A guide to supporting PRESTO

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 guide describes the message exchange scenarios that leverage the PRESTO protocol. The messages described in this document provide the basis for testing interoperability with the PRESTO protocol. The exchanged messages between PRESTO proxies (sender, receiver, and relay) are illustrated to allow others to create proxies and applications that can use and interoperate with the PRESTO protocol. Some directions for such proxy implementations are also highlighted whenever relevant for the overall mechanism understanding. This document is intended to be read alongside the PRESTO Technical Reference [PRESTO-Ref] which provides a normative profile for the set of Web services specifications referenced by this document.

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 guide to supporting PRESTO**

Notice

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

presto	**PRotocole d'Echanges Standard et Ouvert** **A guide to supporting PRESTO**

Table of Contents

1. Introduction

The "PRotocole d'Echanges Standard et Ouvert" 1.0 (aka PRESTO) protocol aims at providing the generic message exchange layer for exchanging eGovernement documents.

PRESTO is designed according to the requirements and constraints that have been gathered by the DGME.

The PRESTO protocol in its initial version builds on the foundations of WS-Addressing [WS-Addressing], MTOM [MTOM], WS-ReliableMessaging [WS-ReliableMessaging], WS-Security [WS-Security], WS-SecureConversation [WS-SecureConversation] (and consequently WS-Trust [WS-Trust]) Web service specifications. The mechanisms described in these specifications provide the basis for supporting the key Message Exchange Patterns (MEP) required by such a protocol (see section § 1.1 "PRESTO Message Exchange Patterns (MEP)").

The PRESTO Technical Reference [PRESTO-Ref] provides a normative profile for this set of Web services specifications referenced, along with clarifications, amendments, and restrictions of these specifications that promote interoperability. This document is intended to be read alongside with this present guide.

The metadata associated with the related PRESTO endpoints are not expressed via policies.

A future version of the PRESTO protocol will use policies to define the metadata associated with the related endpoints and be able to dynamically request and consume these policies based on the foundation of the WS-Policy [WS-Policy], WS-RMPolicy [WS-RMPolicy], WS-SecurityPolicy [WS-SecurityPolicy], and WS-MetadataExchange [WS-MetadataExchange] Web service specifications.

The PRESTO protocol is intended to be primarily implemented by PRESTO proxies. The dichotomy introduced by this guide between a PRESTO sender proxy and a PRESTO receiver proxy is purely logical to reflect their respective role they play in the message exchange. The former one is a service that generates and sends a message to the latter one with regards to the PRESTO protocol. Subsequently, a PRESTO receiver proxy is a service that receives and consumes a message with regards to the PRESTO protocol. As such, PRESTO proxies implement and expose a PRESTO protocol endpoint to the eGovernment partners and enable them to interoperate with any service that "speak" the PRESTO protocol. A PRESTO proxy implementation MAY play both roles.

The message path from a PRESTO sender proxy to a PRESTO receiver proxy MAY involves one or multiple PRESTO relay(s). PRESTO relays are intended to provide routing capabilities and MAY provide additional services.

Furthermore, nothing prevents a (source or target) application to directly implements and exposes a PRESTO protocol endpoint. Such an application can then "speak" directly the PRESTO protocol to talk to a PRESTO receiver proxy or to another application that also exposes such PRESTO protocol endpoint.

However, for clarity, this guide systematically illustrates the role played by both the PRESTO sender and receiver proxies in the message exchange so that, whatever the options and constraints are, the document can provide guidance for a successful and interoperable implementation of the PRESTO protocol.

1.1. PRESTO Message Exchange Patterns (MEP)

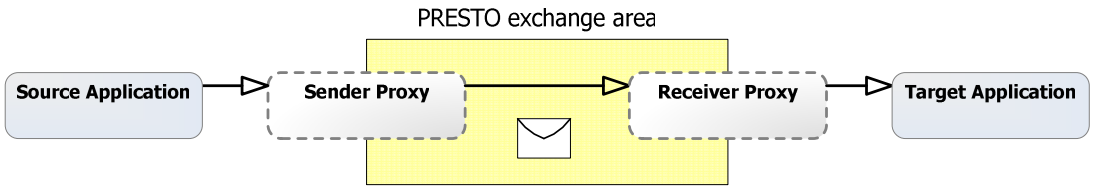
This section describes the message exchange scenarios that leverage the PRESTO protocol based on the feedbacks that have been gathered by the DGME.

The messages described in this document provide the basis for testing interoperability with the PRESTO protocol. The exchanged messages between PRESTO proxies (sender, receiver) and relay(s) are illustrated to allow others to create proxies and applications that can use and interoperate with the PRESTO protocol.

The PROXY's architecture and implementation are outside of the scope of the present document and are addressed in the PRESTO Proxy Specifications [PRESTOProxy]. However this guide may provide some architecture considerations and some directions for such proxy implementations whenever relevant. This may help in the understanding of the underlying mechanisms.

Following subsequent sections describes the four Message Exchange Patterns (MEP) currently supported by this version of the PRESTO protocol.

