Executive Summary

Business Problem

Over time, the Higher Education community has developed multiple methods to exchange data and information with multiple trading partners and operating systems. Each partner has developed their own protocols, standards or utilized commercial products independently. The lack of a common standard to transport electronic data has caused each player in the Higher Education community to develop, maintain and support a wide variety of transport solutions and hindered its ability to use a common protocol across the Higher Education community. Differing transport methods make it difficult to support each trading partners proprietary needs. As technologies advance more organizations are looking for a real-time transport solution. Due to the lack of a standard transport, entities are forced to develop proprietary processes or purchase commercial products with licensing and royalties issues.

The primary means of transporting information today is the use of FTP and/or an electronic mail processes. Each of these has limitations and is not suitable for the exchange of information for real time processing or acknowledgement. Differing transport definitions and standards make it difficult to build interfaces that ensure simplicity and reliability as it relates to electronic exchange of data.

Solution

Since the Higher Education community share common points of interaction, the Data Transport workgroup was formed to develop a common solution for data transport with representation from each of the major players in the community.

The Data Transport Standard (DTS) is a specification for a web service architecture that enables entities to send and respond to any type of request (transaction, inquiry, report) utilizing standard web service protocols. DTS is a culmination of different specifications and defines how to use them together. DTS is a result of a PESC initiative to create a standard method to exchange data within the Higher Education community, regardless of the business process. It is a recommended replacement for POP3/SMTP (e-mail) and an industry wide solution for real-time or immediate requests. DTS offers a solution for transport and may coexist or replace FTP.

Web services are based on request-response patterns. These request-response patterns occur in a synchronous (uninterrupted) mode. DTS defines the request-response pattern to be used in a SOAP format. SOAP (Simple Object Access Protocol) is widely accepted and an open industry standard for exchanging messages. Some of the benefits of the SOAP standard are: speed, extensibility, interoperability and tool integration.

Below is a list of some of the advantages DTS offers to the Higher Education community as a transport standard.

- Guaranteed delivery – real time communication that confirms delivery and receipt, regardless if payload is processed real-time or not.
• Supports immediate and deferred processing requests.
• Single transport method for all business applications (payload insensitive – XML, flat file, binary file, etc).
• No need to analyze payload to determine type and destination.
• Can accommodate a variation of technical platforms (among schools, servicers, lenders, FAMS, guarantors and FSA).
• Highly secure - transparent encryption via HTTPS and authentication via X.509 digital certificates.
• Supports larger payloads.
• No distribution royalties.
• Uses open standards.
• Can be used both for internal and external communications.

The current landscape of the industry offers a unique opportunity for the Higher Education community to promote a common data transport standard. Many in the higher education community are either looking to develop a transport process or streamline and consolidate transports by the adoption or creation of a new transport process. The industry can be proactive and develop a standard that can meet the needs of multiple business sectors within our industry. PESC does not want to be reactive, as each sector develops its own solutions for data transport.
# Table of Contents

EXECUTIVE SUMMARY .................................................................................................................................................. III

TABLE OF CONTENTS .................................................................................................................................................. V

TABLES AND FIGURES ............................................................................................................................................... VI

1 INTRODUCTION ..................................................................................................................................................... 1

1.1 OVERVIEW ....................................................................................................................................................... 1
1.2 PURPOSE ........................................................................................................................................................... 1
1.3 SCOPE ............................................................................................................................................................... 1
1.4 INTENDED AUDIENCE .................................................................................................................................. 2
1.5 ORGANIZATION OF THE DOCUMENT ....................................................................................................... 2

2 OVERVIEW AND DESCRIPTION OF DTS SPECIFICATION .............................................................................. 3

3 DTS WSDL (WEB SERVICE DEFINITION LANGUAGE) ...................................................................................... 4

3.1 TYPES ............................................................................................................................................................ 4
3.2 MESSAGES ..................................................................................................................................................... 5
3.3 INTERFACES (PORTTYPES) .......................................................................................................................... 5
3.4 BINDINGS .................................................................................................................................................... 5
3.5 SERVICES ................................................................................................................................................... 7

4 DTS SOAP DESCRIPTIONS ............................................................................................................................... 9

4.1 SOAP HEADER ELEMENTS .......................................................................................................................... 9
4.2 DTS REQUEST SOAP (SEE THE SUPPLEMENTAL SECTION OF THIS DOCUMENT FOR EXAMPLE SOAP) ....... 10
    Header Elements .................................................................................................................................................. 10
    Body ................................................................................................................................................................... 12
4.3 DTS RESPONSE SOAP (SEE THE SUPPLEMENTAL SECTION OF THIS DOCUMENT FOR EXAMPLE SOAP) ..... 14
    Header Elements ............................................................................................................................................... 14
    Body ................................................................................................................................................................. 18

5 COMMUNICATION OR FRAMEWORK ERRORS GENERATED OUTSIDE DTS .............................................. 19

6 DTS ERROR HANDLING REQUIREMENTS: .................................................................................................... 20

6.1 DTS SERVICE ERROR HANDLING TABLE ............................................................................................... 20
6.2 FAULTS SUMMARY ....................................................................................................................................... 20
6.3 DTS CLIENT ERROR HANDLING ................................................................................................................ 25
6.4 SUPPLEMENTAL DOCUMENTATION ......................................................................................................... 26
    DTS WSDL 1.0 ............................................................................................................................................... 26
    Entire .NET DTS SOAP Request: .................................................................................................................. 31
    Entire .NET DTS SOAP Response: ................................................................................................................ 31

7 RECOMMENDATIONS ............................................................................................................................................. 33

GLOSSARY ................................................................................................................................................................. 34

ACKNOWLEDGEMENTS ....................................................................................................................................... 35

REVISION HISTORY .................................................................................................................................................. 36
Tables and Figures

Tables

Table 3.1 – WSDL Elements ................................................................. 4
Table 6.1 – Faults Summary Table .................................................... 20
1 Introduction

1.1 Overview

This document defines the Data Transport Standard (DTS), utilizing a set of non-proprietary Web Services specifications to transport data, along with clarifications and amendments to those specifications that provide interoperability. The contents were derived by creating reference implementations and from many Information Technology industry resources including, but not limited to:

- Simple Object Access Protocol (SOAP) 1.1
- Web Services Description Language (WSDL) 1.1
- Extensible Markup Language (XML) 1.0 (Second Edition)
- RFC2616: Hypertext Transfer Protocol -- HTTP/1.1
- Web Services Interoperability Basic Profile 1.0
- RFC1950: Zlib Protocol

1.2 Purpose

The purpose of this specification is to describe the Data Transport Standard (DTS) web service through the use of Web Service Definition Language (WSDL). The SOAP elements defined are the minimum required to transport data in a manner that meets the business problem. This document provides the exact minimum structure for the SOAP that is transmitted between a web service and its client. If two entities created a web service client and service that transmits the SOAP described here, the applications should communicate correctly without further work.

It is intended to be a guide for PESC members and the education community in general, to use when implementing the Data Transport Standard throughout the community.

Additional guidance is provided that must be followed in order to ensure interoperability between Java and .Net platforms.

1.3 Scope

This specification provides a technical guideline for a web service to electronically exchange data between organizations. It focuses on defining the transport layer, the DTS WSDL, and DTS SOAP.

While this document identifies the necessary SOAP elements for data exchange using web services, the actual values contained in the SOAP elements and usage definitions for DTS in different industry segments will be specified by those industry segments. These values will be defined by those industry segments.
The following topics are not within the scope of this specification:

- The business reasons for adopting DTS as a data transport
- The integration of DTS with the file processing backend system
- How immediate or deferred (batch) data processing is performed

1.4 Intended Audience
This document is intended for experienced developers. It targets readers that have an intermediate to expert level of knowledge on Web Services, XML, .Net or Java, and cryptography. It is not intended to be a comprehensive tutorial on these topics. Only those topics that must be clarified for the purposes of standardization are covered.

