presto	**PRotocole d'Echanges Standard et Ouvert** **A Technical Reference for the PRESTO protocol**

A Technical Reference for the PRESTO protocol

Version 1.0

Working Draft

Date: 2006/06/27

Abstract

The "PRotocole d'Echanges Standard et Ouvert" 1.0 (aka PRESTO) specification consists of a set a Web services specifications, along with clarifications, amendments, and restrictions of those specifications that promote interoperability. This document provides a technical reference for the mechanisms implemented in the PRESTO protocol and the schemas employed by that profile. This document is intended to be read alongside the PRESTO Guide [PRESTO-Guide] which provides a non-normative description of the overall PRESTO message exchanges model.

Status of this document

This document is version 1 the protocol based on the discussions held within the DGME-SDAE Working Group. Complementary features may be discussed in further work and a new version of the protocol may supersede the current version.

presto	**PRotocole d'Echanges Standard et Ouvert** **A Technical Reference for the PRESTO protocol**

Notice

This document is intended for architects, project leaders or any people that need to get a complete technical definition of PRESTO. People who read this document are supposed to be familiar with Web Service technology.

presto	**PRotocole d'Echanges Standard et Ouvert** **A Technical Reference for the PRESTO protocol**

Table of Contents

1. Introduction

This document defines the "PRotocole d'Echanges Standard et Ouvert" 1.0 (aka PRESTO) profile (hereafter, "Profile"), consisting of a set a Web services specifications, along with clarifications, amendments, and restrictions of those specifications that promote interoperability.

Section § 1 "Introduction" introduces the PRESTO Profile and describes its relationship to other, existing profiles.

Section § 2 "Document Conventions" describes notational conventions utilized by this Profile.

Section § 3 "Profile Conformance" explains what it means to be conformant to the Profile.

Section § 4 "PRESTO: PRotocole d'Echanges Standard et Ouvert" explains what the Profile consists in. Each subsequent subsection addresses a component of the Profile, and consists of two parts: an overview detailing the component specifications and their extensibility points, followed by subsections that address individual parts of the component specifications. Note that there is no relationship between the section numbers in this document and those in the referenced specifications.

Section § 5 "Bindings" describes how to bind the Profile with a transport.

The PRESTO Message Exchange Patterns (MEP) supported by the PRESTO protocol are covered in the PRESTO Guide [PRESTO-Guide]. It is useful to refer to the overall message exchange model described in the PRESTO Guide when reading this technical reference.

1.1. Relationships to WS-I Profiles

Due to the proliferation of differing platforms and technologies in the eGovernment space, it is essential to ensure the different PRESTO Web Service implementations are interoperable, regardless of the technology used to create them. Using WS-I Profiles as a basis for interoperable implementations ensures a baseline of standards that all implementations of a Web Service should adhere to.

More particularly, the Basic Profile specifies a minimum set of specifications that Web Services should support to ensure interoperability across diverse platforms. Current version of the Basic Profile (BP), i.e. version 1.1 (http://www.ws-i.org/Profiles/BasicProfile-

1.1.html) covers SOAP 1.1 (although SOAP 1.2 is ratified as a W3C Recommendation since 2003), XML 1.0 and HTTP 1.1 for messaging and message formats, WSDL 1.1 and XML Schema 1.0 for service description, UDDI v2 for service publication and discovery.

The PRESTO protocol in its initial version builds on the foundations of SOAP 1.2 [SOAP 1.2], WS-Addressing [WS-Addressing], MTOM [MTOM], WS-ReliableMessaging [WS-ReliableMessaging], WS-Security [WS-Security] Web service specifications.

Although the Basic Profile 1.1 does not take into account these specifications, most requirements are still applicable and directly taken into account in the above specifications. For instance, requirements relating to SOAP 1.1 are part of SOAP 1.2.

Whenever applicable, the Basic Profile 1.1 is taken as the reference for the PRESTO protocol.

Furthermore, a forthcoming 2.0 version of the Basic Profile is expected by the end of this year, which should be based upon the following specifications: SOAP 1.2, XML InfoSet, WSDL 1.1 and XML Schema 2001 for service description, WS-Addressing, MTOM, XOP, UDDI v2 & v3 for service publication and discovery. Another profile named Reliable Secure Profile (RSP) 1.0 will be focused on WS-ReliableMessaging and WS-SecureConversation.

As a consequence, some PRESTO protocol requirements present in this document are going to disappear in the future for benefit of the Basic Profile 2.0 and Reliable Secure Profile 1.0. In other words, the PRESTO protocol will simply endorse the requirements of those WS-I Profiles. As the works of the WS-I will progress towards, this Profile will be updated to reflect any necessary changes.

Same applies for the security. The PRESTO protocol leverages the message security that has advantages over transport-based security in that message security transcends the transport. In other words, the delivery transport mechanism the message uses is irrelevant because the security is applicable directly to the message. Such an approach is required by the PRESTO protocol to support routing through PRESTO relays (see related PRESTO Message Exchange Pattern in the PRESTO Guide [PRESTO-Guide]).

Similarly, the PRESTO protocol endorses the WS-I Basic Security Profile (BSP) (currently in Working Group Draft status) for message level security (http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html), which ensures that there is interoperability at the security level and that all parties understand how the messages security is applied. The basic security profile specifies the acceptable security mechanisms for Web Service communication. This includes transport layer security (SSL and TLS) as well as SOAP message security through the WS-Security [WS-Security], and including the security token types supported by this specification. There are additional profiles that specify further token types such as SAML (Security Assertion Markup Language) that the PRESTO protocol can conveys as part of the message for authorization.

