity-body with a <DOCUMENT> entity for each EBT message in the mailbox.

6.5 Scenario C—Download Request

6.5.1 Overview

A Sender downloads an EBT document from their mailbox on a Recipient/Hub. See Appendix C for a sample message exchange for Scenario C. Downloads may be either the simple 'Download', or they may be of the type 'DownloadX' where an ~~explicite~~explicit confirmation of the download is required.

6.5.2 Description

A Sender sends an HTTP request using the HTTP POST method to a Recipient/Hub. The Recipient/Hub replies with an appropriate HTTP response, verifying receipt of the request and indicating whether it was successfully received or whether there were any errors during the transfer.

A response to a successful request includes an encrypted EBT document within the entity-body.

There are two separate kinds of download:

- A simple 'Download' style of request where the hub requires no further action by the recipient after the EBT document has been successfully downloaded. The Hub assumes the document was correctly downloaded based on a successful closure of the TCP socket
- The 'DownloadX' style of request where the recipient MUST explicitly send a DownloadComplete message via HTTP POST to the Hub after an EBT document has been successfully downloaded. Receipt of the DownloadComplete message by the Hub serves as a positive acknowledgement that the Recipient has successfully received an EBT document. The Hub responds the to DownloadComplete request with an HTTP Response of 200 OK indicating success.

6.5.3 Rules

The following rules apply for scenario C:

- Only one EBT document doc_id will be included in a request.
- Only one EBT document will be included in a response

6.5.4 Entity-body – doc_id

Content-Disposition: form-data; doc_id="12345678";

Where 12345678 is a string, with a minimum length of 1 character and a maximum length of 255 characters. The doc_id is a handle and may, or may not, be related to the actual file name of the EBT Document.

7. Market Participant Recipient Server/Hub Processing

The following rules apply with respect to the movement and storage of EBT Document files:

- The ~~Recipient/~~Hubs ~~will~~ shall move files from the Sender's mailbox to an archive mailbox upon completion of the downloading process. These files ~~will~~ shall remain available in an on-line archive for ~~the required~~a minimum ~~time as dictated by the OEB~~2 years. Furthermore, these files shall remain available in an off-line archive for a period of 7 years.
- The method by which the off-line archive documents are transferred to the market participants from the Hubs is subject to the individual service level agreements between the market participants and the Hubs.
- The method by which the off-line archive documents are transferred between the Hubs must be the same as the method by which on-line archive documents are transferred. That is, the appropriately encrypted documents must be placed in the recipient Hub's mailbox using the Hub-To-Hub protocol. ~~The Market Participant Recipient Server will maintain files in the Sender's archive mailbox for a minimum period of time as dictated by the OEB~~
-
- If the ~~Market Participant Recipient Server~~Hubs have ~~has~~ moved files ~~moved~~ to offline secondary storage, the files must be available in the original encrypted/digitally signed state. This offline secondary storage must be a production-level storage device.
- The Points are bound by the same rules as above.

Comment [TS3]: GI 765 – v3.0

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

Formatted: Bullets and Numbering

Appendix A -- Scenario A—Upload Request

A.1 HTTP Request—Upload

```
POST /RecipientServer/mailbox HTTP/1.1
Date: Tue, 20 Dec 2000 08:12:31 GMT
Connection: Keep-Alive
Host: www.ontarioRecipientServer.com
Content-Language: en, fr
Content-Type: multipart/form-data; boundary=EBTpart;
Content-Length: 3222

--EBTpart
Content-Disposition: form-data; name=”sender”

12345678

--EBTpart
Content-Disposition: form-data; name=”user_id”

aUser

--EBTpart
Content-Disposition: form-data; name=”user_password”

aPassword

--EBTpart
Content-Disposition: form-data; name=”request_type”

Upload

--EBTpart
Content-Disposition: form-data; name=”ebt_document”;
filename=”transaction.xml”
Content-Type: application/octet-stream

MIME-Version: 1.0
Content-Type: multipart/encrypted; boundary=”=--”;
protocol="application/pgp-encrypted"

--=--
Content-Type: pgp-encrypted

Version: 1

--=--
Content-Type: application/octet-stream

-----BEGIN PGP MESSAGE-----
Version: 2.6.2

hIwDY32hYGCE8MkBA/wOu7d45aUxF4Q0RKJprD3v5Z9K1YcRJ2fve87lMlDlx4OjeW
4GDdBfLbJE7VUpp13N19GL8e/AqbyyjHH4aS0YoTk10QQ9nnRvjY8nZL3MPXSZg9VG
```

```
QxFeGqzykzmykU6A26MSMexR4ApeeON6xzZWfo+0yOqAq6lb46wsvldZ96YA
AABH78hyX7YX4uT1tNCWEIIBoqqvCeIMpp7UQ2IzBrXg6GtukS8NxbukLeamqVW31y
t21DYOjuLzcMNe/JNsD9vDVCvOOG3OCi8=
=zzaA
-----END PGP MESSAGE-----
--=----
--EBTpart--
```