1.5 Organization of the Document
The Guidelines for the DTS Specification consists of the following sections:

- **Section 1 – Introduction:**
  o provides a high level overview, scope, and assumptions of this document
- **Section 2 – Overview and Description of DTS Specification:**
  o provides an introduction to DTS and highlights topics that readers should be familiar with before reading this document
- **Section 3 - DTS WSDL (Web Services Description Language)**
  o provides a high level overview of the purpose of the DTS WSDL and outlines the requirements for constructing the request and response SOAP
- **Section 4 - DTS SOAP Descriptions:**
  o describes how a DTS Client and Service produces the SOAP requests and responses that meet the requirements of the WSDL
- **Section 5 - Communication or Framework Errors Generated Outside DTS:**
  o describes a list of possible external errors that can be encountered when creating a web service
- **Section 6 - DTS Error Handling Requirements:**
  o provides a list of all error conditions defined within the DTS framework and how they are reported
- **Section 7 – Recommendations:**
  o provides recommendations for topics of consideration that are outside the scope of the specification but may impact an organizations implementation of DTS
- **Section 8 – Glossary:**
  o provides a list of terms and definitions
- **Section 9 – Acknowledgements**
2 OVERVIEW AND DESCRIPTION OF DTS SPECIFICATION

DTS is designed to be a standard for creating web services that provide real-time transfer of data. It is built on the standard request/response scenario of web services. It is designed so any type of business could use this as a transport mechanism. DTS is based on computer industry standards in the Web Services arena:

- **WSDL** – Describes what functionality a Web Service offers, how it communicates and where you can find it
- **SOAP** - Specialized XML used as the basis for transporting data for Web Services
- **Digital Signatures** - Used for authentication
- **HTTPS** - Used for encryption and communication layer

DTS is designed to provide functionality in the following areas:

- **Core Services** – Build SOAP based on the specification
- **Security** (Encryption/Authentication) - Ensure that the information is encrypted and authenticated
- **Assured Delivery** – Assures that the request gets to the designated endpoint
- **Application Integration** – Provides business application integration to gain access to the low level SOAP elements for processing
- **Error Handling** – Provides error handling for each area in the specification
3 DTS WSDL (Web Service Definition Language)

The Web Services Definition Language (WSDL) is an XML-based language used to describe the services a business offers and to provide a way for individuals and other businesses to access those services electronically. It also provides a roadmap to how the SOAP will look in the Web Service transaction. The definition of the SOAP provides the foundation for this entire specification. Any organization implementing the specification must ensure that the SOAP produced from their applications conforms to the DTS WSDL.

There are five high level elements contained within a WSDL document. The table below summarizes each section.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Types</td>
<td>A container for abstract type definitions defined using XML Schema</td>
</tr>
<tr>
<td>message</td>
<td>A definition of an abstract message that may consist of multiple parts, each part may be of a different type</td>
</tr>
<tr>
<td>portType</td>
<td>An abstract set of operations supported by one or more endpoints (commonly known as an interface); operations are defined by an exchange of messages</td>
</tr>
<tr>
<td>binding</td>
<td>A concrete protocol and data format specification for a particular portType</td>
</tr>
<tr>
<td>Service</td>
<td>A collection of related endpoints, where an endpoint is defined as a combination of a binding and an address (URI)</td>
</tr>
</tbody>
</table>

Table 3.1 – WSDL Elements

3.1 Types

The WSDL types element is a container for XML Schema type definitions. The type definitions placed here are referenced from higher-level message definitions in order to define the structural details of the message. The types element contains zero or more schema elements from the http://www.w3.org/2001/XMLSchema namespace.

Below is an example of part of the types section defined in the DTS WSDL:

```xml
<xsd:complexType name="DTSRoutingElementsType">
  <xsd:sequence>
    <xsd:element name="DTSUUID" minOccurs="1" maxOccurs="1" type="xsd:string"/>
    <xsd:element name="DTSTransmissionDateTime" minOccurs="1" maxOccurs="1" type="xsd:string"/>
    <xsd:element name="DTSSourceID" minOccurs="1" maxOccurs="1" type="xsd:string"/>
    <xsd:element name="DTSSourceSubCode" minOccurs="1" maxOccurs="1" type="xsd:string"/>
    <xsd:element name="DTSRecipientID" minOccurs="1" maxOccurs="1" type="xsd:string"/>
    <xsd:element name="DTSRecipientSubCode" minOccurs="1" maxOccurs="1" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>

<!-- DTSRouting is of type DTSRoutingElementsType – see above -->
<xsd:element name="DTSRequestRouting" type="tns:DTSRoutingElementsType"/>
```
3.2 Messages

The WSDL message element defines an abstract message that can serve as the input or output of an operation. Messages consist of one or more part elements where each part is associated with an element. The messages and their parts must be named making it possible to refer to them from elsewhere in the WSDL definition.

Below is an example of part of the message section defined in the DTS WSDL:

```xml
<message name="DTSHeaders">
  <part name="DTSRequestRouting" element="tns:DTSRequestRouting"/>
  <part name="DTSResponseRouting" element="tns:DTSResponseRouting"/>
  <part name="DTSRequestServiceExpectation" element="tns:DTSRequestServiceExpectation"/>
  <part name="DTSRequestPayloadType" element="tns:DTSRequestPayloadType"/>
  <part name="DTSResponsePayloadType" element="tns:DTSResponsePayloadType"/>
  <part name="DTSResponseAcknowledge" element="tns:DTSResponseAcknowledge"/>
  <part name="DTSRequestSignature" element="tns:DTSRequestSignature"/>
  <part name="DTSResponseSignature" element="tns:DTSResponseSignature"/>
</message>
```

3.3 Interfaces (portTypes)

The WSDL portType element defines a group of operations also known as an interface in most environments. A portType element contains zero or more operation elements. Each portType must be given a unique name making it possible to refer to it from elsewhere in the WSDL definition. Each operation element contains a combination of input and output elements, and when you have an output element you can also have a fault (exception) element.

Below is an example from the portTypes section defined in the DTS WSDL:

```xml
<portType name="submitDTS">
  <operation name="submitDTS">
    <input message="tns:DTSRequest"/>
    <output message="tns:DTSResponse"/>
  </operation>
</portType>
```

3.4 Bindings

The WSDL binding element describes the concrete details of using a particular portType with a given protocol. The binding element contains several extensibility elements as well as a WSDL operation element for each operation in the portType which it describes. A binding must be given a unique name so you can refer to it from elsewhere in the WSDL definition. The binding must also specify which portType it describes through the type attribute.
Below is an example from the bindings section defined in the DTS WSDL:

```xml
<binding name="submitDTS" type="tns:submitDTS">
  <soap:binding style='"document"' transport='"http://schemas.xmlsoap.org/soap/http"'/>
  <operation name="submitDTS">
    <soap:operation soapAction='"http://www.datatransportstandard.com/submitDTS"'/>
    <input>
      <soap:body use='"literal"'/>
      <soap:header message='"tns:DTSHeaders"' part='"DTSRequestSignature"' use='"literal"'/>
      <soap:header message='"tns:DTSHeaders"' part='"DTSRequestRouting"' use='"literal"'/>
      <soap:header message='"tns:DTSHeaders"' part='"DTSRequestServiceExpectation"' use='"literal"'/>
      <soap:header message='"tns:DTSHeaders"' part='"DTSRequestPayloadType"' use='"literal"'/>
    </input>
    <output>
      <soap:body use='"literal"'/>
      <soap:header message='"tns:DTSHeaders"' part='"DTSResponseSignature"' use='"literal"'/>
      <soap:header message='"tns:DTSHeaders"' part='"DTSResponseRouting"' use='"literal"'/>
      <soap:header message='"tns:DTSHeaders"' part='"DTSResponsePayloadType"' use='"literal"'/>
      <soap:header message='"tns:DTSHeaders"' part='"DTSResponseAcknowledge"' use='"literal"'/>
    </output>
  </operation>
</binding>
```
3.5 Services

The WSDL service element defines a collection of ports, or endpoints, that expose a particular binding. You must give each port a name and assign it a binding. Then, within the port element, you use an extensibility element to define the address details specific to the binding.

