Proposed Transport Standard

Tim Bornholtz

John Gill

Kim Shiflette

Friday, December 12, 2003

EEAT - Proposed Transport Standard

General

The Proposed Transport Standard (PTS) is a Web Services based transport system that is able to support many business data exchange needs. Initially designed to support synchronous and asynchronous transport models, it is possible to support batch, near-time and real-time needs within this framework. Additionally, current designs in progress include two different client configurations: both participants have client software and server software, one participant has client software and utilizes that software to communicate with a participant's server.

Specifics

Introduction

PTS utilizes SOAP, HTTP, SSL, XML, Base64, zLib compression, and UUID. The protocol uses SOAP over HTTPS to ensure the privacy of the transmission. XML is used to describe routing information related to a transmission, allowing data transport systems to be focused on moving data with little specific knowledge about the information contained within the transmission. UUID is utilized to generate unique message tracking numbers to ensure easier identification of specific messages. Base64 and zLib are used to compress (zlib) and encode (Base64) the data to be moved. This makes a PTS system payload insensitive, capable of moving any type of data between business partners.

Details

Terms

HTTP

: Hyper Text Transport Protocol is the client/server TCP/IP protocol commonly used on the internet for the exchange of HTML documents. Because of the Textual/XML nature of SOAP, HTTP is also used for the exchange of SOAP messages.

SOAP

: Simple Object Access Protocol is a light-weight set of conventions for invoking code using XML over HTTP.

SSL

: Secure Sockets Layer is used to produce a secure connection over which to transmit information. The secure connection is based on "session encryption" which uses keys to transparently encrypt data as it is sent and transparently decrypt data as it is received.

XML

: Extensible Markup Language (XML) is used with the HPCP to format the data being transmitted. Use of XML, allows for flexibility and expandability of the data being sent - including both the CommonLine business application data, and additional data useful to the transport subsystem itself, such as optional information items, message correlation identifiers, and system performance and status information. Most importantly, use of XML allows for changes to the data being sent - such as the inclusion of additional data - in the future with minimal change to the transport subsystem. All that is required to process new data is for the sender to add the data to the transmission and for the receiver to extract (parse) the new XML formated data from the transmission and provide it to the application that will work with it.

  XML uses "tags" to identify the specific type of data being sent as well as the beginning and ending of the sections of data with in the larger transmission (which includes security and

transport information). Tags also use "attributes" to specify the identification and characteristics of the data.

1) tags are case sensitive
2) attribute values are always quoted

Base64

A file format using 64 ASCII characters to encode the six bit Binary Data values 0-63.

To convert data to base 64, the first byte is placed in the most significant eight bits of a 24-bit buffer, the next in the middle eight, and the third in the least significant eight bits. If there a fewer than three bytes to encode, the corresponding buffer bits will be zero. The buffer is then used, six bits at a time, most significant first, as indices into the string "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/" and the indicated character output. If there were only one or two input bytes, the output is padded with two or one "=" characters respectively. This prevents extra bits being added to the reconstructed data. The process then repeats on the remaining input data.

Base 64 is used when transmitting binary data through text-only media such as electronic mail.

RFC

Request For Comment documents

URI

Universal Resource Identifier

UUID

Universal Unique Identifier. These are 128 bit numbers that are guaranteed to be unique. The mechanism used to guarantee that UUIDs are Unique is through combinations of hardware addresses, time stamps and random seeds.

zLib

A lossless compressed data format that it independent of CPU type, operating system, file system, and character set, and can be used for interchange. In addition, zLib can be implemented readily in a manner not covered by patents, allowing it to be utilized freely. (See RFC 1950)

Business Requirements

1. Support batch, real-time, and near real-time requests.
2. Single transport method for all business application (payload insensitive).
3. Process should not require you to open payload to determine type and destination.
4. Can accommodate a variety of technical platforms (among schools, servicers, lenders, FAMS, guarantors and FSA).
5. Data must arrive in a defined sequence.
6. Highly secure – encryption required. Must adhere to privacy acts (cannot pass text in the clear, password protected, etc.)
7. Cost not a barrier to adoption or use.
8. No set size limits.
9. Guaranteed delivery.
10. No distribution royalties (no cost to the user that wants to utilize an implementation of the Transport Standard)
11. Support low-tech institutions (can be placed on the desktop).
12. Easy to use.
13. Easy for integration.
14. All institutions must be able to implement the technical solution.

15. Must use open standards.

EEAT Analysis

1. Support batch, real-time, and near real-time requests.
   According to EEAT member research, Web Services can accomplish this. For a definition of the different levels of service this implies, review the draft document included in this packet titled "EEAT Definitions of Real-Time, Near-Time, and Batch".

2. Single transport method for all business application
   The data transport is payload insensitive allowing any text or binary data to be transmitted. Based on discussions with DTBW members, the intent of this requirement is for payload insensitivity, allowing any data type to be moved through the PTS (payload insensitive)

3. Process should not require you to open payload to determine type and destination.
   Information for parsing the payload such as source and destination will be contained in the header information. This requirement speaks to providing payload package level routing information at the transport level. This is contrasted with many current systems where the header data record of the payload must be examined to determine the routable destination.