A.2 HTTP Response—Upload

See Appendix D.1 “200 OK Upload” for a sample HTTP response.

Appendix B -- Scenario B—Directory Request

B.1 HTTP Request—Directory

```
POST /RecipientServer/mailbox HTTP/1.1
Date: Tue, 20 Dec 2000 08:12:31 GMT
Connection: Keep-Alive
Host: www.ontarioRecipientServer.com
Content-Language: en, fr
Content-Type: multipart/form-data; boundary=EBTpart;
Content-Length: 3222

--EBTpart
Content-Disposition: form-data; name=”sender”

12345678

--EBTpart
Content-Disposition: form-data; name=”user_id”

aUser

--EBTpart
Content-Disposition: form-data; name=”user_password”

aPassword

--EBTpart
Content-Disposition: form-data; name=”request_type”

Directory
--EBTpart--
```

B.2 HTTP Response—Directory

See Appendix D.4 “200 OK Directory” for a sample HTTP response.

Appendix C -- Scenario C—Download Request

C.1 HTTP Request—Simple Download

```
POST /RecipientServer/mailbox HTTP/1.1
Date: Tue, 20 Dec 2000 08:12:31 GMT
Connection: Keep-Alive
Host: www.ontarioRecipientServer.com
Content-Language: en, fr
Content-Type: multipart/form-data; boundary=EBTpart;
Content-Length: 3222

--EBTpart
Content-Disposition: form-data; name=”sender”

12345678

--EBTpart
Content-Disposition: form-data; name=”user_id”

aUser

--EBTpart
Content-Disposition: form-data; name=”user_password”

aPassword

--EBTpart
Content-Disposition: form-data; name=”request_type”

Download

--EBTpart
Content-Disposition: form-data; doc_id=”12345678”

--EBTpart--
```

C.2 HTTP Request—Download with Explicit Confirmation

```
POST /RecipientServer/mailbox HTTP/1.1
Date: Tue, 20 Dec 2000 08:12:31 GMT
Connection: Keep-Alive
Host: www.ontarioRecipientServer.com
Content-Language: en, fr
Content-Type: multipart/form-data; boundary=EBTpart;
Content-Length: 3222

--EBTpart
Content-Disposition: form-data; name=”sender”

12345678

--EBTpart
```

```
Content-Disposition: form-data; name="user_id"

aUser

--EBTpart
Content-Disposition: form-data; name="user_password"

aPassword

--EBTpart
Content-Disposition: form-data; name="request_type"

DownloadX

--EBTpart
Content-Disposition: form-data; doc_id="12345678"

--EBTpart--
```

C.3 HTTP Response—Download

See Appendix D.2 "200 OK Download" for a sample HTTP response.

C.4 HTTP Request—DownloadComplete

```
POST /RecipientServer/mailbox HTTP/1.1
Date: Tue, 20 Dec 2000 08:12:31 GMT
Connection: Keep-Alive
Host: www.ontarioRecipientServer.com
Content-Language: en, fr
Content-Type: multipart/form-data; boundary=EBTpart;
Content-Length: 3222

--EBTpart
Content-Disposition: form-data; name="sender"

12345678

--EBTpart
Content-Disposition: form-data; name="user_id"

aUser

--EBTpart
Content-Disposition: form-data; name="user_password"

aPassword

--EBTpart
Content-Disposition: form-data; name="request_type"

DownloadComplete
```

```
--EBTpart
Content-Disposition: form-data; doc_id=”12345678”

--EBTpart--
```

C.5 HTTP Response—DownloadComplete

See Appendix D.3 “200 OK DownloadComplete” for a sample HTTP response.