Below is an example from the services section defined in the DTS WSDL:

```xml
<service name="submitDTS">
  <port name="submitDTS" binding="tns:submitDTS">
    <soap:address
      location="http://localhost:8080/services/ReferenceImplementation/submitDTS"/>
  </port>
</service>
```

Note: The Supplemental Documentation section of this document includes the full DTS WSDL.
4  DTS SOAP Descriptions

A DTS Client and Service must produce SOAP requests and responses that meet the requirements of the WSDL located in the supplemental documentation section of this document. There are additional guidelines that a DTS Client and Service must follow to allow for interoperability between .Net and Java. **Non-compliance with these additional guidelines WILL produce interoperability problems between the .Net and Java environments.**

**Guideline 1:**
On the root element of each header element, the `xsi:type` attribute must be present:
```
xsi:type="Qualified Name Of Element"
```

The following example shows the qualified name for the DTSRequestRouting header element:
```
xsi:type="DTSRequestRouting"
```

**Guideline 2:**
On the root element of each header element, the `xmlns` attribute must be present:
```
xmns="namespace of the service"
```

The following example shows the namespace attribute for the DTSRequestRouting header element:
```
xmns="urn:org:pesc:datatransport"
```

**Note:** The examples of the .Net and Java SOAP within this document do not exactly match.
- Namespace monikers are being added to the Java SOAP elements. Even though the monikers are present, they do not impact the interoperability between .Net and Java because the Qualified Name and Namespace of each element match what is expected.
- The `mustunderstand="0"` attribute is always being added to the Java SOAP header elements. The java toolkit used to produce the examples always adds the `mustunderstand` attribute even when validation is not required.

**Note:** The toolsets used in the reference implementations have dictated the creation of these guidelines. As technology advances and these toolsets mature the need for these guidelines may diminish or be removed.

4.1  SOAP Header Elements

The specification defines several header elements designed to allow any line of business the capability of transferring data without interrogating the request and/or response payload. Most of the header elements within the specification do not have any pre-determined values associated with them. This will allow each line of business to determine what values are necessary to perform their business functions. See the annotations (Specification defined vs. Business defined) beside each header element to determine if the value is necessary within the specification or to be defined by a line of business.

The following sections describe each of the DTS request and response SOAP packets.
4.2 DTS Request SOAP (See the supplemental section of this document for example SOAP)

Header Elements

- **DTSRequestRouting Element (Complex Type) (Business defined)**
  This element identifies routing information for the request.

  - **sourceId:** Identifies the source of the request.
  - **sourceIdSubCode:** Identifies a secondary code that helps identify the source of the request.
  - **recipientId:** Identifies the recipient of the request.
  - **recipientIdSubCode:** Identifies a secondary code that helps identify the recipient of the request.
  - **UUID:** Is a unique identifier of the request. See the recommendations section of this document for further clarification of UUID.
  - **transmissionDateTime:** The date/time stamp of the request must be in a GMT format: yyyy-MM-ddTHH:mm:ss:SSSTZD (Z or +hh:mm or –hh:mm).

  **Java Example DTSRequestRouting SOAP Element**

  ```
  <ns1:DTSRequestRouting soapenv:mustUnderstand="0" xsi:type="ns1:DTSRequestRouting"
  xmlns:ns1="urn:org:pesc:datatransport">
  <ns1:UUID>99d3c8ee-289a-4c17-b221-0e96155589e2 </ns1:UUID>
  <ns1:recipientID>A</ns1:recipientID>
  <ns1:recipientIDSubCode>Chitty</ns1:recipientIDSubCode>
  <ns1:sourceID>B</ns1:sourceID>
  <ns1:sourceIDSubCode>Malinoski</ns1:sourceIDSubCode>
  <ns1:transmissionDateTime>2006-10-23T22:33:20.281Z</ns1:transmissionDateTime>
  </ns1:DTSRequestRouting>
  ```

  **.Net Example DTSRequestRouting SOAP Element**

  ```
  <DTSRequestRouting xsi:type="DTSRequestRouting" xmlns="urn:org:pesc:datatransport">
  <UUID>99d3c8ee-289a-4c17-b221-0e96155589e2 </UUID>
  <transmissionDateTime>2006-10-23T22:33:20.281Z </transmissionDateTime>
  <sourceID>A</sourceID>
  <sourceIDSubCode>Chitty</sourceIDSubCode>
  <recipientID>B</recipientID>
  <recipientIDSubCode>Malinoski</recipientIDSubCode>
  </DTSRequestRouting>
  ```
• **DTSRequestPayloadType Element (Complex Type) (Business Defined)**
  This element identifies the type of payload within the request. This element is defined as a complex type to address .Net and Java interoperability.

  Value: Is the actual value of the payload type.

  **Java Example DTSRequestPayloadType SOAP Element**
  ```xml
  <ns2:DTSRequestPayloadType soapenv:mustUnderstand="0" xsi:type="ns2:DTSRequestPayloadType"
  xmlns:ns2="urn:org:pesc:datatransport">
  <ns2:value> CRC01Request </ns2:value>
  </ns2:DTSRequestPayloadType>
  ```

  **.Net Example DTSRequestPayloadType SOAP Element**
  ```xml
  <DTSRequestPayloadType xsi:type="DTSRequestPayloadType" xmlns="urn:org:pesc:datatransport">
  <value>CRC01Request</value>
  </DTSRequestPayloadType>
  ```

• **DTSRequestServiceExpectation Element (Complex Type) (Business Defined)**
  This element is to be used to identify how the transaction should be processed. This element is defined as a complex type to address .Net and Java interoperability.

  Value: Is the actual value of the service expectation.

  **Java Example DTSRequestServiceExpectation SOAP Element**
  ```xml
  <ns3:DTSRequestServiceExpectation soapenv:mustUnderstand="0" xsi:type="ns3:DTSRequestServiceExpectation"
  xmlns:ns3="urn:org:pesc:datatransport">
  <ns3:value>Immediate </ns3:value>
  </ns3:DTSRequestServiceExpectation>
  ```

  **.Net Example DTSRequestServiceExpectation SOAP Element**
  ```xml
  <DTSRequestServiceExpectation xsi:type="DTSRequestServiceExpectation" xmlns="urn:org:pesc:datatransport">
  <value>Immediate</value>
  </DTSRequestServiceExpectation>
  ```
• DTSRequestSignature Element (Complex Type) (Specification Defined)

This element is to be used to hold the digital signature of the compressed and encoded request payload. The data in this element must be base64 encoded. This element is defined as a complex type to address .Net and Java interoperability.