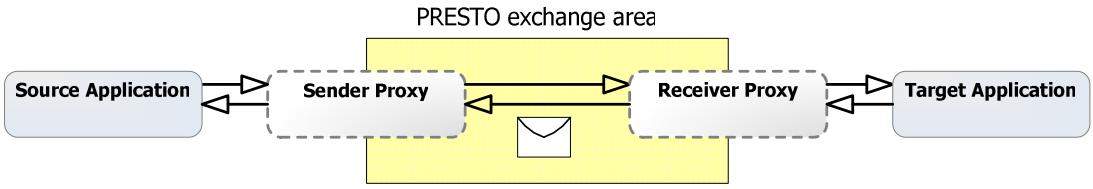
1.1.1. One-Way Message Exchange



This pattern consists in sending exactly one message as follows:

1. As illustrated above, the source application invokes the PRESTO sender proxy. The initiating (source) application transfers the business data (including one or more binary documents if any) as well as the identity of the target application.
2. The PRESTO sender proxy accordingly generates a PRESTO message containing the XML business data (including the binary documents if any) as an opaque payload in the body of the SOAP envelope. The PRESTO sender proxy fills the PRESTO technical headers in the SOAP envelope of the PRESTO message and sends the message to the PRESTO receiver proxy.
3. The PRESTO receiver proxy receives the PRESTO message.
4. The PRESTO receiver proxy sends an acknowledgement of receipt to the originating PRESTO sender proxy. The acknowledgement of receipt simply states that the message has been successfully received.
5. The PRESTO sender proxy receives the acknowledgement of receipt.
6. The PRESTO receiver proxy extracts the business data (including the binary documents if any) on behalf of the target application.

1.1.2. Request-Reply Message Exchange



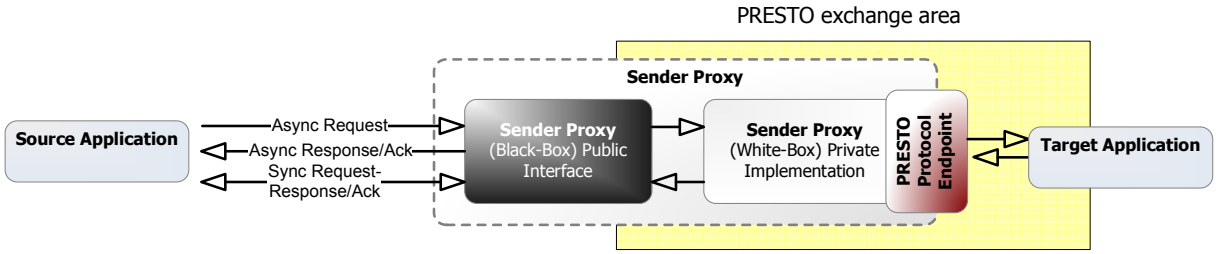
This pattern extends the previous One-Way MEP by adding a response message as follows:

1. Likewise, as an additional step to the previous step 6, the target application transfers its response (business data) to its PRESTO receiver proxy as well as the identity of the query.
2. The PRESTO receiver proxy generates a PRESTO message containing the XML business data (including the binary documents if any) as an opaque payload in the body of the SOAP envelope. The PRESTO receiver proxy fills the PRESTO technical headers in the SOAP envelope of the PRESTO message and sends the message to the originating PRESTO sender proxy.
3. The PRESTO sender proxy receives the PRESTO message.

4. The PRESTO sender proxy sends an acknowledgement of receipt to the PRESTO receiver proxy. The acknowledgement of receipt simply states that the message has been successfully received.
5. The PRESTO receiver proxy receives the acknowledgement of receipt.
6. The PRESTO sender proxy extracts the XML business data (and the binary documents as attachment(s) if any) for the source application. This step ends the message exchange and related session(s).

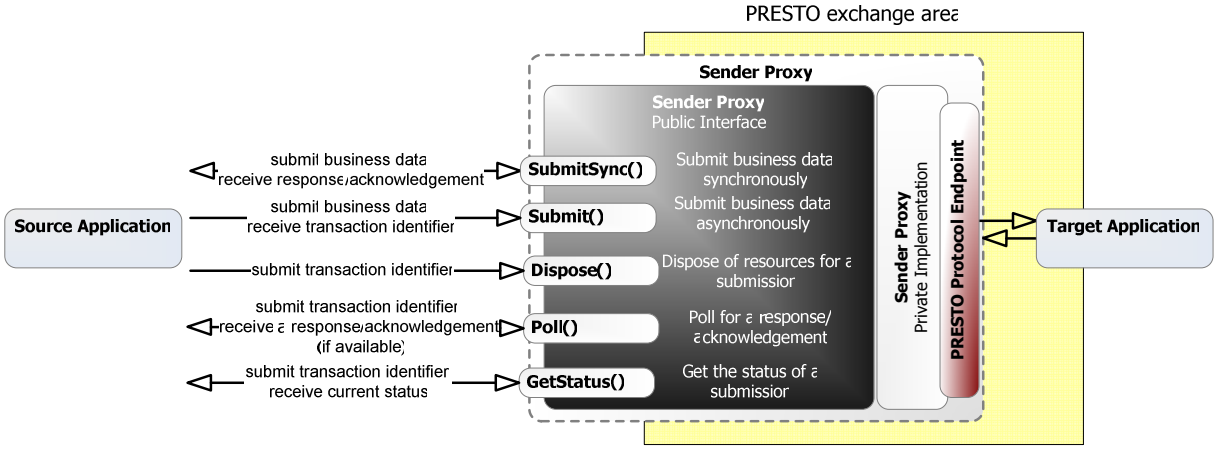
Considering the above descriptions for this pattern and the previous one, the basic architecture of a PRESTO proxy may be divided into two main areas:

- A (black box) public side used by applications that are submitting or receiving business data,
- A (white box) private side that implements a PRESTO protocol endpoint that confirms with the PRESTO protocol specifications and consequently interacts with any PRESTO proxy.



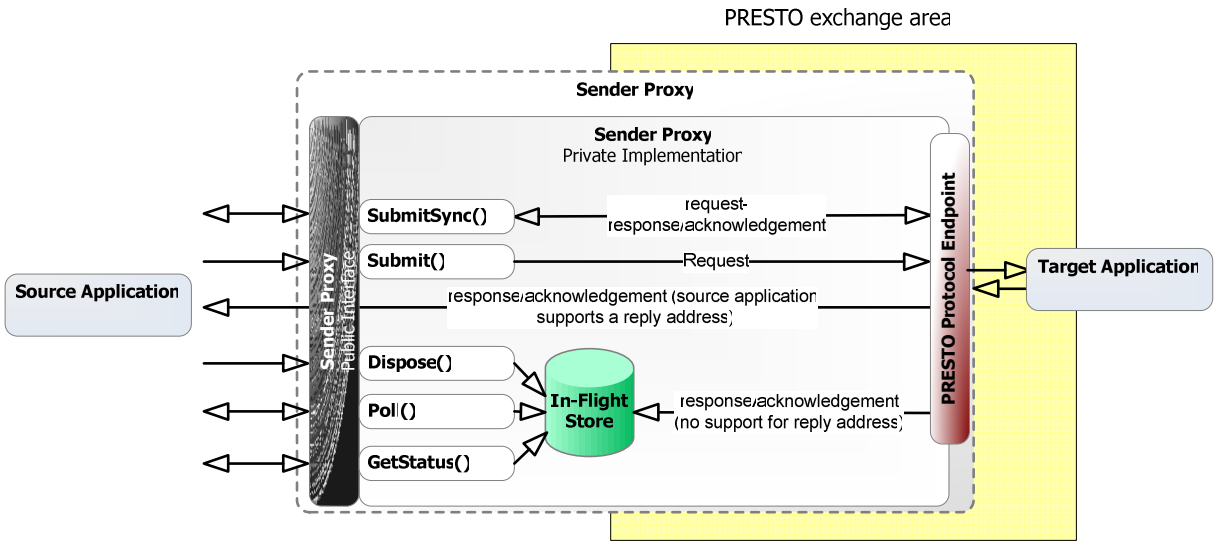
The public side of the PRESTO proxy describes how an application interacts with the service, what operations are supported, the necessary protocols and message formats that the service supports to interact with, and some additional non-functional requirements placed by the service – such as what level of security is expected for the exchange if any. It is how the application sees the PRESTO proxy.

As an illustration, for a source application that supports only web services according to the Basic Profile 1.0, the PRESTO sender proxy MAY offer to this application an interface of Basic Profile 1.0 operations.

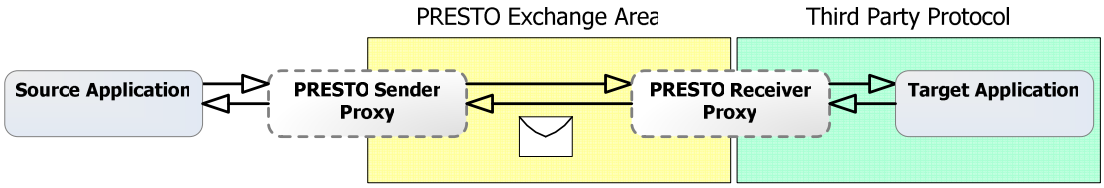


The above methods exposed by the public interface are for illustration purposes only.

The public interface of a PRESTO proxy is concerned with interaction with applications. However, the private implementation, beyond the PRESTO protocol endpoint it exposes in a standardized way to promote interoperability, is concerned with how the service will actually process requests, integrate with other PRESTO processors, return responses and/or acknowledgement, and ensure that – operationally – it provides the correct information to other processors.



1.1.3. Message Exchange with a third party (through another protocol)



This pattern consists in exchanging business data (including one or more binary documents if any) as messages with a third party. The third party is a non PRESTO exchange network based on third party protocol. This pattern only illustrates the case where a third party application is the target destination of an initial PRESTO message.