As the BSP progresses towards, this Profile will be updated to reflect any necessary changes.

1.2. Relations to the WS-RAMP Profile

This document is partly based on the WS-RAMP Profile which is a similar initiative driven by IBM, Ford and DaimlerChrysler (ftp://www6.software.ibm.com/software/developer/library/ramp.pdf).

The chapters marked as (**) are extracted portions of the WS-RAMP profile and are under the following copyright.

1.2.1. WS-RAMP Copyright Notice

1.3. Profile Identification and Versioning

This document is identified by a name (in this case, PRESTO Profile) and a version number (here, 0.1). Together, they identify a particular *profile instance*.

Version numbers are composed of a major and minor portion, in the form "major.minor". They can be used to determine the precedence of a profile instance; a higher version number (considering both the major and minor components) indicates that an instance is more recent, and therefore supersedes earlier instances.

Instances of profiles with the same name (e.g., "Example Profile 1.1" and "Example Profile 5.0") address interoperability problems in the same general scope (although some developments may require the exact scope of a profile to change between instances).

One can also use this information to determine whether two instances of a profile are backwards-compatible; that is, whether one can assume that conformance to an earlier profile instance implies conformance to a later one. Profile instances with the same name and major version number (e.g., "Example Profile 1.0" and "Example Profile 1.1") MAY be considered compatible. Note that this does not imply anything about compatibility in the other direction; that is, one cannot assume that conformance with a later profile instance implies conformance to an earlier one.

2. Document Conventions

2.1. Notational Conventions

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC 2119].

Normative statements of requirements in the Profile (i.e., those impacting conformance, as outlined in "Conformance Requirements") are presented in the following manner:

Rnnnn *Statement text here*.

> where "nnnn" is replaced by a number that is unique among the requirements in the Profile, thereby forming a unique requirement identifier.

Requirement identifiers can be considered to be namespace qualified, in such a way as to be compatible with QNames from Namespaces in XML (http://www.w3.org/TR/REC-xml-

names). If there is no explicit namespace prefix on a requirement's identifier (e.g., "R9999" as opposed to "bp10:R9999"), it should be interpreted as being in the namespace identified by the conformance URI of the document section it occurs in. If it is qualified, the prefix should be interpreted according to the namespace mappings in effect, as documented below.

Some requirements clarify the referenced specification(s), but do not place additional constraints upon implementations. For convenience, clarifications are annotated in the following manner: C

Some requirements are derived from ongoing standardization work on the referenced specification(s). For convenience, such forward-derived statements are annotated in the following manner: xxxx, where "xxxx" is an identifier for the specification (e.g., "WSDL20" for WSDL Version 2.0). Note that because such work was not complete when this document was published, the specification that the requirement is derived from may change; this information is included only as a convenience to implementers.

2.2. Namespaces

This specification uses a number of namespace prefixes throughout; their associated URIs are listed below. Note that the choice of any namespace prefix is arbitrary and not semantically significant. Namespace URI of the general form "some-URI" represents some application dependent or context-dependent URI as defined in RFC 2396 [RFC 2396].

Prefix	XML Namespace	Reference(s)
s11	http://schemas.xmlsoap.org/soap/envelope	SOAP 1.1 [SOAP 1.1]
s12	http://www.w3.org/2003/05/soap-envelope	SOAP 1.2 [SOAP 1.2]
wsdl	http://schemas.xmlsoap.org/wsdl	WSDL 1.1 [WSDL 1.1]
soap	http://schemas.xmlsoap.org/wsdl/soap	WSDL 1.1 [WSDL 1.1]
uddi	urn:uddi-org:api_v2	UDDI 2.0 [UDDI 2.0]
xs	http://www.w3.org/2001/XMLSchema	XML Schema [Part 1, 2]
ds	http://www.w3.org/2000/09/xmldsig#	XML Digital Signatures
xenc	http://www.w3.org/2001/04/xmlenc#	XML Digital Encryption

wsa	http://schemas.xmlsoap.org/ws/2004/08/addressing	WS-Addressing [WS-Addressing]
wsrm	http://schemas.xmlsoap.org/ws/2005/02/rm	WS-ReliableMessaging [WS-ReliableMessaging]
wssc	http://schemas.xmlsoap.org/ws/2005/02/sc	WS-SecureConversation [WS-SecureConversation]
wsu	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd	WS-SecurityUtility
wsse	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.1.xsd	WS-Security Extensions [WS-Security]
wst	http://schemas.xmlsoap.org/ws/2005/02/trust	WS-Trust [WS-Trust]
wsp	http://schemas.xmlsoap.org/ws/2004/09/policy	WS-Policy [WS-Policy]
sp	http://schemas.xmlsoap.org/ws/2005/07/securitypolicy	WS-SecurityPolicy [WS-SecurityPolicy]

3. Profile Conformance

Conformance to the Profile is defined by adherence to the set of *requirements* defined for a specific *target*, within the *scope* of the Profile. This section explains these terms and describes how conformance is defined and used.

3.1. Conformance Requirements

Requirements state the criteria for conformance to the Profile. They typically refer to an existing specification and embody refinements, amplifications, interpretations and clarifications to it in order to improve interoperability. All requirements in the Profile are considered normative, and those in the specifications it references that are in-scope (see "Conformance Scope") should likewise be considered normative. When requirements in the Profile and its referenced specifications contradict each other, the Profile's requirements take precedence for purposes of Profile conformance.