Value: Is the actual value of the Base64 encoded digital signature.

```
Java Example DTSRequestSignature SOAP Element
<ns4:DTSRequestSignature soapenv:mustUnderstand="0"
  xsi:type="ns4:DTSRequestSignature"
  xmlns:ns4="urn:org:pesc:datatransport">
  <ns4:value>
    GFvcLUUB6yUFesaOAwN76OTMgH0cEd6ET7XDy/X7iDu/zyxcCeJh32S561DH+c0zifsSnSNRbsDirQUCmqH1lfZmJACE1X
    ukZGwwQB98yBav8epOkenHyM/NnwldCTFIl32p1YFWirJ+S1B/o/1of3feCahZDjQzqtSpZGbGkE=
  </ns4:value>
</ns4:DTSRequestSignature>
```

```
.Net Example DTSRequestSignature SOAP Element
<DTSRequestSignature xsi:type="DTSRequestSignature" xmlns="urn:org:pesc:datatransport">
  <value>
    hHijrVgkCNV1lCdySeddy2eLxCL9/v8RjxKh1atAQ7lH/AuTyWnNjJ01C0qFdljy8XJDemgXTvFideh7x81ldzRAzr8IUVL1cbUf59z
    kZbUNaPnl108pLyrO666vI+EV5wMeUweRvU4XLfZbdW3qb551Ua+Vac/F55E=
  </value>
</DTSRequestSignature>
```

• DTSRequestPayloadBytes Element (Complex Type) (Specification Defined)

This element holds the decompressed byte count of the request payload. Every DTS Service will have system resource constraints (memory, disk space, etc.) that should to be taken into consideration when attempting to decompress the request payload. The byte count is required to provide a reliable method to assist the DTS Service in determining the best method to decompress the request payload. This element is defined as a complex type to address .Net and Java interoperability.

Value: Is the actual value the decompressed byte count of the request payload.

```
Java Example DTSRequestPayloadBytes SOAP Element
<ns5:DTSRequestPayloadBytes soapenv:mustUnderstand="0"
  xsi:type="ns5:DTSRequestPayloadBytes"
  xmlns:ns5="urn:org:pesc:datatransport">
  <ns5:value>32</ns5:value>
</ns5:DTSRequestPayloadBytes>
```

```
.Net Example DTSRequestPayloadBytes SOAP Element
<DTSRequestPayloadBytes xsi:type="DTSRequestPayloadBytes" xmlns="urn:org:pesc:datatransport">
  <value>32</value>
</DTSRequestPayloadBytes>
```

Body

• DTSRequest (Type: String) (Business Defined)

This is the payload of the transaction. The data in this element must be zlib compressed and base64 encoded.

```
Java Example submitDTSRequest SOAP Element
<soapenv:Body>
  <DTSRequest xmlns="urn:org:pesc:datatransport">
    eJwLSk1LLUrNS05V8MwtyEnNTc1LLMnMz1MISKzMyU9MUQQAyZlMDQ==
  </DTSRequest>
</soapenv:Body>
```
<soapenv:Body>

.Net Example submitDTSRequest SOAP Element
<soap:Body>
  <DTSRequest xmlns="urn:org:pesc:datatransport">eJwLzwtyElVCEktLgEAGNYEkw==</DTSRequest>
</soap:Body>
4.3 **DTS Response SOAP (See the supplemental section of this document for example SOAP)**

**Header Elements**

- **DTSResponseRouting Element (Complex Type) (Business Defined)**
  This element is to be used to identify routing information for the response.
  
  ```
  <ns1:DTSResponseRouting soapenv:mustUnderstand="0"
  xsi:type="ns1:DTSResponseRouting"
  xmlns:ns1="urn:org:pesc:datatransport">
    <ns1:UUID>99d3c8ee-289a-4c17-b221-0e96155589e2</ns1:UUID>
    <ns1:recipientID>A</ns1:recipientID>
    <ns1:recipientIDSubCode>Chitty</ns1:recipientIDSubCode>
    <ns1:sourceID>B</ns1:sourceID>
    <ns1:sourceIDSubCode>Malinoski</ns1:sourceIDSubCode>
    <ns1:transmissionDateTime>2006-10-23T22:33:26.653Z</ns1:transmissionDateTime>
  </ns1:DTSResponseRouting>
  ```

  ```
  <DTSResponseRouting xsi:type="DTSResponseRouting" xmlns="urn:org:pesc:datatransport">
    <UUID>99d3c8ee-289a-4c17-b221-0e96155589e2</UUID>
    <transmissionDateTime>2006-10-23T22:33:26.653Z</transmissionDateTime>
    <sourceID>B</sourceID>
    <sourceIDSubCode>Malinoski</sourceIDSubCode>
    <recipientID>A</recipientID>
    <recipientIDSubCode>Chitty</recipientIDSubCode>
  </DTSResponseRouting>
  ```

- **DTSResponsePayloadType Element (Complex Type) (Business Defined)**
  This element is to be used to identify the type of payload within the response. This element is defined as a complex type to address .Net and Java interoperability.
Value: Is the actual value of the payload type.

Java Example DTSResponsePayloadType SOAP Element
<ns2:DTSResponsePayloadType soapenv:mustUnderstand="0" xsi:type="ns2:DTSResponsePayloadType" xmlns:ns2="urn:org:pesc:datatransport">
  <ns2:value>CRC01 Response</ns2:value>
</ns2:DTSResponsePayloadType>

.Net Example DTSResponsePayloadType SOAP Element
<DTSResponsePayloadType xsi:type="DTSResponsePayloadType" xmlns="urn:org:pesc:datatransport">
  <value>CRC01 Response</value>
</DTSResponsePayloadType>
• **DTSResponseAcknowledge Element (Complex Type) (Business Defined)**
This element is used to identify how the Service handled or will handle the transaction. This element is defined as a complex type to address .Net and Java interoperability.

Value: Is the actual value of the Response Acknowledge.

Java Example DTSResponseAcknowledge SOAP Element
```
<ns3:DTSResponseAcknowledge soapenv:mustUnderstand="0" xsi:type="ns3:DTSResponseAcknowledge"
  xmlns:ns3="urn:org:pesc:datatransport">  
  <ns3:value>immediate</ns3:value>
</ns3:DTSResponseAcknowledge>
```

.Net Example DTSResponseAcknowledge SOAP Element
```
<DTSResponseAcknowledge xsi:type="DTSResponseAcknowledge" xmlns="urn:org:pesc:datatransport">
  <value>immediate</value>
</DTSResponseAcknowledge>
```

• **DTSResponseSignature Element (Complex Type) (Specification Defined)**
This element is to be used to hold the digital signature of the compressed and encoded response payload. The data in this element must be base64 encoded. This element is defined as a complex type to address .Net and Java interoperability.

Value: Is the actual value of the Base64 encoded digital signature.

Java Example DTSResponseSignature SOAP Element
```
<ns4:DTSResponseSignature soapenv:mustUnderstand="0" xsi:type="ns4:DTSResponseSignature"
  xmlns:ns4="urn:org:pesc:datatransport">  
  <ns4:value>
    WvpTbXxyHRvY5B4GfnE+TV8J1eAjpZv1T2Me0cNMM39TUJnsQkmPWWOzgCIv1Pv1h7bic1LsdFKEKQ/15Ak
    Wv4lizX9LYkTnV9EEej4ildY5rLvgCz9OkyXZ10kX00kpnCnkTeVNkJQMPy+i=qK5ZUsT0UI5/CFjQOOtLm4OhKk=
  </ns4:value>
</ns4:DTSResponseSignature>
```

.Net Example DTSResponseSignature SOAP Element
```
<DTSResponseSignature xsi:type="DTSResponseSignature" xmlns="urn:org:pesc:datatransport">
  <value>
    i1uVrl4X2bSVFteVdAfA2ZV1KhM0cc23PKCRN+e/bj+i=sZz3cSvn7K6u3GhX2qsJ8XI10yX65LXcw/Htse/6
    DGPuYHeSvqCv0dl4aoAZEXWoepMTTiv6OU3Yghd9kDwNaaVPJmn3nPYj4X7+w/h5C1OHvcbYDQ1Vjg1JE=
  </value>
</DTSResponseSignature>
```

• **DTSResponsePayloadBytes Element (Complex Type) (Specification Defined)**
This element holds the decompressed byte count of the response payload. Every DTS Client will have system resource constraints (memory, disk space, etc.) that should be taken into consideration when attempting to decompress the request payload. The byte count is required to provide a reliable method to assist the DTS Client in determining the best method to decompress the response payload. This element is defined as a complex type to address .Net and Java interoperability.