Appendix D -- Sample HTTP Responses

D.1 200 OK. Upload

```
HTTP/1.1 200 OK
Date: Tue, 20 Dec 2000 08:12:31 GMT
Connection: Keep-Alive
Host: www.ontarioRecipientServer.com
Content-Type: text/XML
Content-Length: 1234

<?xml version=”1.0” encoding=”UTF-8”?>
<RESPONSE xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-
instance"
xsi:noNamespaceSchemaLocation="http://www.oeb.gov.on.ca/Response.
xsd">
  <HTTP_RESPONSE>
    <STATUS_CODE>200</STATUS_CODE>
    <REASON_PHRASE>OK</REASON_PHRASE>
    <REQUEST_TYPE>Upload</REQUEST_TYPE>
    <TIMESTAMP>Tue, 20 Dec 2000 08:12:31 GMT</TIMESTAMP>
  </HTTP_RESPONSE>
  <UPLOAD new_doc_id=”12345678” doc_name=”abcdefg.xml”/>
</RESPONSE>
```

D.2 200 OK. Download

```
HTTP/1.1 200 OK
Date: Tue, 20 Dec 2000 08:12:31 GMT
Connection: Keep-Alive
Host: www.ontarioRecipientServer.com
Content-Length: 3222
Content-Disposition: attachment; filename=transaction.xml
Content-Type: application/octet-stream

MIME-Version: 1.0
Content-Type: multipart/encrypted; boundary=”=--”;

protocol="application/pgp-encrypted"

--=--
Content-Type: pgp-encrypted

Version: 1

--=--
Content-Type: application/octet-stream

-----BEGIN PGP MESSAGE-----
Version: 6.5.2

hIwDY32hYGCE8MkBA/wOu7d45aUxF4Q0RKJprD3v5Z9K1YcRJ2fve87lMlDlx4OjeW
4GDdBfLbJE7VUpp13N19GL8e/AqbyyjHH4aS0YoTk10QQ9nnRvjY8nZL3MPXSZg9VG
QxFeGqzykzmykU6A26MSMexR4ApeeON6xzZWfo+0yOqAq6lb46wsvldZ96YA
```

```
AABH78hyX7YX4uT1tNCWEIIBoqqvCeIMpp7UQ2IzBrXg6GtukS8NxbukLeamqVW31y
t21DYOjuLzcMNe/JNsD9vDVCvOOG3OCi8=
=zzaA
-----END PGP MESSAGE-----
--=----
```

D.3 200 OK. DownloadComplete

```
HTTP/1.1 200 OK
Date: Tue, 20 Dec 2000 08:12:31 GMT
Connection: Keep-Alive
Host: www.ontarioRecipientServer.com
Content-Type: text/XML
Content-Length: 1234

<?xml version=”1.0” encoding=”UTF-8”?>
<RESPONSE xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-
instance"
xsi:noNamespaceSchemaLocation="http://www.oeb.gov.on.ca/Response.
xsd">
  <HTTP_RESPONSE>
    <STATUS_CODE>200</STATUS_CODE>
    <REASON_PHRASE>OK</REASON_PHRASE>
    <REQUEST_TYPE>DownloadComplete</REQUEST_TYPE>
    <TIMESTAMP>Tue, 20 Dec 2000 08:12:31 GMT</TIMESTAMP>
  </HTTP_RESPONSE>
  <DOWNLOADCOMPLETE doc_id=”12345678” />
</RESPONSE>
```