Requirement levels, using [RFC2119] language (e.g., MUST, MAY, SHOULD) indicate the nature of the requirement and its impact on conformance. Each requirement is individually identified (e.g., R9999) for convenience.

For example;

R9999 *WIDGETs SHOULD be round in shape*.

This requirement is identified by "R9999", applies to the target WIDGET (see below), and places a conditional requirement upon widgets; i.e., although this requirement must be met to maintain conformance in most cases, there are some situations where there may be valid reasons for it not being met (which are explained in the requirement itself, or in its accompanying text).

Each requirement statement contains exactly one requirement level keyword (e.g., "MUST") and one conformance target keyword (e.g., "MESSAGE"). Additional text may be included to illuminate a requirement or group of requirements (e.g., rationale and examples); however, prose surrounding requirement statements must not be considered in determining conformance.

Definitions of terms in the Profile are considered authoritative for the purposes of determining conformance.

None of the requirements in the Profile, regardless of their conformance level, should be interpreted as limiting the ability of an otherwise conforming implementation to apply security countermeasures in response to a real or perceived threat (e.g., a denial of service attack).

3.2. Conformance Targets

Conformance targets identify what artifacts (e.g., SOAP message, WSDL description, UDDI registry data) or parties (e.g., SOAP processor, end user) requirements apply to.

This allows for the definition of conformance in different contexts, to assure unambiguous interpretation of the applicability of requirements, and to allow conformance testing of artifacts (e.g., SOAP messages and WSDL descriptions) and the behavior of various parties to a Web service (e.g., clients and service instances).

Requirements' conformance targets are physical artifacts wherever possible, to simplify testing and avoid ambiguity.

The following conformance targets are used in the Profile:

- **MESSAGE** - protocol elements that transport the ENVELOPE (e.g., SOAP/HTTP messages)
- **ENVELOPE** - the serialization of the s12:Envelope element and its content
- **DESCRIPTION** - descriptions of types, messages, interfaces and their concrete protocol and data format bindings, and the network access points associated with Web services (e.g., WSDL descriptions)
- **SENDER_PROXY -** software that generates a message according to the protocol associated with it and that sends it to a receiver proxy potentially through a message path that involves one or multiple relay(s).
- **RECEIVER_PROXY -** software that consumes a message according to the protocol associated with it.
- **RELAY -** software in the message path to the receiver proxy.

3.3. Conformance Scope

The scope of the Profile delineates the technologies that it addresses; in other words, the Profile only attempts to improve interoperability within its own scope. Generally, the Profile's scope is bounded by the specifications referenced by it.

The Profile's scope is further refined by extensibility points. Referenced specifications often provide extension mechanisms and unspecified or open-ended configuration parameters; when identified in the Profile as an extensibility point, such a mechanism or parameter is outside the scope of the Profile, and its use or non-use is not relevant to conformance.

Note that the Profile may still place requirements on the use of an extensibility point. Also, specific uses of extensibility points may be further restricted by other profiles, to improve interoperability when used in conjunction with the Profile.

Because the use of extensibility points may impair interoperability, their use should be negotiated or documented in some fashion by the parties to a Web service; for example, this could take the form of an out-of-band agreement.

The Profile's scope is defined by the referenced specifications in Appendix A, as refined by the extensibility points in Appendix B.

4. PRESTO: PRotocole d'Echanges Standard et Ouvert

This section describes the "PRotocole d'Echanges Standard et Ouvert" 1.0 (aka PRESTO) as a normative profile for the set of Web services specifications referenced by the subsequent subsections.

4.1. PRESTO Supported Transport Protocols

The PRESTO protocol is fundamentally transport agnostic. The vehicle for the PRESTO protocol can be a high-level protocol like HTTP (it sits higher in the stack than a network protocol such as TCP), as well as any network protocol such as TCP or UDP that are suitable.

Section § 5 "Bindings" of this document describes the currently supported bindings with a specific transport protocol for the Profile.

The first version of PRESTO defines a HTTP binding only. Subsequent versions of the profile may consider other bindings (FTP, SMTP…).

4.2. PRESTO Messages

The PRESTO message format is XML, which is the de facto format for the communication between applications. XML provides a defined structure that that gives meaning to data.

This Profile mandates the use of the W3C Recommendation XML at version 1.1. The XML specification is at http://www.w3.org/XML.

As such, using XML to provide the basic message structure makes sense to ensure interoperability between systems and applications in a heterogeneous eGovernment space.

The "envelope" format for messages enables message metadata to be stored alongside the payload. The PRESTO message uses the SOAP format as an envelope for encapsulating both the metadata and the payload.

This Profile mandates the use of SOAP version 1.2 [SOAP 1.2].

This section of the Profile incorporates the following specifications by reference, and defines extensibility points within them:

- SOAP 1.2 (W3C Recommendation 24 June 2003) [SOAP 1.2]

The PRESTO SOAP message conveys in its headers technical metadata that are concerned with message addressing, reliability, and security. These areas are respectively covered in the following sections § 4.4 "Addressing Messages", § 4.6 "Reliable Delivery of Messages and QoS", and 4.7 "Security".