Value: Is the actual value the decompressed byte count of the response payload.

Java Example DTSResponsePayloadBytes SOAP Element
```
<ns5:DTSResponsePayloadBytes soapenv:mustUnderstand="0">
</ns5:DTSResponsePayloadBytes SOAP Element
```


<value>32</value>
</ns5:DTSRequestPayloadBytes>

.Net Example DTSRequestSignature SOAP Element

<?xml version="1.0"?>
<ns5:DTSRequestSignature xsi:type="ns5:DTSRequestSignature" xmlns:ns5="urn:org:pesc:datatransport">
  <ns5:value>32</ns5:value>
</ns5:DTSRequestSignature>
Body

- **DTSResponse (Type: String) (Business Defined)**
  This is the response of the transaction. The data in this element must be zlib compressed and base64 encoded.

Java Example submitDTSResult SOAP Element

```
<soapenv:Body>
  <DTSResponse xmlns="urn:org:pesc:datatransport">
    eJxzCQlWCEpNSy1KzUIOVIDMLchJzU3NK0ksyczPA0oUF+TnFacCAP0rD/A=
  </DTSResponse>
</soapenv:Body>
```

.Net Example submitDTSResult SOAP Element

```
<soap:Body>
  <DTSResponse xmlns="urn:org:pesc:datatransport">
    eJwLTi0qy0xOdc7PLShKLS5OTdExstA1hgiGsiEJGxrChMN5izLTKn2DdYxNjAHxPBa4
  </DTSResponse>
</soap:Body>
```
5 COMMUNICATION OR FRAMEWORK ERRORS GENERATED OUTSIDE DTS

The following is a non-inclusive list of possible external errors that should be given consideration in conjunction with a DTS Implementation:

TCP/IP Specification Errors

HTTP/S Specification Errors
   e.g. 400/500/505

Web Service errors generated natively by the Java or .Net Toolkits
   e.g. De-serialization Errors for child elements of a SOAP header element missing.
6 DTS ERROR HANDLING REQUIREMENTS:

Each component of a DTS enabled application must provide error handling and report exceptions to the appropriate component. See the reference implementation guide for example components.

6.1 DTS Service Error Handling Table

The table below defines the three categories that can produce an error.

<table>
<thead>
<tr>
<th>Element or Condition</th>
<th>Element is Missing</th>
<th>Element Exists but Length is Zero</th>
<th>Processing Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Routing Header Aggregate</td>
<td>DTS.HEADER.001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source ID</td>
<td>DTS.HEADER.001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recipient ID</td>
<td>DTS.HEADER.001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UUID</td>
<td>DTS.HEADER.001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transmission DateTime</td>
<td>DTS.HEADER.001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PayloadType</td>
<td>DTS.HEADER.002</td>
<td>DTS.HEADER.002</td>
<td></td>
</tr>
<tr>
<td>Request Service Expectation</td>
<td>DTS.HEADER.003</td>
<td>DTS.HEADER.003</td>
<td></td>
</tr>
<tr>
<td>Request Signature</td>
<td>DTS.HEADER.004</td>
<td>DTS.HEADER.004</td>
<td></td>
</tr>
<tr>
<td>Request PayloadBytes</td>
<td>DTS.HEADER.005</td>
<td>DTS.HEADER.005</td>
<td></td>
</tr>
<tr>
<td>Compressed, encoded Payload</td>
<td></td>
<td>DTS.PAYLOAD.001</td>
<td></td>
</tr>
<tr>
<td>Payload cannot be decoded</td>
<td></td>
<td>DTS.PAYLOAD.002</td>
<td></td>
</tr>
<tr>
<td>Payload cannot be decompressed</td>
<td></td>
<td>DTS.PAYLOAD.003</td>
<td></td>
</tr>
<tr>
<td>Payload too large to decompress</td>
<td></td>
<td>DTS.PAYLOAD.004</td>
<td></td>
</tr>
<tr>
<td>No public key avail or cannot be found</td>
<td></td>
<td>DTS.SECURITY.001</td>
<td></td>
</tr>
<tr>
<td>Could not validate digital signature</td>
<td></td>
<td>DTS.SECURITY.002</td>
<td></td>
</tr>
<tr>
<td>Application unable to process transaction</td>
<td></td>
<td></td>
<td>DTS.APPLICATION</td>
</tr>
</tbody>
</table>

6.2 Faults Summary

Detailed fault descriptions are provided below:
Detail Error Conditions Reported to DTS Client via SOAP Fault

- Error handling for DTSRequestRouting Element

  Condition 1 - DTS Request Routing Header Element is null (not found in SOAP)
  
  Fault Code: DTS.HEADER.001
  Fault String: DTS SOAP Header Error with DTS Request Routing Information.
  Fault Detail: DTS Request Routing Information is not set in the SOAP Packet

  Conditions 2 through 5 Fault Details should be combined if more than 1 of the error conditions exists.

  Condition 2 - Length of Source Id = 0
  
  Fault Code: DTS.HEADER.001
  Fault String: DTS SOAP Header Error with DTS Request Routing Information.
  Fault Detail: DTS Request Routing Source Id value is missing.

  Condition 3 - Length of Recipient Id = 0
  
  Fault Code: DTS.HEADER.001
  Fault String: DTS SOAP Header Error with DTS Request Routing Information.
  Fault Detail: DTS Request Routing Recipient Id value is missing.

  Condition 4 - Length of UUID = 0
  
  Fault Code: DTS.HEADER.001
  Fault String: DTS SOAP Header Error with DTS Request Routing Information.
  Fault Detail: DTS Request Routing UUID value is missing.

  Condition 5 - Length of Transmission Date Time = 0
  
  Fault Code: DTS.HEADER.001
  Fault String: DTS SOAP Header Error with DTS Request Routing Information.
  Fault Detail: DTS Request Transmission Date Time value is missing.

- Error handling for DTSRequestPayloadType Element

  Condition 1 - Request Payload Type Element is null (not found in SOAP)
  
  Fault Code: DTS.HEADER.002
  Fault String: DTS SOAP Header Error with DTS Request Payload Type Information
  Fault Detail: DTS Request Payload Type Information is not set in the SOAP Packet
Condition 2 - Length of value = 0
Fault Code: DTS.HEADER.002
Fault String: DTS SOAP Header Error with DTS Request Payload Type Information
Fault Detail: DTS Request Payload Type value is missing.

- Error handling for DTSRequestServiceExpectation Element

Condition 1 - Request Service Expectation Element is null (not found in SOAP)
Fault Code: DTS.HEADER.003
Fault String: DTS SOAP Header Error with DTS Request Service Expectation Information
Fault Detail: DTS Request Service Expectation Information is not set in the SOAP Packet

Condition 2 - Length of value = 0
Fault Code: DTS.HEADER.003
Fault String: DTS SOAP Header Error with DTS Request Service Expectation Information
Fault Detail: DTS Request Service Expectation value is missing
• Error handling for DTSRequestSignature Element

  Condition 1 - Request Signature Element is null (not found in SOAP)
  Fault Code:          DTS.HEADER.004
  Fault String:        DTS SOAP Header Error with DTS Request Signature Information
  Fault Detail:        DTS Request Signature Information is not set in the SOAP Packet

  Condition 2 - Length of value = 0
  Fault Code:          DTS.HEADER.004
  Fault String:        DTS SOAP Header Error with DTS Request Signature Information
  Fault Detail:        DTS Request Signature value is missing

• Error handling for DTSRequestPayloadBytes Element

  Condition 1 - Request Payload Bytes Element is null (not found in SOAP)
  Fault Code:          DTS.HEADER.005
  Fault String:        DTS SOAP Header Error with DTS Request Payload Bytes Information
  Fault Detail:        DTS Request Payload Bytes Information is not set in the SOAP Packet

  Condition 2 - Length of value = 0
  Fault Code:          DTS.HEADER.005
  Fault String:        DTS SOAP Header Error with DTS Request Payload Bytes Information
  Fault Detail:        DTS Request Payload Bytes value is missing

• Error handling for submitDTSRequest

  Condition 1 - Length of Compressed and Encoded Payload = 0
  Fault Code:          DTS.PAYLOAD.001
  Fault String:        DTS Payload Error
  Fault Detail:        DTS Payload is missing.