D.4 200 OK. Directory

```
HTTP/1.1 200 OK
Date: Tue, 20 Dec 2000 08:12:31 GMT
Connection: Keep-Alive
Host: www.ontarioRecipientServer.com
Content-Type: text/XML
Content-Length: 1234

<?xml version=”1.0” encoding=”UTF-8”?>
<RESPONSE xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.oeb.gov.on.ca/Response.x
sd">
  <HTTP_RESPONSE>
    <STATUS_CODE>200</STATUS_CODE>
    <REASON_PHRASE>OK</REASON_PHRASE>
    <REQUEST_TYPE>Directory</REQUEST_TYPE>
    <TIMESTAMP>Tue, 20 Dec 2000 08:12:31 GMT</TIMESTAMP>
  </HTTP_RESPONSE>
  <DIRECTORY>
    <DOCUMENT doc_id=”12345678”
         doc_file=”12345678.xml”
         doc_received_date=” Tue, 20 Dec 2000 07:10:22 GMT”/>
    <DOCUMENT doc_id=”23456789”
         doc_file=”23456789.xml”
         doc_received_date=” Tue, 20 Dec 2000 07:05:06 GMT”/>
```

```
  </DIRECTORY>
</RESPONSE>
```

D.5 400 Bad Request

```
HTTP/1.1 400 Bad Request
Date: Tue, 20 Dec 2000 08:12:31 GMT
Connection: Keep-Alive
Host: www.ontarioRecipientServer.com
Content-Type: text/XML
Content-Length: 1234

<?xml version=”1.0” encoding=”UTF-8”?>
<RESPONSE xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.oeb.gov.on.ca/Response.x
sd">
  <HTTP_RESPONSE>
    <STATUS_CODE>400</STATUS_CODE>
    <REASON_PHRASE>Bad_Request</REASON_PHRASE>
    <REQUEST_TYPE>Upload</REQUEST_TYPE>
    <TIMESTAMP>Tue, 20 Dec 2000 08:12:31 GMT</TIMESTAMP>
  </HTTP_RESPONSE>
</RESPONSE>
```

D.6 403 Forbidden

```
HTTP/1.1 403 Forbidden
Date: Tue, 20 Dec 2000 08:12:31 GMT
Connection: Keep-Alive
Host: www.ontarioRecipientServer.com
Content-Type: text/XML
Content-Length: 1234

<?xml version=”1.0” encoding=”UTF-8”?>
<RESPONSE xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.oeb.gov.on.ca/Response.x
sd">
  <HTTP_RESPONSE>
    <STATUS_CODE>403</STATUS_CODE>
    <REASON_PHRASE>Forbidden</REASON_PHRASE>
    <REQUEST_TYPE>Upload</REQUEST_TYPE>
    <TIMESTAMP>Tue, 20 Dec 2000 08:12:31 GMT</TIMESTAMP>
  </HTTP_RESPONSE>
</RESPONSE>
```

D.7 404 Not Found

```
HTTP/1.1 404 Not Found
Date: Tue, 20 Dec 2000 08:12:31 GMT
Connection: Keep-Alive
Host: www.ontarioRecipientServer.com
Content-Type: text/XML
Content-Length: 1234

<?xml version=”1.0” encoding=”UTF-8”?>
```

```
<RESPONSE xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.oeb.gov.on.ca/Response.x
sd">
  <HTTP_RESPONSE>
    <STATUS_CODE>404</STATUS_CODE>
    <REASON_PHRASE>Not_Found</REASON_PHRASE>
    <REQUEST_TYPE>Download</REQUEST_TYPE>
    <TIMESTAMP>Tue, 20 Dec 2000 08:12:31 GMT</TIMESTAMP>
  </HTTP_RESPONSE>
</RESPONSE>
```

D.8 408 Request Time-Out

```
HTTP/1.1 408 Request Time-Out
Date: Tue, 20 Dec 2000 08:12:31 GMT
Connection: Keep-Alive
Host: www.ontarioRecipientServer.com
Content-Type: text/XML
Content-Length: 1234

<?xml version=”1.0” encoding=”UTF-8”?>
<RESPONSE xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.oeb.gov.on.ca/Response.x
sd">
  <HTTP_RESPONSE>
    <STATUS_CODE>408</STATUS_CODE>
    <REASON_PHRASE>Request_Time-Out</REASON_PHRASE>
    <REQUEST_TYPE>Upload</REQUEST_TYPE>
    <TIMESTAMP>Tue, 20 Dec 2000 08:12:31 GMT</TIMESTAMP>
  </HTTP_RESPONSE>
</RESPONSE>
```