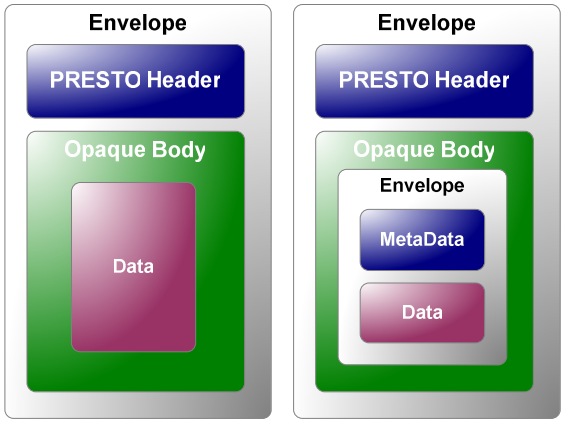
The business data are OPAQUE to the PRESTO protocol. The business data are located in the body of the PRESTO SOAP message

Only one root level element MUST be present under the s12:Body element. The PRESTO technical envelope format permits any payload to be nested inside the PRESTO SOAP message as business data.

If business metadata (metadata associated with business data such as binary documents) are required, those business metadata shall not be considered as part of PRESTO header but as part of the opaque body.

Payload can be:

- Data.
- Envelope consisting in metadata and data.



4.3. Sending and Receiving Messages

4.3.1. Use of Document-Literal WSDL**

The WS-I Basic Profile defines the terms "document-literal" and "rpc-literal" (http://ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.html#WSDLMSGS) as applies to the permitted WSDL authoring styles. This profile limits the choice of WSDL authoring style permitted by requirement R2705 (http://ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.html#R2705) to "document-literal".

R3001 *A wsdl:port in a DESCRIPTION MUST use a "document-literal binding"*.

While the Profile requires use of "document-literal" style WSDL, it should be noted that the convention known as "document-literal wrapped" may be used to capture RPC semantics.

The characteristics of the document/literal wrapped pattern are as follows:

- The input and output wsdl:messages each have a single part
- The wsdl:part for the input and output messages each have an @element attribute
- The referenced xsd:element declaration of the input message's wsdl:part/@element has a name attribute with the same value as the wsdl:operation for which the wsdl:message will be used
- The referenced xsd:element declaration of the output message's wsdl:part/@element has a name attribute with the same value as the wsdl:operation for which the wsdl:message will be used, appended with "Response"

- The element's complex type has no attributes
- The WSDL binding is a document-literal binding

For example,

CORRECT:

```
<types>
   <schema>
      <element name="myMethod"/>
         <complexType>
            <sequence>
               <element name="x" type="xsd:int"/>
            </sequence>
         </complexType>
      </element>
      <element name="myMethodResponse"/>
         <complexType>
            <sequence>
               <element name="y" type="xsd:string"/>
            </sequence>
         </complexType>
      </element>
   </schema>
</types>
<message name="myMethodRequest">
   <part name="body" element="myMethod"/>
</message>
<message name="myMethodResponse">
   <part name="body" element="myMethodResponse"/>
</message>
<message name="empty"/>
<portType name="myPortType">
   <operation name="myMethod">
      <input message="myMethodRequest"/>
      <output message="myMethodResponse"/>
   </operation>
</portType>
```

4.3.2. Overriding Requirements of the WS-I Basic Profile 1.1**

Normally, a profile would NEVER contradict a referenced specification's requirements. However, in this case, it has proven to be a necessary evil in order to accommodate composition of the WS-I Basic Profile with the WS-Addressing and WS-ReliableMessaging specifications. We hope that WS-I will take note and address the composition issues in a subsequent revision of the WS-I Basic Profile.

4.3.2.1. SOAP Envelope in HTTP Response Message**

Typically, the HTTP Response Code of a one-way WSDL would be "202 Accepted" with no SOAP envelope in the entity body of the HTTP Response Message. However, certain Web

services specifications such as WS-ReliableMessaging can cause the generation of SOAP messages that are not considered to be application-level "responses" yet need to be sent back to the originator of the message carried on the HTTP Request Message. For example, when using WS-ReliableMessaging, if the wsrm:AcksTo EPR is the anonymous URI, then the RM Destination (e.g. PRESTO Receiver Proxy) would send back SequenceAcknowledgement messages to the RM Source (e.g. PRESTO Sender Proxy) on the HTTP Response Message. However, if the request message is a one-way message, there is no response message to flow back – meaning the RM Destination must generate a new SOAP envelope for the specific purpose of delivering the SequenceAcknowledgement.

This would violate the WS-I Basic Profile requirements R2714 and R2750, which state that the response to a one-way message must include an HTTP response code of "2xx" without a SOAP envelope, and that the receiver of this flow must ignore any envelope if there is one. However, since the anonymous EPR was chosen by the RM Source, one would assume that it would be expecting this behavior. Therefore, the Profile overrides the requirements R2714 and R2750 of the WS-I Basic Profile 1.1.

R3002 *An HTTP Response MESSAGE corresponding to a WSDL one-way operation MAY contain a SOAP envelope in its entity body.*

4.4. Addressing Messages**

In order to effect the types of message exchange patterns (MEPs) required to enable the PRESTO usage scenarios some form of addressing is required.
This Profile mandates the use of the WS-Addressing submission to the W3C. At such time as the W3C WS Addressing WG produces a Proposed Recommendation (PR), consideration will be given as to adopting its use.

This section of the Profile incorporates the following specifications by reference, and defines extensibility points within them:

- WS-Addressing (W3C Member Submission 10 August 2004) [WS-Addressing]

4.4.1. Use of Addressing Header Blocks**

4.4.1.1. Presence of Message Headers Blocks

R0001 *An ENVELOPE MUST contain exactly one wsa:To header*.

R0002 *An ENVELOPE MUST contain exactly one wsa:MessageId header*.

R0003 *An ENVELOPE MUST contain exactly one wsa:Action header*.