  Note: This is a valid check since compressing and encoding a zero byte string will always result in a non-zero byte length string.

  Condition 2 - Payload cannot be decoded
  Fault Code:          DTS.PAYLOAD.002
  Fault String:        DTS Payload Error
  Fault Detail:        DTS Payload decoding error.
Condition 3 - Payload cannot be decompressed
Fault Code: DTS.PAYLOAD.003
Fault String: DTS Payload Error
Fault Detail: DTS Payload decompress error.

Condition 4 - Payload too large
Fault Code: DTS.PAYLOAD.004
Fault String: DTS Payload Error
Fault Detail: DTS Payload could not be decompressed because of size limitations.

- Error handling for DTS Service Security

Condition 1 - Could not find public key for source.
Fault Code: DTS.SECURITY.001
Fault String: DTS Security Error
Fault Detail: Could not find public key for Request Routing Source Id.

Condition 2 - Could not validate digital signature.
Fault Code: DTS.SECURITY.002
Fault String: DTS Security Error
Fault Detail: Could not validate digital signature.

Service Backend Processing Error Handling
If the backend processing of a DTS Service cannot be invoked or encounters errors during processing the error conditions must be reported to the DTS Client via the following SOAP fault:

Any Condition
Fault Code: DTS.APPLICATION
Fault String: Backend Processing Error
Fault Detail: Provided by the DTS Service Application.

Example DTSRequestRouting Fault SOAP Element

```xml
<soapenv:Fault>
  <faultcode>soapenv:DTS.HEADER.001</faultcode>
  <faultstring>DTS SOAP Header Error with DTS Routing Information</faultstring>
  <detail>
    <string>DTS Request Routing Source Id value is missing.</string>
  </detail>
  <detail>
    <string>DTS Request Routing UUID value is missing.</string>
  </detail>
</soapenv:Fault>
```
6.3 **DTS Client Error Handling**

The DTS SOAP response must meet the following requirements:

- **DTSResponseRouting Element is not null (must be present in SOAP)**
  - Length of SourceId must be > 0
  - Length of RecipientID must be > 0
  - Length of UUID must be > 0
  - Length of Transmission Date Time must be > 0

- **DTSResponsePayloadType Element is not null (must be present in SOAP)**
  - Length of value > 0

- **DTSResponseAcknowledge Element is not null (must be present in SOAP)**
  - Length of value > 0

- **DTSResponseSignature Element is not null (must be present in SOAP)**
  - Length of value > 0

- **DTSResponsePayloadBytes Element is not null (must be present in SOAP)**
  - Length of value > 0

If the DTS SOAP response violates any of the above requirements, action taken by the DTS client will be implementation specific for each receiving entity.
6.4 SUPPLEMENTAL DOCUMENTATION

This section contains additional documentation to assist entities to understand the specification. It includes the DTS Web Service Definition Language (WSDL) which describes the web service that is created using the specification and example SOAP Request and Response packets for both .Net and JAVA.

DTS WSDL 1.0

```xml
  <types>
    <xsd:schema targetNamespace="urn:org:pesc:datatransport">
      <xsd:complexType name="DTSRoutingElementsType">
        <xsd:sequence>
          <xsd:element name="DTSUUID" minOccurs="1" maxOccurs="1" type="xsd:string"/>
          <xsd:element name="DTSTransmissionDateTime" minOccurs="1" maxOccurs="1" type="xsd:string"/>
          <xsd:element name="DTSSourceID" minOccurs="1" maxOccurs="1" type="xsd:string"/>
          <xsd:element name="DTSSourceSubCode" minOccurs="1" maxOccurs="1" type="xsd:string"/>
          <xsd:element name="DTSRecipientID" minOccurs="1" maxOccurs="1" type="xsd:string"/>
          <xsd:element name="DTSRecipientSubCode" minOccurs="1" maxOccurs="1" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>

      <xsd:complexType name="DTSServiceExpectationElementsType">
        <xsd:sequence>
          <xsd:element name="value" minOccurs="1" maxOccurs="1" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>

      <xsd:complexType name="DTSPayloadElementsType">
        <xsd:sequence>
          <xsd:element name="value" minOccurs="1" maxOccurs="1" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>

      <xsd:complexType name="DTSAcknowledgeElementsType">
        <xsd:sequence>
          <xsd:element name="value" minOccurs="1" maxOccurs="1" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>

      <xsd:complexType name="DTSSignatureElementsType">
        <xsd:sequence>
          <xsd:element name="value" minOccurs="1" maxOccurs="1" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>

      <xsd:complexType name="DTSSizeType">
        <xsd:sequence>
          <xsd:element name="value" minOccurs="1" maxOccurs="1" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>

      <xsd:element name="DTSRequestPayloadType" type="tns:DTSPayloadElementsType"/>
      <xsd:element name="DTSRequestRouting" type="tns:DTSRoutingElementsType"/>
      <xsd:element name="DTSRequestServiceExpectation" type="tns:DTSServiceExpectationElementsType"/>
      <xsd:element name="DTSRequestSignature" type="tns:DTSSignatureElementsType"/>
      <xsd:element name="DTSRequestPayloadBytes" type="tns:DTSSizeType"/>
      <xsd:element name="DTSResponsePayloadType" type="tns:DTSPayloadElementsType"/>
      <xsd:element name="DTSResponseRouting" type="tns:DTSRoutingElementsType"/>
      <xsd:element name="DTSResponsePayloadBytes" type="tns:DTSSizeType"/>
      <xsd:element name="DTSResponseAcknowledge" type="tns:DTSAcknowledgeElementsType"/>
    </xsd:schema>
  </types>
</definitions>
```
<xsd:element name="DTSResponseSignature" type="tns:DTSSignatureElementsType"/>
<xsd:element name="DTSResponsePayloadBytes" type="tns:DTSSizeType"/>
<xsd:element name="Request" type="xsd:string"/>
<xsd:element name="Response" type="xsd:string"/>
</xsd:schema>
</types>
<message name="DTSRequest">
    <part name="DTSRequest" element="tns:Request"/>
</message>
<message name="DTSResponse">
    <part name="DTSResponse" element="tns:Response"/>
</message>
<message name="DTSHeaders">
    <part name="DTSRequestRouting" element="tns:DTSRequestRouting"/>
    <part name="DTSResponseRouting" element="tns:DTSResponseRouting"/>
    <part name="DTSRequestServiceExpectation" element="tns:DTSRequestServiceExpectation"/>
    <part name="DTSRequestPayloadType" element="tns:DTSRequestPayloadType"/>
    <part name="DTSRequestPayloadBytes" element="tns:DTSRequestPayloadBytes"/>
    <part name="DTSResponsePayloadType" element="tns:DTSResponsePayloadType"/>
    <part name="DTSResponsePayloadBytes" element="tns:DTSResponsePayloadBytes"/>
    <part name="DTSResponseSignature" element="tns:DTSResponseSignature"/>
    <part name="DTSResponsePayloadBytes" element="tns:DTSResponsePayloadBytes"/>
</message>