D.9 500 Internal Server Error

```
HTTP/1.1 500 Internal Server Error
Date: Tue, 20 Dec 2000 08:12:31 GMT
Connection: Keep-Alive
Host: www.ontarioRecipientServer.com
Content-Type: text/XML
Content-Length: 1234

<?xml version=”1.0” encoding=”UTF-8”?>
<RESPONSE xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.oeb.gov.on.ca/Response.x
sd">
  <HTTP_RESPONSE>
    <STATUS_CODE>500</STATUS_CODE>
    <REASON_PHRASE>Internal_Server_Error</REASON_PHRASE>
    <REQUEST_TYPE>Directory</REQUEST_TYPE>
    <TIMESTAMP>Tue, 20 Dec 2000 08:12:31 GMT</TIMESTAMP>
  </HTTP_RESPONSE>
</RESPONSE>
```

D.10 501 Not Implemented

```
HTTP/1.1 501 Not Implemented
Date: Tue, 20 Dec 2000 08:12:31 GMT
```

```
Connection: Keep-Alive
Host: www.ontarioRecipientServer.com
Content-Type: text/XML
Content-Length: 1234

<?xml version="1.0" encoding="UTF-8"?>
<RESPONSE xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.oeb.gov.on.ca/Response.x
sd">
  <HTTP_RESPONSE>
    <STATUS_CODE>501</STATUS_CODE>
    <REASON_PHRASE>Not_Implemented</REASON_PHRASE>
    <REQUEST_TYPE>Download</REQUEST_TYPE>
    <TIMESTAMP>Tue, 20 Dec 2000 08:12:31 GMT</TIMESTAMP>
  </HTTP_RESPONSE>
</RESPONSE>
```

D.11 505 HTTP Version Not Supported

```
HTTP/1.1 505 HTTP Version Not Supported
Date: Tue, 20 Dec 2000 08:12:31 GMT
Connection: Keep-Alive
Host: www.ontarioRecipientServer.com
Content-Type: text/XML
Content-Length: 1234

<?xml version="1.0" encoding="UTF-8"?>
<RESPONSE xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.oeb.gov.on.ca/Response.x
sd">
  <HTTP_RESPONSE>
    <STATUS_CODE>505</STATUS_CODE>
    <REASON_PHRASE>HTTP_Version_Not_Supported</REASON_PHRASE>
    <REQUEST_TYPE>Upload</REQUEST_TYPE>
    <TIMESTAMP>Tue, 20 Dec 2000 08:12:31 GMT</TIMESTAMP>
  </HTTP_RESPONSE>
</RESPONSE>
```

Appendix E -- EBT Public Key Management

This appendix describes the details of the manual management of keys for both hub and spoke. This Appendix should be viewed in the light of "best practices." Further, the intent of this document is to provide guidelines by which EBT hubs and market participants may conduct the management of keys.

E.1 Background

In the transfer of data to and from hub or spoke there are inherent risks of security. Primary among these risks is the possibility of public or private keys falling into unauthorized hands. The suggested infrastructure contained in this appendix addresses the management of keys in order to safe-guard the data transfer process. The person or persons responsible for safe guarding public keys should be kept to a minimum and shall be referenced in this appendix as a security unit (Key Manager Unit, KMU).

E.2 Scope

The suggested EBT Key Management Protocol defines the technical and functional standard for manually managing public and private keys between EBT senders and EBT recipients in the Ontario deregulated electric market.

The management categories included in this protocol document are:

- Public Key / Private Key Creation / Public Key protection
- Public Key distribution
- Public Key Storage (PKCS #12)
- Retiring / Revoking Public keys
- Private Key security / Private Key Storage
- Expiring keys and Keys that expire
- Window key management (between expired or revoked keys)
- Spoke Proactive expiration (window anticipation)
- Use of Expired or Revoked keys

The intricacies of X.509 certificates, encryption and hashing algorithms, non-repudiation (digital signatures), Storage only repositories – Certificate Servers (Key Servers), Public Key Infrastructures (PKI), and Certification Authorities are beyond the scope of this document.