For example,

CORRECT:

```
<s12:Envelope
      xmlns:s11="http://schemas.xmlsoap.org/soap/envelope"
      xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing" >
  <s12:Header>
    <wsa:To>http://example.com/service</wsa:To>
    <wsa:ReplyTo s12:mustUnderstand='1'>
      <wsa:Address>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</wsa:Address>
    </wsa:ReplyTo>
    <wsa:MessageId>uuid:aaaabbbb-cccc-dddd-eeee-ffffffffffff</wsa:MessageId>
    <wsa:Action>http://example.com/service/tbd</wsa:Action>
  </s12:Header>
  <s12:Body>
      <!-- body contents OPAQUE to the PRESTO protocol -->
  </s12:Body>
</s12:Envelope>
```

4.4.1.2. Relating Output Operation to Input Operation

In the context of asynchronous messaging, it is necessary that the envelope contain correlation metadata sufficient to enable the receiver of a response message to correlate that response with the original request message.

R0010 *Any ENVELOPE MAY contain a wsa:ReplyTo header block.*

R0011 *If, in a WSDL description, a wsdl:operation is described with a wsdl:output, the ENVELOPE contained within the message corresponding to the wsdl:input of that same wsdl:operation MUST contain a wsa:ReplyTo header block.*

R0012 *If, in a WSDL description, a wsdl:operation is described with a wsdl:output, the ENVELOPE contained within the message corresponding to the wsdl:output of that same wsdl:operation MUST contain a wsa:RelatesTo header block containing the value of the wsa:MessageId of the corresponding input message.*

4.4.1.3. Use of s12:mustUnderstand attribute

A SENDER_PROXY that sends a request message whose response is expected to be delivered to the endpoint specified in the wsa:ReplyTo SOAP header block needs to ensure that the receiver of that message understands the processing semantics of the wsa:ReplyTo header block.

R0020 *If an ENVELOPE contains a wsa:ReplyTo header block, that header block MUST have a s12:MustUnderstand attribute with a value of '1'.*

4.4.2. Asynchronous Request/Response Considerations**

4.4.2.1. Expectations of the HTTP response message

In the context of the WS-I Basic Profile, the only reason the response message to an HTTP request would be empty is when the request message is a one-way message. However, the WS-Addressing specification introduces a wsa:ReplyTo SOAP header block that changes the nature of a request/response WSDL operation as it relates to the HTTP binding. If a request message includes a wsa:ReplyTo header that does not use the WS-Addressing Anonymous URI, then the response message is expected to be sent to the wsa:ReplyTo endpoint reference (EPR) rather than in the HTTP response message. Since the response message will not be sent in the HTTP response message, the HTTP response code on the original connection would be "202 Accepted". While this doesn't violate any WS-I Basic Profile requirements it is a change in behavior.

R0100 *If an envelope corresponding to a wsdl:input contains a wsa:ReplyTo EPR with an address other than the WS-Addressing Anonymous URI, then the corresponding HTTP response MESSAGE MAY have an HTTP return code of "202 Accepted" and an empty entity body.*

4.4.3. Composition with WS-Security**

The requirements in this section apply whenever WS-Security is being used in conjunction with WS-Addressing.

4.4.3.1. Signing Headers Blocks

R2000 *When present in an ENVELOPE, each of the following SOAP header blocks MUST be included in the signature whenever the s12:Body is being signed: wsa:To, wsa:From, wsa:Action, wsa:ReplyTo, wsa:FaultsTo, wsa:MessageId, wsa:RelatesTo.*

4.5. Managing Attachments

This section of the Profile incorporates the following specifications by reference, and defines extensibility points within them:

- MTOM 1.0 (W3C Recommendation 25 January 2005) [MTOM]
- XOP 1.0 (W3C Recommendation 25 January 2005) [XOP]

4.5.1. Transport of binary documents

In order to respect the XML infoset, the binary documents must be included in a XML document as a base 64 binary element.

The binary content of the XML element may be described using a xmime:contentType attribute as defined in the W3C Working Group Note "Describing Media Content of Binary Data in XML" (http://www.w3.org/TR/xml-media-types/).

R2000 *The binary document MUST be included in the MESSAGE using binary 64 encoding.*

4.5.2. Optimizing the transport of binary data

Using base 64 encoding inflates the size of the documents. The use of MTOM/XOP mechanism facilitates the transport of the documents by selecting and optimizing the binary elements from the XML data.

The base 64 binary encoded elements must be in a canonical lexical representation of the xs:base64Binary data type.

R2000 *To be optimized, the characters of all binary elements of the MESSAGE MUST be in the canonical form of the XML Schema type xs:base64Binary.*

4.6. Reliable Delivery of Messages and QoS

The PRESTO protocol requires reliable message delivery as a quality of service characteristic such as that provided by the WS-ReliableMessaging (WS-RM) specification. Use of the WS-RELIABLEMESSAGING protocol is mandatory.

This section of the Profile incorporates the following specifications by reference, and defines extensibility points within them:

- WS-ReliableMessaging (WS-RM) [WS-ReliableMessaging]

4.6.1. Sequence Header Block**

4.6.1.1. Use of s12:mustUnderstand attribute

When a message is sent reliably, using the WS-ReliableMessaging protocol, it is imperative to the success of the protocol that the recipient of that message understand the processing semantics of the wsrm:Sequence header block.