Interestingly enough, this pattern is fully reversible. In other words, the same pattern can also be used when the third party application is the source application.

The PRESTO receiver proxy acts as a gateway, translating the PRESTO messages from/to third party messages. The routing of the message onto the third party network is not part of the PRESTO protocol and depends on the third party protocol characteristics.

1. Similarly to the previous patterns, the source application invokes the PRESTO sender proxy. The initiating (source) application transfers the business data (including one or more binary documents if any) as well as the identity of the target application.

2. The PRESTO sender proxy generates a PRESTO message containing the XML business data as an opaque payload in the body of the SOAP envelope; the binary documents if any are referenced in the XML business data and transferred as attachments. The PRESTO sender proxy fills the PRESTO technical headers in the SOAP envelope of the PRESTO message and sends the message to the PRESTO receiver proxy.
3. The PRESTO receiver proxy receives the PRESTO message.
4. The PRESTO receiver proxy sends an acknowledgement of receipt to the PRESTO sender proxy, which states that the message has been successfully received.
5. The PRESTO sender proxy receives the acknowledgement of receipt.
6. The PRESTO receiver proxy extracts the XML business data (and the binary documents transmitted as attachment(s) if any) on behalf of the target application and generates, on that basis and accordingly to the third party message translation rules a valid third party message.

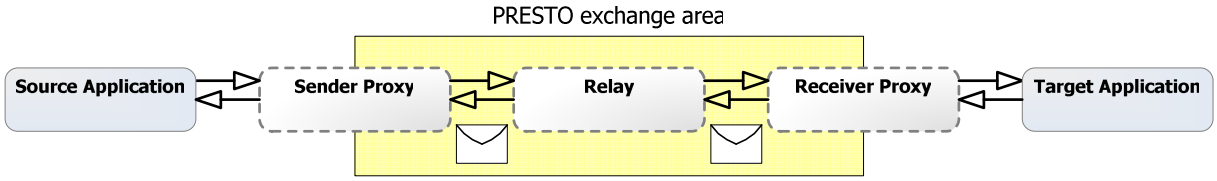
   The third party message translation rules define the logic to map the received PRESTO message to a valid third party message, including the addressing and the routing third party rules. These are not part of the PRESTO protocol. The way the mapping is defined and enforced on both ways is the responsibility of specific PRESTO proxy's implementations. For the pattern requirements, the PRESTO proxy acts as a gateway between the two protocols, i.e. the PRESTO protocol and the third party protocol. It is at least a dual protocol stack.

   The resulting valid third party message is finally routed to the target application inside the third party network.
7. The target application transfers its response (business data) to the PRESTO receiver proxy using the third party protocol with a valid third party message.
8. On that basis, the PRESTO receiver proxy generates a PRESTO message (enforcing the translation rules that define the logic to map the received third party message to a valid PRESTO message) and finally sends it to the originating PRESTO sender proxy.
9. The PRESTO sender proxy receives the PRESTO message.
10. The PRESTO sender proxy sends an acknowledgement of receipt to the PRESTO receiver proxy. The acknowledgement of receipt simply states that the message has been successfully received.
11. The PRESTO receiver proxy receives the acknowledgement of receipt.
12. Eventually, the PRESTO sender proxy extracts the business data (and the binary documents as attachment(s) if any) for the source application. This step ends the message exchange and related session(s).

13. Related PRESTO proxy

1.2. Extensibility points

1.2.1. Message Routing



The Message Routing pattern is an extensibility point that will be addressed in a further version of the PRESTO protocol.

This pattern is detailed in a specific document.

2. Using This Document

In this document we will cover what you need to know and the steps you need to take to support the PRESTO protocol either as a sender proxy or as a receiver proxy.

Following is a brief navigational summary of the parts of this document that are pertinent for each role.

In order to support PRESTO as a *PROXY* (*sender or receiver) proxy*, you will need to implement or use a Web Services stack that supports the following Web service specifications with regards to the PRESTO Technical Reference [PRESTO-Ref]: WS-Addressing [WS-Addressing], MTOM [MTOM], WS-ReliableMessaging [WS-ReliableMessaging], WS-Security [WS-Security].

3. Document Conventions

3.1. 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].

Namespace URI of the general form "some-URI" represents some application dependent or context-dependent URI as defined in RFC 2396 [RFC 2396].

3.2. Namespaces

For brevity of the examples used for illustration in this document and, unless overridden by a namespace inside an XML fragment, we list hereafter the XML namespaces and corresponding prefixes used throughout the document.

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]

4. Addressing Messages

The PRESTO protocol makes mandatory the use of WS-ReliableMessaging which is based on WS-Addressing. Therefore this chapter is only provided as an illustration of the detailed use of addressing headers.

In order to effect the types of message exchange patterns (MEPs) required to enable the PRESTO usage scenarios some form of addressing is required.