Further, the following documents are take precedence in the case of a conflict between this appendix content herein and OEB protocols:

- Ontario EBT Hub to Hub Protocol
- Ontario Electronic Business Transactions (EBT) Standards Document

E.3 References

For additional information, please see the following documents:

- "Ontario Electronic Business Transactions (EBT) Standards Document" (Business Rules Document), Ontario EBT Working Group.
- "Ontario EBT Hub to Hub Protocol", Ontario EBT Working Group;

E.4 Spoke and Hub Key Management

The public key storage and distribution process is currently manual. The use of certificates and keyservers are a future implementation option. In lieu of the aforementioned scenario, it is suggested that the following approach be defined that will ensure:

- A secure and auditable public key exchange with the sender providing proof of possession of the private key and maintaining an audit trail for expiration and revocation purposes.
- Proper policing and management (import/export, expiration, revocation) of the keys.
- Public human contact or contacts (KMU) to be responsible for the keys.

E.5 Public Key / Private Key Creation / Public Key protection

Each Market Participant is responsible for generating and policing their own public/private key pairs. An individual employed by the Market Participant's business entity shall be publicly designated as the Key Manager for that entity. The generated key pair should meet the requirements set forth in the Transport Protocol document (OpenPGP). The public key of the Market Participant will be "self-signed" before distribution.

E.6 Public Key distribution

As a best practice, the public keys should be distributed via e-mail or other method agreeable to the sender and recipient of the Public Key. It is advised that the Public Key be distributed solely by the unit (KMU) responsible for and in possession of the private key. Upon receipt of a Public Key the recipient will confirm authenticity of the public key by contacting the sender/owner of the Public Key via telephone and requesting a

reading of the Public Key's fingerprint. If the fingerprint of the Public Key received matches the fingerprint of that sent, the recipient should sign the Public Key in the PGP Public Key Ring. A Key Distribution Log will be kept by the Key Management Unit and made available to other Market Participants at their request. Making the distribution log available should be done so that everyone is apprized who has a copy of the public key. The log shall contain

1. The date sent
2. Method of distribution
3. Key ID(s)
4. Fingerprint(s)
5. Size or CRC of the extracted key(s)
6. The name of the person to whom the key was sent
7. The date they received the key(s) and confirmed authenticity
8. The production activation date/time of the key(s).

E.7 Public Key Storage

Public keys should be stored on securable electronic media such as a floppy disk, magnetic tape, or CD-ROM. When the electronic media is connected or installed to/on a computer, it should only be accessible by the Key Manager Unit and only when they are at the console of the computer.

E.8 Revoking Public keys

When a public key is revoked a new public/private key pair should be created at that time. The Market Participant is responsible for distributing the revoked key and the new key via the agreed distribution mechanism. They should refer to their Key Distribution Log to ensure that all parties who received the original/revoked key will be sent and do receive a copy of the revoked and new keys. If a Key Management entity of the KMU leaves employment of the Market Participant's business entity the public key should be revoked.

E.9 Private Key Security / Private Key Storage

As a best practice, the Key Management Unit is the only entity authorized to handle the private key on any type of electronic media. When the private key is installed on a computer attached to a network, proper physical and system security should be in place to ensure that the private key file is only accessible by the KMU and only when they are physically located at the computer's console. It is recommended that a dedicated user account be setup for this purpose, and that any automated encryption and decryption programs that make use of the private key for decryption and signing purposes also be

executed under the context of this user account. The user account should have no network logon privileges. When the private key file is stored on removable electronic media that media should be stored in a physically secure location.

E.10 Expiring keys and Keys that expire

Keys will be set to expire every 2 years per the Transport Protocol. Certificates are being left out for now, so no further action is necessary.

E.11 Proactive/windowed key expiration

New keys should be generated and distributed prior to the old key expiring, in order to allow for the distribution timeframe and to prepare automated systems so they do not report errors. It is recommended that the new key be distributed at least 7 and no more than 28 business days prior to the key expiration date. This will allow ample time for the key to be received and installed before it starts being used. The EBT sender will communicate an exact time and file identification to the EBT recipient when a new Public Key is activated and first used.

E.12 Use of Expired or Revoked keys

Expired and revoked keys shall not be used to encrypt or sign any documents. They may be used at the operator's discretion to decrypt or verify signature on archived documents