<portType name="submitDTS">
    <operation name="submitDTS">
        <input message="tns:DTSRequest"/>
        <output message="tns:DTSResponse"/>
    </operation>
</portType>

<binding name="submitDTS" type="tns:submitDTS">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="submitDTS">
        <soap:operation soapAction="http://www.datatransportstandard.com/submitDTS"/>
        <input>
            <soap:body use="literal"/>
            <soap:header message="tns:DTSHeaders" part="DTSRequestSignature" use="literal"/>
            <soap:header message="tns:DTSHeaders" part="DTSRequestRouting" use="literal"/>
            <soap:header message="tns:DTSHeaders" part="DTSRequestServiceExpectation" use="literal"/>
            <soap:header message="tns:DTSHeaders" part="DTSRequestPayloadType" use="literal"/>
            <soap:header message="tns:DTSHeaders" part="DTSRequestPayloadBytes" use="literal"/>
        </input>
        <output>
            <soap:body use="literal"/>
            <soap:header message="tns:DTSHeaders" part="DTSResponseSignature" use="literal"/>
            <soap:header message="tns:DTSHeaders" part="DTSResponseRouting" use="literal"/>
            <soap:header message="tns:DTSHeaders" part="DTSResponsePayloadType" use="literal"/>
            <soap:header message="tns:DTSHeaders" part="DTSResponsePayloadBytes" use="literal"/>
        </output>
    </operation>
</binding>

<service name="submitDTS">
    <port name="submitDTS" binding="tns:submitDTS">
        <soap:address location="http://localhost:8080/services/ReferenceImplementation/submitDTS"/>
    </port>
</service>
</definitions>
Entire Java DTS SOAP Request:

```xml
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header>
    <ns1:DTSRequestRouting soapenv:mustUnderstand="0"
      xsi:type="ns1:DTSRequestRouting"
      xmlns:ns1="urn:org:pesc:datatransport">
      <ns1:UUID>99d3c8ee-289a-4c17-b221-0e96155589e2</ns1:UUID>
      <ns1:recipientID>A</ns1:recipientID>
      <ns1:recipientIDSubCode>Chitty</ns1:recipientIDSubCode>
      <ns1:sourceID>B</ns1:sourceID>
      <ns1:sourceIDSubCode>Malinoski</ns1:sourceIDSubCode>
      <ns1:transmissionDateTime>2006-10-23T22:33:26.654Z</ns1:transmissionDateTime>
    </ns1:DTSRequestRouting>
    <ns2:DTSRequestPayloadType soapenv:mustUnderstand="0"
      xsi:type="ns2:DTSRequestPayloadType"
      xmlns:ns2="urn:org:pesc:datatransport">
      <ns2:value>CR:C AppSend</ns2:value>
    </ns2:DTSRequestPayloadType>
    <ns3:DTSRequestServiceExpectation soapenv:mustUnderstand="0"
      xsi:type="ns3:DTSRequestServiceExpectation"
      xmlns:ns3="urn:org:pesc:datatransport">
      <ns3:value>Immediate</ns3:value>
    </ns3:DTSRequestServiceExpectation>
    <ns4:DTSRequestSignature soapenv:mustUnderstand="0"
      xsi:type="ns4:DTSRequestSignature"
      xmlns:ns4="urn:org:pesc:datatransport">
      GFvLCUUB6yUFJeaOAwN76OTmgH0cEd6ET7XDy/XjIlDu/zyx6CeJh32S561DHcOzflSnSRTqDirUQTcMcMg1hfZmJACE1XukZGwwQB98yBav8epOkenhym/NrwlCTFlI3r2p1YFWir+j/S1b/w/o/1of3feDahZDjQzqtSpZGbKE=
    </ns4:DTSRequestSignature>
    <ns5:DTSRequestPayloadBytes soapenv:mustUnderstand="0"
      xsi:type="ns5:DTSRequestPayloadBytes"
      xmlns:ns5="urn:org:pesc:datatransport">
      32
    </ns5:DTSRequestPayloadBytes>
  </soapenv:Header>
  <soapenv:Body>
    <DTSRequest xmlns="urn:org:pesc:datatransport">
      eJwLSk1LLLrN505VMytEnNtc1LLmMz1M1SKzMy9U9MQQAYZIMDO==
    </DTSRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

Entire Java DTS SOAP Response:

```xml
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header>
    <ns1:DTSResponseRouting soapenv:mustUnderstand="0"
      xsi:type="ns1:DTSResponseRouting"
      xmlns:ns1="urn:org:pesc:datatransport">
      <ns1:UUID>99d3c8ee-289a-4c17-b221-0e96155589e2</ns1:UUID>
      <ns1:recipientID>B</ns1:recipientID>
      <ns1:recipientIDSubCode>Malinoski</ns1:recipientIDSubCode>
      <ns1:sourceID>A</ns1:sourceID>
      <ns1:sourceIDSubCode>Chitty</ns1:sourceIDSubCode>
      <ns1:transmissionDateTime>2006-10-23T22:33:26.654Z</ns1:transmissionDateTime>
    </ns1:DTSResponseRouting>
    <ns2:DTSResponsePayloadType soapenv:mustUnderstand="0"
      xsi:type="ns2:DTSResponsePayloadType"
      xmlns:ns2="urn:org:pesc:datatransport">
      </ns2:DTSResponsePayloadType>
    <ns3:DTSResponseServiceExpectation soapenv:mustUnderstand="0"
      xsi:type="ns3:DTSResponseServiceExpectation"
      xmlns:ns3="urn:org:pesc:datatransport">
      Immediate
    </ns3:DTSResponseServiceExpectation>
    <ns4:DTSResponseSignature soapenv:mustUnderstand="0"
      xsi:type="ns4:DTSResponseSignature"
      xmlns:ns4="urn:org:pesc:datatransport">
      GFvLCUUB6yUFJeaOAwN76OTmgH0cEd6ET7XDy/XjIlDu/zyx6CeJh32S561DHcOzflSnSRTqDirUQTcMcMg1hfZmJACE1XukZGwwQB98yBav8epOkenhym/NrwlCTFlI3r2p1YFWir+j/S1b/w/o/1of3feDahZDjQzqtSpZGbKE=
    </ns4:DTSResponseSignature>
    <ns5:DTSResponsePayloadBytes soapenv:mustUnderstand="0"
      xsi:type="ns5:DTSResponsePayloadBytes"
      xmlns:ns5="urn:org:pesc:datatransport">
      32
    </ns5:DTSResponsePayloadBytes>
  </soapenv:Header>
  <soapenv:Body>
    <DTSResponse xmlns="urn:org:pesc:datatransport">
      eJwLSk1LLLrN505VMytEnNtc1LLmMz1M1SKzMy9U9MQQAYZIMDO==
    </DTSResponse>
  </soapenv:Body>
</soapenv:Envelope>
```
<soapenv:Envelope xmlns:ns2="urn:org:pesc:datatransport">
  <ns2:value>CR:C Response</ns2:value>
</ns2:DTSResponsePayloadType>
<ns3:DTSResponseAcknowledge soapenv:mustUnderstand="0" xsi:type="ns3:DTSResponseAcknowledge" xmlns:ns3="urn:org:pesc:datatransport">
  <ns3:value>Deferred</ns3:value>
</ns3:DTSResponseAcknowledge>
<ns4:DTSResponseSignature soapenv:mustUnderstand="0" xsi:type="ns4:DTSResponseSignature" xmlns:ns4="urn:org:pesc:datatransport">
  <ns4:value>IDfMApmOuWL93MjbaQr3Ak5gf0z6RQsGQqe77R0p9J1wh4WBxEx1A9EVtqGkynzwyNGJipVRN1DXJn3srz4fEtml6WLhvLsHLu097jXym2SFEDMKX6zKBFvijjEB9Y5V5CCYXumZdKAYnAq9eWwVN+4BV1f/F07bFwwuT5Xc=</ns4:value>
</ns4:DTSResponseSignature>
<ns5:DTSResponsePayloadBytes soapenv:mustUnderstand="0" xsi:type="ns5:DTSResponsePayloadBytes" xmlns:ns5="urn:org:pesc:datatransport">
  <ns5:value>63</ns5:value>
</ns5:DTSResponsePayloadBytes>
</soapenv:Header>
<soapenv:Body>
  <DTSResponse xmlns="urn:org:pesc:datatransport">
    eJwLTi0qy0xOdcc7PLShKLS5OTdExMdAJhgiG5iEJGxvBhMNSizLTKn2DdcwtdADwhBa2
  </DTSResponse>
</soapenv:Body>
</soapenv:Envelope>
Entire .NET DTS SOAP Request:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Header>
    <DTSRequestPayloadBytes xsi:type="DTSRequestPayloadBytes" xmlns="urn:org:pesc:datatransport">
      <value>11</value>
    </DTSRequestPayloadBytes>
    <DTSRequestPayloadType xsi:type="DTSRequestPayloadType" xmlns="urn:org:pesc:datatransport">
      <value>CRC01Request</value>
    </DTSRequestPayloadType>
    <DTSRequestRouting xsi:type="DTSRequestRouting" xmlns="urn:org:pesc:datatransport">
      <UUID>99d3c8ee-289a-4c17-b221-0e96155589e2</UUID>
      <transmissionDateTime>2006-10-23T22:33:26.654Z</transmissionDateTime>
      <sourceID>A</sourceID>
      <sourceIDSubCode>Chitty</sourceIDSubCode>
      <recipientID>B</recipientID>
      <recipientIDSubCode>Malinoski</recipientIDSubCode>
    </DTSRequestRouting>
    <DTSRequestServiceExpectation xsi:type="DTSRequestServiceExpectation" xmlns="urn:org:pesc:datatransport">
      <value>Immediate</value>
    </DTSRequestServiceExpectation>
    <DTSRequestSignature xsi:type="DTSRequestSignature" xmlns="urn:org:pesc:datatransport">
      <value>ucukbXZjGBEJ5pQUJw844XbMuMxvSuUX4BCMGHRrZ1LMrk8qRGYmGmG3UaIDOEHeordZaVv29beDYK2UPu6TFQFO6NLGg3NWNPUsUm64Gq9ueuvdYlgkmpKWmFACv4gASUlyqQDQ8yfhwYcMEsRGcwpSSWPL4=</value>
    </DTSRequestSignature>
  </soap:Header>
  <soap:Body>
    <DTSRequest xmlns="urn:org:pesc:datatransport">eJzLSM3JyVcoyUgtSgUAGcYETQ==</DTSRequest>
  </soap:Body>
</soap:Envelope>
```