The location of a PRESTO receiver proxy can be determined at runtime by querying a local the UDDI repository. This preserves one of the core tenets of a service-oriented architecture – location transparency. The address returned from the query represents the physical location of the PRESTO receiver proxy. This may take the form of a HTTP URL if communication with the PRESTO receiver proxy is over the HTTP protocol (default binding for the PRESTO protocol).

A PRESTO SOAP message that contains business data includes SOAP headers containing the address of the endpoint that is to receive and process the business data. This endpoint is likely to be the address of a PRESTO receiver proxy that is responsible for forwarding the message to the target application. In addition to the endpoint address, a set of properties can be attached that describe the target application required, and any further technical or application-specific properties dependent on the requirements of the target application.

This section describes the use of the WS-Addressing [WS-Addressing] Web Service specification that the PRESTO protocol mandates.

With WS-Addressing, the recipient of the message (the endpoint address) is identified by the wsa:To element, and the operation to be performed on the PRESTO proxy identified by the wsa:Action element. The wsa:From element identifies the source of the message, and the wsa:ReplyTo element specifies where to send replies. Finally, the wsa:FaultTo element identifies where to send errors. Associated with each of the endpoint address elements are optional reference properties represented by the wsa:ReferenceProperties element.

4.1. Expressing PRESTO Proxy addresses

4.1.1. One-Way Message Exchange

This scenario exercises the asynchronous, one-way message exchange pattern (as defined in section § 1.1.1 "One-Way Message Exchange"): a message is sent from a PRESTO sender proxy to a PRESTO receiver proxy.

The PRESTO sender proxy sends a PRESTO message, which carries the request payload in the body. For the illustration purposes, the payload can typically be a unique UTF-8 encoded string.

The PRESTO receiver proxy sends back an HTTP 202 response. This operation is not part of the PRESTO protocol and is simply a consequence of the HTTP binding currently uniquely defined in the PRESTO Technical Reference [PRESTO-Ref].

Message:

```
<s12:Envelope xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
              xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
  <s12:Header>
    <wsa:Action s12:mustUnderstand="1">
      http://tempuri.org/ServicePortType/Foo
    </wsa:Action>
    <wsa:To s12:mustUnderstand="1">
      http://presto-receiver/inbox
    </wsa:To>
  </s12:Header>
  <s12:Body>

    <!—OPAQUE payload to the PRESTO protocol -->
    <ping xmlns="http://tempuri.org/">Hello</ping>

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

HTTP Response: HTTP 202 Accepted

4.1.2. Request-Reply Message Exchange, Anonymous PRESTO Sender Proxy

This scenario exercises the asynchronous, request-reply message exchange pattern (as defined in section § 1.1.2 "Request-Reply Message Exchange"): a message is sent from a PRESTO sender proxy to a PRESTO receiver proxy and then a response is sent from the PRESTO receiver proxy to the PRESTO sender proxy.

For brevity, we have omitted the HTTP acknowledgements that are not part of the PRESTO protocol and are simply a consequence of the HTTP binding currently uniquely defined in the PRESTO Technical Reference [PRESTO-Ref].

WS-Addressing defines a well-known anonymous URI for use by endpoints that cannot have a stable, resolvable URI: http://schemas.xmlsoap.org/ws/2004/08/addressing/anonymous. This is the case here for the reply endpoint. Consequently, the wsa:ReplyTo field is filled with this anonymous constant, which implies a synchronous response from the receiver.

Indeed, requests whose reply endpoint, source endpoint and/or fault endpoint use this address MUST provide some out-of-band mechanism for delivering replies or faults (e.g. returning the reply on the same transport connection). This mechanism may be a simple request/reply transport protocol (e.g., HTTP GET or POST in for the currently supported binding for the PRESTO protocol).

The PRESTO sender proxy sends a PRESTO message, which carries the request payload in the body. For the illustration purposes, the payload can typically be a unique UTF-8 encoded string.

The PRESTO receiver proxy sends back a PRESTO message, which carries a response payload in the body. For the illustration purposes, the payload can typically be a unique UTF-8 encoded response string.

Request:

```
<s12:Envelope xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
              xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
  <s12:Header>
    <wsa:Action s12:mustUnderstand="1">
      http://tempuri.org/ServicePortType/Foo
    </wsa:Action>
    <wsa:To s12:mustUnderstand="1">
      http://presto-receiver/inbox
    </wsa:To>
    <wsa:MessageID>
        urn:uuid:bf121bf2-38b7-4910-b8a3-f8ca65437e33
    </wsa:MessageID>
    <wsa:ReplyTo>
        <wsa:Address>http://schemas.xmlsoap.org/ws/2004/08/addressing/anonymous</wsa:Address>
    </a:ReplyTo>
  </s12:Header>
  <s12:Body>
```