R1010 *If an ENVELOPE contains a wsrm:Sequence header block, that header block MUST have a s12:MustUnderstand attribute with a value of '1'.*

For example,

INCORRECT:

```
<s12:Envelope
      xmlns:s11="http://schemas.xmlsoap.org/soap/envelope"
      xmlns:wsrm="http://schemas.xmlsoap.org/ws/2005/02/rm"
      xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
  <s12:Header>
    <wsa:MessageID>http://example.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546817</wsa:MessageID>
    <wsa:To>http://example.org/service</wsa:To>
    <wsa:From>http://example.com/client/rm</wsa:From>
    <wsa:Action>http://example.org/service/credit</wsa:Action>
    <wsa:ReplyTo s12:mustUnderstand='1'>
      <wsa:Address>http://example.com/client</wsa:Address>
    </wsa:ReplyTo>
    <wsrm:Sequence>
      <wsrm:Identifier>
        http://example.org/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546817
      </wsrm:Identifier>
      <wsrm:MessageNumber>42</wsrm:MessageNumber>
    </wsrm:Sequence>
  </s12:Header>
  <s12:Body>

      <!-- body contents OPAQUE to the PRESTO protocol -->

  </s12:Body>
</s12:Envelope>
```

CORRECT:

```
<s12:Envelope
      xmlns:s11="http://schemas.xmlsoap.org/soap/envelope"
      xmlns:wsrm="http://schemas.xmlsoap.org/ws/2005/02/rm"
      xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
  <s12:Header>
    <wsa:MessageID>http://example.com/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546817</wsa:MessageID>
    <wsa:To>http://example.org/service</wsa:To>
    <wsa:From>http://example.com/client/rm</wsa:From>
```

```
    <wsa:Action>http://example.org/service/credit</wsa:Action>
    <wsa:ReplyTo s12:mustUnderstand='1'>
      <wsa:Address>http://example.com/client</wsa:Address>
    </wsa:ReplyTo>
    <wsrm:Sequence s12:mustUnderstand='1'>
      <wsrm:Identifier>
        http://example.org/guid/0baaf88d-483b-4ecf-a6d8-a7c2eb546817
      </wsrm:Identifier>
      <wsrm:MessageNumber>42</wsrm:MessageNumber>
    </wsrm:Sequence>
  </s12:Header>
  <s12:Body>

      <!-- body contents OPAQUE to the PRESTO protocol -->

  </s12:Body>
</s12:Envelope>
```

4.6.2. SequenceAcknowledgement Header Block**

4.6.2.1. Piggy-backing SequenceAcknowledgement

If a SOAP envelope happens to be going to the same endpoint that was specified by a wsrm:CreateSequence/wsrm:AcksTo EPR, then the RM destination (e.g. PRESTO Receiver Proxy) can add a wsrm:SequenceAcknowledgement SOAP header (for each relevant Sequence) to the envelope instead of sending them in separate messages to the RM source (e.g. PRESTO Sender Proxy). This practice is commonly referred to as "piggy-backing" of acknowledgements.

R1021 *A wsrm:SequenceAcknowledgement header, for each relevant Sequence, MAY be included in any ENVELOPE destined for the endpoint specified by the wsrm:CreateSequence/wsrm:AcksTo EPR.*

4.6.2.2. Carrying SequenceAcknowledgement on HTTP response for one-way operations

The Basic Profile requirements R2714 (http://ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.html#R2714) and R2750 (http://ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.html#R2750) conflict with the ability to leverage the HTTP response message to carry wsrm:SequenceAcknowledgements for a reliable one-way operation. As this requirement is likely to be commonplace in such cases where the RM Source is hidden behind a firewall, this profile overrides the Basic Profile requirements R2714 and R2750.

R1022 *A MESSAGE corresponding to the HTTP response message for a one-way operation MAY contain a SOAP envelope with a wsrm:SequenceAcknowledgement SOAP header block and no child element of the s12:Body in its entity body.*

R1023 *A CONSUMER that specifies the WS-Addressing Anonymous URI as the address in a wsrm:CreateSequence/wsrm:AcksTo EPR MUST be prepared to receive and process HTTP response messages for a one-way operation that contain a SOAP envelope in their entity bodies.*

4.6.3. Composition with WS-Addressing**

4.6.3.1. Use of WS-Addressing Anonymous URI

R1050 *If the WS-Addressing Anonymous URI is used in the wsrm:AcksTo element of a wsrm:CreateSequence ENVELOPE, all SequenceAcknowledgement messages for the created Sequence MUST be sent back on the response flow of the HTTP connection on which a message is received containing a wsrm:Sequence header block with a wsrm:Identifier element containing the same value as the created Sequence.*

The ability to retry the transmission of messages is a key component of WS-ReliableMessaging. However, this requires that the sender of a message must be able to establish a connection with the receiving endpoint. As such, using the anonymous URI in the wsa:ReplyTo EPR of the request message violates this because the sender of the response can not initiate new connections back to this EPR as needed to support WS-ReliableMessaging.

R1051 *In a message exchange in which both the input (request) and output (response) messages are to be delivered reliably using WS-ReliableMessaging, the input ENVELOPE's wsa:ReplyTo EPR MUST NOT be the WS-Addressing Anonymous URI.*

4.6.4. Composition with WS-Security**

In order to effect end-to-end secure, reliable messaging between business partners, the typical usage scenario will compose the use of WS-ReliableMessaging, WS-Addressing, and WS-Security, as constrained by the WS-I Basic Security Profile (BSP) 1.0. The requirements in this section apply whenever WS-Security is being used in conjunction with WS-ReliableMessaging.

4.6.4.1. Signing Header Blocks

In order to ensure the integrity of the WS-ReliableMessaging protocol elements (e.g. that they have not been tampered-with by a "man-in-the-middle"), they need to be digitally signed.