Entire .NET DTS SOAP Response:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Header>
    <DTSResponseAcknowledge xsi:type="DTSResponseAcknowledge" xmlns="urn:org:pesc:datatransport">
      <value>immediate|deferred|received</value>
    </DTSResponseAcknowledge>
    <DTSResponsePayloadBytes xsi:type="DTSResponsePayloadBytes" xmlns="urn:org:pesc:datatransport">
      <value>63</value>
    </DTSResponsePayloadBytes>
    <DTSResponsePayloadType xsi:type="DTSResponsePayloadType" xmlns="urn:org:pesc:datatransport">
      <value>Returned</value>
    </DTSResponsePayloadType>
    <DTSResponseRouting xsi:type="DTSResponseRouting" xmlns="urn:org:pesc:datatransport">
      <UUID>99d3c8ee-289a-4c17-b221-0e96155589e2</UUID>
      <transmissionDateTime>2006-10-23T22:33:26.654Z</transmissionDateTime>
      <sourceID>B</sourceID>
      <sourceIDSubCode>Malinoski</sourceIDSubCode>
      <recipientID>A</recipientID>
      <recipientIDSubCode>Chitty</recipientIDSubCode>
    </DTSResponseRouting>
    <DTSResponseSignature xsi:type="DTSResponseSignature" xmlns="urn:org:pesc:datatransport">
      <value>wnPdRP8m3haSrvcqhCrqbd+tMWssUZaQNYa+p3p7Ji/2soqJj21oZsgDAvLYLdWQPvFccqjAnhuVR3ALQamR4f6AysK+jVWplT0LH+V5YRNm3wKJ77WnpvJ7Y1uyYLYgsbubyYZ6+IDw86F5bcbYGB8IKRC6zzMgqUe11TBSU=</value>
    </DTSResponseSignature>
  </soap:Header>
  <soap:Body>
    <DTSResponse xmlns="urn:org:pesc:datatransport">
```

7 Recommendations

It is the recommendation of the PESC DTS Technical Committee that the UUID sub element of the DTSRequestRouting and DTSResponseRouting header elements be formatted according to the IETF Draft Uuid Specification. Supporting documentation can be found at the following link: http://en.wikipedia.org/wiki/UUID.

It is the recommendation of the PESC DTS Technical Committee that the uncompressed source data not exceed 50 megabytes. Technology will allow for data sizes greater than 50 meg, but this requires modifications to the implementation of the specification and web server configuration parameters.

The error handling section of the specification does not require values being present in the DTSRequestRouting SourceIDSubCode, DTSRequestRouting, RecipientIdSubCode, DTSResponseRouting, SourceIDSubCode, and DTSResponseRouting, RecipientIDSubCode header sub-elements. It is the recommendation of the PESC DTS Technical Committee that the IDSubCode sub elements be populated with the assigning organization of the SourceId and RecipientId codes respectively.

For example: SourceId = 12345678, SourceIDSubCode = ABCDE

It is recognized entities may have internal uses of the specification. When used internally entities may chose to not implement the security portions of this specification.
Glossary

This section should list a glossary of terms and their definitions helpful in deciphering this document. – Will be provided at later date.
Acknowledgements

PESC DTS Workgroup would like to recognize the following individuals and their respective organizations for their time and effort with the development of the DTS Specification

Kim Ballard – Sallie Mae
Tim Bornholtz – FSA
Nathan Chitty – Nelnet
Laura Damkoehler – ELM
John Gill – TG
Wei Li – Sallie Mae
Mark Malinoski – AES
Barry Needalman – ASA
Gary Sandler – ELM
Zahida Shariff – V-TEK
Kim Shiflette – USA Funds
## Revision History

<table>
<thead>
<tr>
<th>DATE</th>
<th>SECTION/PAGE</th>
<th>DESCRIPTION</th>
<th>REQUESTED BY</th>
<th>MADE BY</th>
</tr>
</thead>
<tbody>
<tr>
<td>5/24/05</td>
<td>Whole Document</td>
<td>Initial Version</td>
<td>Mark Malinoski</td>
<td>Nathan Chitty</td>
</tr>
<tr>
<td>8/2/05</td>
<td>Whole Document</td>
<td>Format, grammar, and style review and changes.</td>
<td>Gary Sandler</td>
<td>Laura Damkoehler</td>
</tr>
<tr>
<td>05/19/06</td>
<td>Whole Document</td>
<td>Header and footer date changes</td>
<td>Kim Shiflette</td>
<td>Kim Shiflette</td>
</tr>
<tr>
<td>10/23/06</td>
<td>Whole Document</td>
<td>Typographical changes and example consistency corrections</td>
<td>Mark Malinoski</td>
<td>Nathan Chitty</td>
</tr>
</tbody>
</table>