4. Can accommodate a variety of technical platforms (among schools, servicers, lenders, FAMS, guarantors and FSA).

   This requirement is fairly broad, but an acceptable level of compliance should be achieved with the EEAT recommendation to consider SOAP. SOAP is a specification with implementations currently existing for most every widespread platform.

5. Data must arrive in a defined sequence. Client controlled

   Based on clarification from the DTBW this is describing a dependency aware transport. {Because all data passed through the transport is handled in a synchronous manner, all data will always arrive in the order sent.}

6. Highly secure – encryption required. Must adhere to privacy acts (cannot pass text in the clear, password protected, etc.)
   128 bit SSL encryption will be used for US communications. Authentication is still being discussed. International exchanges must adhear to international requirements.

7. Cost not a barrier to adoption or use.
   This requirement seems to be focused on long term operational costs. There are many implementation methods available for all tools and libraries currently under evaluation, including no cost, low cost, and commercial.

8. No set size limits
   DTBW has directed the EEAT to recommend size limits for this transport method. One of the Research projects in the timeline is related to determining size limits.

9. Guaranteed delivery. (acknowledgement)
   This requirement is for both an acknowledgement of the receipt of payload, and a method of handling failed transmissions. All data passed through the transport is passed synchronously and thus will have an immediate acknowledgment of receipt.

10. No distribution royalties.
    The PTS must not be based on any technology for which there is a royalty. There are many implementation methods available for all tools and libraries currently under evaluation, including no cost, low cost, and commercial.

11. Support low-tech institutions

and

12. Easy to use.

and

13. Easy for integration.
    PTS must be able to be packaged into an intuitive package for non-technically proficient users.

14. All institutions must be able to implement the technical solution.
    PTS has no restrictions on who implements, or how the implementation is constructed, provided that the implementation adheres to the published Technical Specifications.

15. Must use Open Standards.
    PTS is based on common industry technologies and practices. No non-standard or fringe technologies are being considered. (IETF, W3L, PESC, Etc.)

Resource Needs

Developer:
Java or .Net

Hardware

Timelines

Task #	Task Name	Start	Finish	Resource Name	Dependencies
	* Note calculating Start/Finish dates from work. A realistic expectation is 2-3 hours of work per week for EEAT volunteers				
	Analysis				
1	*Define Business Requirements*	08/01/03	11/09/03	DTB Workgroup	
2	Clarify EEAT analysis questions	11/10/03	11/20/03	DTB Workgroup	5
3	Respond to EEAT Analysis.	01/25/04	02/08/04	DTB Workgroup	6
4	*Analysis of Business Requirements*				1
5	Identify clarafication questions	10/30/03	11/19/03	EEAT	4
6	Re-analysis based on DBT Claraifactions	11/20/03	01/25/04	EEAT	2,7
7	*Research*				8,11,14,15,16,17
8	Size Limitations - SOAP				9,10
9	Java	12/01/03	01/10/04	Tom Jacobs / Tim Bornholtz	
10	.Net	01/05/04	01/16/04	<none assigned>	
11	Registry system/Distributed Registry system				12,13
12	LDAP	01/05/04	01/12/04	Mark Malinoski	
13	UDDI	01/12/04	01/19/04	Mark Malinoski	
14	Webservices Messaging systems	Previously Completed Research			
15	Interoperbility - Java & .Net	Previously Completed Research			
16	Client only solutions (Mailboxing)	Previously Completed Research			
17	Size Limitations - XML Signatures				18,19
18	Java	01/11/04	01/25/04	Tim Bornholtz	
19	.Net	01/11/04	01/25/04	<none assigned>	
	Documentation				
20	*Protocol Specification Draft*				21,22
21	Draft	01/25/04	02/06/04	EEAT	6
22	Revision 1	02/18/04	02/26/04	EEAT	24
	Development				
23	*Create Prototypes*				24,25
24	Implement Draft Specification	02/06/04	02/18/04	Tim Bornholtz / John Gill	21
25	Revise Draft Implementation	02/26/04	03/06/04	Tim Bornholtz / John Gill	22
26	*Reference Implementation*				27,28
27	Java	03/06/04	03/31/04	Tim Bornholtz / PTI	25
28	.Net	03/06/04	03/31/04	Rob Morris / Nelnet	25
	Test				
29	*Develop Test Cases*	02/06/04	02/24/04	Kim Shifflette	21
30	*Develop Test Scripts*	02/24/04	03/09/04	Kim Shifflette	29
31	*Testing Environment*	03/06/04	03/20/04	EEAT	25,30
	Documentation				
32	Technical Documentation	03/06/04	03/20/04	John Gill/ Kim Shiflette	25
33	Quality of Service Proposal	03/20/04	04/03/04	Kim and JC	32