R2010 *When present in an ENVELOPE, each of the following SOAP header blocks MUST be included in the signature whenever the s12:Body is being signed: wsrm:Sequence, wsrm:SequenceAcknowledgement.*

4.6.4.2. Enabling Detection of Replay Attacks

The value of the wsu:Timestamp element must be changed for retransmissions so that the security layer at the RM Destination (e.g. PRESTO Receiver Proxy) can use that information to distinguish between a valid retransmission of an unacknowledged message and a potential replay attack mounted by a third party.

R2011 *The wsse:Security header block in an ENVELOPE MUST contain a wsu:Timestamp child element.*

R2012 *The value of the wsu:Timestamp element in an ENVELOPE MUST be the system time of the RM Source at time of transmission for each transmission of a message (the initial and each successive transmission of an unacknowledged message).*

4.6.5. Delivery Assurance

In order to ensure a maximum quality of service when exchanging documents, the ExactlyOnce assurance is required. The InOrder assurance is optional.

R2011 *The ReliableMessaging delivery assurance of MESSAGES MUST be ExactlyOnce.*

4.7. Security

The PRESTO usage scenarios require end-to-end message security as a quality of service characteristic such as that provided by the WS-I Basic Security Profile (BSP) 1.0 (http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html) specification.

Use of the WS-I BSP is mandatory of message integrity and optional for business data confidentiality. Where security is needed, WS-I BSP is the sanctioned mechanism.

At the time of this publication, the BSP1.0 is still a Working Group Draft. As the BSP progresses towards, this Profile will be updated to reflect any necessary changes.

This section of the Profile incorporates the following specifications by reference, and defines extensibility points within them:

- WS-I Basic Security Profile (BSP) 1.0 (WS-I Working Group Draft 20 January 2006) (http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html)

4.7.1. Constraining the Basic Security Profile

This profile further constrains certain of the options permitted by the WS-I Basic Security Profile to improve the prospect for interoperability.

4.7.1.1. Use of X.509v3 Certificates for signature and encryption

X.509v3 Certificates are used to guarantee the integrity and confidentiality of PRESTO messages.

R2011 *When signing the MESSAGE, a X509v3 certificate MUST be used as a security token.*

R2012 *When encrypting the MESSAGE, a X509v3 certificate MUST be used as a security token.*

4.7.1.2. Message Signature

The business data are OPAQUE to the PRESTO protocol, therefore when a signature of the message is required, the entire business data must be signed.

R2011 *When the MESSAGE is signed, the entire s12:Body element MUST be signed.*

As recommended by the W3C, the following algorithms are recommended by the DGME for the signature of PRESTO messages: RSAwithSHA1 for signature, Canonical with Comments for canonicalization, XPath for transformations.

The signature algorithms must be compliant with the Basic Security Profile 1.0.

4.7.1.3. Message Encryption

The business data are OPAQUE to the PRESTO protocol, therefore when an encryption of the message is required; the entire business data must be encrypted.

R2011 *When the MESSAGE is encrypted,, the entire s12:Body element MUST be encrypted.*

The encryption algorithms must be compliant with the Basic Security Profile 1.0.

5. Bindings

This section describes how to bind the Profile with the currently supported transports.

5.1. HTTP Transport Protocol Binding

With the exemptions listed in the previous chapters, the HTTP Transport binding is done with compliance to the HTTP Transport Binding in the Basic Profile 1.1.

Appendix A: Referenced Specifications

[MTOM]
Ed. Noah Mendelsohn et al, "SOAP Message Transfer Optimization Mechanism," July 2004.

[PRESTO-Ref]
"A Technical Reference for the PRESTO protocol," April 2006.

[RFC 2119]
S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," RFC 2119, Harvard University, March 1997.

[RFC 2396]
T. Berners-Lee, et al, "Uniform Resource Identifier (URI): Generic Syntax," RFC 2396bis, W3C/MIT, July 2004.

[SOAP 1.1]
Don Box et al, W3C Note "Simple Object Access Protocol (SOAP) 1.1", May 2000.

[SOAP 1.2]
M. Gudgin et al, "SOAP Version 1.2 Part 1: Messaging Framework," June 2003.

[UDDI]
Ed. Tom Bellwood, "UDDI Version 2.04 API Specification," July 2002.

[WSDL 1.1]
Ed. Erik Christensen et al, "Web Service Description Language (WSDL) 1.1", March 2001.

[WS-Addressing]
D. Box et al, "Web Services Addressing (WS-Addressing)," August 2004.

[WS-Policy]
D. Box et al, "Web Services Policy Framework (WS-Policy)," September 2004

[WS-MetadataExchange]
Keith Ballinger et al, "Web Services Metadata Exchange (WS-MetadataExchange)," September 2004

[WS-RM]
Ruslan Bilorusets et al, "Web Services Reliable Messaging (WS-ReliableMessaging)," February 2005.

[WS-RMPolicy]
Stefan. Batres et al, "Web Services Reliable Messaging Policy Assertion (WS-RM Policy)," February 2005.

[WS-SecureConv]
Steve Anderson et al, "Web Services Secure Conversation Language (WS-SecureConversation)," February 2005.

[WS-Security]
A. Natalin et al, "Web Services Security: SOAP Message Security 1.0", May 2004.

[WS-SecX509]
Ed. Phillip Hallam-Baker et al, "Web Services Security: X.509 Token Profile V1.0", March 2004.

[WS-SecurityPolicy]
Giovanni Della-Libera et al, "Web Services Security Policy Language (WS-SecurityPolicy)," July 2005.