```
    <!—OPAQUE payload to the PRESTO protocol -->

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

Reply:

```
<s12:Envelope xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
              xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
  <s12:Header>
    <wsa:Action s12:mustUnderstand="1">
      http://tempuri.org/ServicePortType/FooResponse
    </wsa:Action>
    <wsa:RelatesTo>
      urn:uuid:bf121bf2-38b7-4910-b8a3-f8ca65437e33
    </wsa:RelatesTo>
  </s12:Header>
  <s12:Body>

    <!—OPAQUE payload to the PRESTO protocol -->

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

4.1.3. Request-Reply Message Exchange, Addressable PRESTO Sender Proxy

This scenario exercises the asynchronous, request-reply message exchange pattern (as defined in section § 1.1.2 "Request-Reply Message Exchange"): a message is sent from a PRESTO sender proxy to a PRESTO receiver proxy and then a response is sent back in turn from the PRESTO receiver proxy to the PRESTO sender proxy.

For brevity, we have omitted the HTTP acknowledgements that are not part of the PRESTO protocol and are simply a consequence of the HTTP binding currently uniquely defined in the PRESTO Technical Reference [PRESTO-Ref].

The PRESTO sender proxy sends a PRESTO message, which carries the request payload in the body. For the illustration purposes, the payload can typically be a unique UTF-8 encoded string.

The PRESTO receiver proxy sends a PRESTO message, which carries a response payload in the body. For the illustration purposes, the payload can typically be a unique UTF-8 encoded response string.

Request:

```
<s12:Envelope xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
              xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
  <s12:Header>
    <wsa:Action s12:mustUnderstand="1">
      http://tempuri.org/ServicePortType/Foo
    </wsa:Action>
    <wsa:To s12:mustUnderstand="1">
```

```
      http://presto-receiver/inbox
    </wsa:To>
    <wsa:MessageID>
      urn:uuid:bf121bf2-38b7-4910-b8a3-f8ca65437e33
    </wsa:MessageID>
    <wsa:ReplyTo>
      <wsa:Address> http://presto-sender/inbox</wsa:Address>
    </wsa:ReplyTo>
  </s12:Header>
  <s12:Body>

    <!—OPAQUE payload to the PRESTO protocol -->

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

Reply:

```
<s12:Envelope xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
              xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
  <s12:Header>
    <wsa:Action s12:mustUnderstand="1">
      http://tempuri.org/ServicePortType/FooResponse
    </wsa:Action>
    <wsa:RelatesTo>
      urn:uuid:bf121bf2-38b7-4910-b8a3-f8ca65437e33
    </wsa:RelatesTo>
  </s12:Header>
  <s12:Body>

    <!—OPAQUE payload to the PRESTO protocol -->

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

4.2. Composing with WS-Security

Taking into account the message routing capability of the PRESTO protocol (see section § 1.2.1 "Message Routing"), security applied at the transport level only protects message between two nodes onto the message path.

Security at the message level aims to overcome the issues with transport-level security, and various specifications introduce message security information into the SOAP header. The most prominent of these specifications is WS-Security [WS-Security] that is mandates by the PRESTO protocol for integrity.

WS-Security defines a set of SOAP headers used to ensure the preservation of integrity messages sent from the initial PRESTO sender proxy to the ultimate PRESTO receiver proxy, and regardless of the networks and intermediary PRESTO relays traversed.

As stated by the PRESTO Technical Reference [PRESTO-Ref], in order to properly preserve the integrity messages, the (opaque) body and all relevant headers need to be included in the signature.

Specifically, the message information headers for addressing MUST be signed with the body in order to "bind" the two together.

Some processors may use message identifiers, (e.g. wsa:MessageID), as part of a uniqueness metric in order to detect replays of messages. Care should be taken to ensure that a unique identifier is actually used.

The message identifier MUST be combined with a wsu:Timestamp.