[WS-Trust]
Steve Anderson et al, "Web Services Trust Language (WS-Trust)," February 2005.

[XMLDSIG]
Eastlake III, D., Reagle, J., and Solo, D., "XML-Signature Syntax and Processing", http://www.ietf.org/rfc/rfc3275.txt, March 2002.

[XMLENC]
Imamura, T., Dillaway, B., and Simon, E., "XML Encryption Syntax and Processing", http://www.w3.org/TR/xmlenc-core/, August 2002.

[XML Schema, Part 1]
H. Thompson et al, "XML Schema Part 1: Structures," May 2001.

[XML Schema, Part 2]
P. Biron et al, "XML Schema Part 2: Datatypes," May 2001.

[XOP]
Ed. Noah Mendelsohn et al, "XML-binary Optimized Packaging (XOP)," June 2004.

Appendix B: Extensibility Points

This section identifies extensibility points, as defined in "Scope of the Profile," for the Profile's component specifications.

These mechanisms are out of the scope of the Profile; their use may affect interoperability, and may require private agreement between the parties to a Web service.

- Support of timestamp token when it is defined in the "Référentiel Général de Sécurité",
- Support of oversized messages (Go), chunking algorithm,
- Support of WS-SecureConversation,
- Support of WS-Policy,
- Routing and multi-hop, message lifetime.

Appendix C: Glossary and Acronyms

Authentication

The process of validating security credentials.

Authorization

The process of granting access to a secure resource based on the security credential provided.

Canonicalization

The process of converting an XML document to a form that is consistent to all parties. Used when signing documents and interpreting signatures.

Claim

A claim is a statement made about a PRESTO sender proxy, a PRESTO receiver proxy or other resource (e.g., name, identity, key, group, privilege, capability, etc.).

Deserialization

The process of constructing an XML Infoset from an octet stream. It is the method used to create the Infoset representation of a message from the wire format for a message.

Effective Policy

An effective policy, for a given policy subject, is the resultant combination of the policies attached to policy scopes that contain the policy subject.

Exchange Pattern

The model used for message exchange between PRESTO proxies.

Factory

A factory is a Web service that can create a resource from an XML representation.

Message

A message is a complete unit of data available to be sent or received by services. It is a self-contained unit of information exchange. A message always contains a SOAP envelope, and may include additional MIME parts as specified in MTOM, and/or transport protocol headers.

Message Path

The set of SOAP nodes traversed between the original source (initial PRESTO sender proxy) and ultimate receiver (final PRESTO receiver proxy).

Policy

A policy is a collection of policy alternatives.

Policy Alternative

A policy alternative is a collection of policy assertions.

Policy Assertion

A policy assertion represents a domain-specific individual requirement, capability, other property, or a behavior.

Policy Expression

A policy expression is an XML Infoset representation of a policy, either in a normal form or in an equivalent compact form.

PRESTO Proxy

A PRESTO proxy is a service.

PRESTO Sender Proxy

A PRESTO sender proxy is a service that generates a message according to the PRESTO protocol and that sends it to a PRESTO receiver proxy potentially through a message path that involves one or multiple PRESTO Relay(s).

PRESTO Receiver Proxy

A PRESTO receiver proxy is a service that consumes a message according to the PRESTO protocol.

PRESTO Relay

A PRESTO relay is a SOAP intermediary in the message path to the PRESTO receiver proxy.

Protocol Composition

Protocol composition is the ability to combine protocols while maintaining technical coherence and absent any unintended functional side effects.

Resource

A resource is defined as any entity addressable by an endpoint reference where the entity can provide an XML representation of itself.

Security Context

A security context is an abstract concept that refers to an established authentication state and negotiated key(s) that may have additional security-related properties.

Security Context Token

A Security Context Token (SCT) is a wire representation of the security context abstract concept, and which allows a context to be named by a URI and used with [WS-Security].

Security Token

A security token represents a collection of claims.

Security Token Service

A security token service (STS) is a Web service that issues security tokens (see [WS-Security]). That is, it makes assertions based on evidence that it trusts, to whoever trusts it (or to specific recipients). To communicate trust, a service requires proof, such as a signature to prove knowledge of a security token or set of security token. A PRESTO proxy itself can generate tokens or it can rely on a separate STS to issue a security token with its own trust statement (note that for some security token formats this can just be a re-issuance or co-signature). This forms the basis of trust brokering.

Serialization

The process of representing an XML Infoset as an octet stream. It is the method used to create the wire format for a message.

Service

A software entity whose interactions with other entities are via messages. Note that that a service need not be connected to a network.

Signature

A signature is a value computed with a cryptographic algorithm and bound to data in such a way that intended recipients of the data can use the signature to verify that the data has not been altered and has originated from the signer of the message, providing message integrity and authentication. The signature can be computed and verified with either symmetric or asymmetric key algorithms.

SOAP Intermediary

A SOAP intermediary is a SOAP processing node that is neither the original message sender nor the ultimate receiver.

Symmetric Key Algorithm

An encryption algorithm where the same key is used for both encrypting and decrypting a message.

Trust

Trust indicates that one entity is willing to rely upon a second entity to execute a set of actions and/or to make set of assertions about a set of subjects and/or scopes.

Trust Domain

A Trust Domain is an administered security space in which the source and target of a request can determine and agree whether particular sets of credentials from a source satisfy the relevant security policies of the target. The target may defer the trust decision to a third party (if this has been established as part of the agreement) thus including the trusted third party in the Trust Domain.

Web Service

A Web service is a reusable piece of software that interacts exchanging messages over the network through XML, SOAP, and other industry recognized standards.