```
<wsu:Timestamp wsu:Id="uuid-8343c2a7-32fe-4e7f-bb77-d26c2cf0f29e-6">
  <wsu:Created>2006-03-28T21:23:22.614Z</wsu:Created>
  <wsu:Expires>2006-03-28T21:28:22.614Z</wsu:Expires>
</wsu:Timestamp>
```

5. Managing Attachments

When sending (large) binary documents (potentially many MB in size) as part of the business data payload, The PRESTO protocol mandates the use of two specifications by the W3C: XML-binary Optimized Packaging [XOP] and the SOAP Message Transmission Optimization Mechanism [MTOM]. MTOM with XOP allows sending the binary content outside of the SOAP envelope (as a separate part of a multi-part MIME message) without Base64 encoding it.

In place of the Base64-encoded data in the XML InfoSet of the business data, an XOP specific element provides a link to the XOP optimized binary content in the MIME message. The resulting package (after serialization of the InfoSet) is known as a XOP package, and contains the XML document (XOP Document) and the extracted content in a MIME Multipart/Related package.

MTOM describes how to optimize transmission of the SOAP message (in particular the envelope), and relates to the XOP specification where it describes how the optimization is implemented.

The several use cases already identified for the PRESTO protocol require to associate binary documents with a PRESTO message.

The PRESTO sender proxy handles these above operations. The PRESTO receiver proxy that receives the SOAP message is responsible for reconstructing the original message by decoding the optimized binary content back into the Base64 representation, although this is not mandatory.

5.1. Composing with WS-ReliableMessaging

The composition with WS-ReliableMessaging [WS-ReliableMessaging] enables to recover from connection failures when sending (large) binary documents.

The use of WS-ReliableMessaging (see section § 6 "Reliable Delivery of Messages and QoS") enables to detect the failure and automatically re-establish a connection and resend the failed message.

Since WS-ReliableMessaging requires buffering (so it can resend on failure), the recommended approach to send large binary documents is to use chunking. This is where the PRESTO sender proxy divides up for instance a 4GB file into say 1 million messages each 4KB in size and sends them in buffered mode. The PRESTO receiver proxy reconstitutes the file by appending all 1 million messages to form the original 4GB file. Here, WS-ReliableMessaging buffers only a few of the 4KB messages at a time. In case of a failure, WS-ReliableMessaging will automatically resend the 1 or few failed messages.

In terms of implementation, the chunking/dechunking functionality can be encapsulated in the (white box) private side of a PRESTO proxy.

5.2. Composing with WS-Security

As stated by the PRESTO Technical Reference [PRESTO-Ref], in order to properly preserve the integrity messages, all relevant headers, the (opaque) body, and the binary documents need to be included in the signature.

6. Reliable Delivery of Messages and QoS

This section describes the use of the specification WS-ReliableMessaging [WS-ReliableMessaging] by the PRESTO protocol to ensure reliable and in-order delivery of messages.

In essence, a PRESTO sender proxy delivers a message to a PRESTO receiver proxy. The PRESTO receiver proxy acknowledges the receipt by sending a message back to the PRESTO sender proxy. The routing path comes from the various messages used in the WS-ReliableMessaging standard, such as CreateSequence, LastMessage, SequenceAcknowledgement, and TerminateSequence.

At a protocol level, the PRESTO sender proxy, e.g. the Reliable Messaging Source, when it receives a message from the source application, issues a CreateSequence message to the PRESTO receiver proxy, the Reliable Messaging Destination. The destination creates a sequence identifier and returns a CreateSequenceResponse message. The sender then proceeds to deliver one or more messages, each with an incrementing message number. The last message it tags with a wsrm:LastMessage token. The destination then replies with a SequenceAcknowledgement message containing details of the received messages. If some are missing, the source re-transmits the messages with a request to the destination to send an acknowledgement when received. After acknowledgement of all messages by the destination, the source sends a TerminateSequence message.

The following use cases depict the WS-ReliableMessaging handshaking protocol mechanism according to the MEP supported by the PRESTO protocol (see section § 1.1 "PRESTO Message Exchange Patterns (MEP)").

6.1. One-Way Message Exchange, Anonymous PRESTO Sender Proxy

This scenario exercises the first basic form and pattern of reliable message exchange. It uses PRESTO the synchronous, one-way message exchange pattern (as defined in section § 1.1.1 "One-Way Message Exchange") to send a message from a PRESTO sender proxy to a PRESTO receiver proxy.

The PRESTO sender proxy and PRESTO receiver proxy establish a reliable sequence with a simplex CreateSequence/CreateSequenceResponse (CS/CSR) handshake. The CreateSequenceResponse is returned on the HTTP response of the CreateSequence HTTP request.

The PRESTO sender proxy sends a series of three PRESTO messages, each of which carries a payload in the body. For the illustration purposes, the payload can be simply an UTF-8 encoded string.

Messages differ only by the value of the wsrm:MessageID header, the presence or absence of the wsrm:LastMessage indicator, and possibly the body contents that are opaque to the PRESTO protocol.

The PRESTO receiver proxy sends back acknowledgements for each sequence message it receives. All acknowledgements are carried on HTTP responses. Each acknowledgement would typically acknowledge the range in the previous acknowledgement plus the message that was sent on the HTTP request – except in the case that the PRESTO receiver proxy decides not to accept the message.

A PRESTO sender proxy may retransmit certain messages as it deems appropriate or as necessary.

Once the PRESTO sender proxy receives an acknowledgement covering the range 1 through 3, it sends a TerminateSequence message to the PRESTO receiver proxy. The HTTP response for that message should be 202 Accepted. For brevity, we have omitted the HTTP acknowledgements that are not part of the PRESTO protocol and are simply a consequence of the HTTP protocol binding currently uniquely defined in the PRESTO Technical Reference [PRESTO-Ref].

The wsa:ReplyTo and the wsrm:AcksTo EPRs MUST both be anonymous.

CreateSequence Message:

```
<s12:Envelope xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
              xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
  <s12:Header>
    <wsa:Action s12:mustUnderstand="1">
      http://schemas.xmlsoap.org/ws/2005/02/rm/CreateSequence
    </wsa:Action>
    <wsa:ReplyTo>
      <wsa:Address>
        http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
      </wsa:Address>
    </wsa:ReplyTo>
    <wsa:MessageID>
      urn:uuid:addabbbf-60cb-44d3-8c5b-9e0841629a36
    </wsa:MessageID>
    <wsa:To s12:mustUnderstand="1">
      http://presto-receiver/inbox
    </wsa:To>
  </s12:Header>
  <s12:Body>
    <CreateSequence xmlns="http://schemas.xmlsoap.org/ws/2005/02/rm">
      <AcksTo>
        <wsa:Address>
          http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
        </wsa:Address>
      </AcksTo>
    </CreateSequence>
  </s12:Body>
</s12:Envelope>
```

CreateSequenceResponse Message:

```
<s12:Envelope xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
              xmlns:s12="http://www.w3.org/2003/05/soap-envelope">
  <s12:Header>
    <wsa:Action s12:mustUnderstand="1">
      http://schemas.xmlsoap.org/ws/2005/02/rm/CreateSequenceResponse
    </wsa:Action>
    <wsa:RelatesTo>
      urn:uuid:addabbbf-60cb-44d3-8c5b-9e0841629a36
    </wsa:RelatesTo>
    <wsa:To s12:mustUnderstand="1">
      http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
    </wsa:To>
  </s12:Header>
```

```
  <s12:Body>
    <CreateSequenceResponse xmlns="http://schemas.xmlsoap.org/ws/2005/02/rm">
      <Identifier>
        urn:uuid:0afb8d36-bf26-4776-b8cf-8c91fddb5496
      </Identifier>
    </CreateSequenceResponse>
  </s12:Body>
</s12:Envelope>
```

Payload Message:

```
<s12:Envelope xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
              xmlns:wsrm="http://schemas.xmlsoap.org/ws/2005/02/rm"
              xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
  <s12:Header>
    <wsrm:Sequence s12:mustUnderstand="1">
      <wsrm:Identifier>
        urn:uuid:0afb8d36-bf26-4776-b8cf-8c91fddb5496
      </wsrm:Identifier>
      <wsrm:MessageNumber>1</wsrm:MessageNumber>
    </wsrm:Sequence>
    <wsa:Action s12:mustUnderstand="1">urn:wsrm:tbd</wsa:Action>
    <wsa:To s12:mustUnderstand="1">
      http://presto-receiver/inbox
    </wsa:To>
  </s12:Header>
  <s12:Body>

    <!—OPAQUE payload to the PRESTO protocol -->

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

SequenceAcknowledgement Message:

```
<s12:Envelope xmlns:wsrm="http://schemas.xmlsoap.org/ws/2005/02/rm"
              xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
              xmlns:s12="http://www.w3.org/2003/05/soap-envelope">
  <s12:Header>
    <wsrm:SequenceAcknowledgement>
      <wsrm:Identifier>
        urn:uuid:0afb8d36-bf26-4776-b8cf-8c91fddb5496
      </wsrm:Identifier>
      <wsrm:AcknowledgementRange Lower="1" Upper="1" />
    </wsrm:SequenceAcknowledgement>
    <wsa:Action s12:mustUnderstand="1">
      http://schemas.xmlsoap.org/ws/2005/02/rm/SequenceAcknowledgement
    </wsa:Action>
    <wsa:To s12:mustUnderstand="1">
      http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
    </wsa:To>
  </s12:Header>
  <s12:Body></s12:Body>
</s12:Envelope>
```

TerminateSequence Message:

```
<s12:Envelope xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
```

```
            xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
  <s12:Header>
    <wsa:Action s12:mustUnderstand="1">
      http://schemas.xmlsoap.org/ws/2005/02/rm/TerminateSequence
    </wsa:Action>
    <wsa:To s12:mustUnderstand="1">
      http://presto-receiver/inbox
    </wsa:To>
  </s12:Header>
  <s12:Body>
    <TerminateSequence xmlns="http://schemas.xmlsoap.org/ws/2005/02/rm">
      <Identifier>
        urn:uuid:0afb8d36-bf26-4776-b8cf-8c91fddb5496
      </Identifier>
    </TerminateSequence>
  </s12:Body>
</s12:Envelope>
```

6.2. One-Way Message Exchange, Addressable PRESTO Sender Proxy

This scenario exercises the second basic form and pattern of reliable message exchange. It uses PRESTO the asynchronous, one-way message exchange pattern to send a message from a PRESTO sender proxy to a PRESTO receiver proxy.

The PRESTO sender proxy and PRESTO receiver proxy establish a reliable sequence with a simplex CS/CSR handshake. The CreateSequenceResponse is the first message to be sent on the "back channel."

All HTTP responses (the currently available transport protocol binding) – both for the PRESTO sender proxy to a PRESTO receiver proxy – are 202 Accepted. For brevity, we have omitted these HTTP acknowledgements that are not part of the PRESTO protocol and are simply a consequence of the HTTP protocol binding currently uniquely defined in the PRESTO Technical Reference [PRESTO-Ref].

The PRESTO sender proxy sends a series of three PRESTO messages, each of which carries a payload in the body.

Messages differ only by the value of the wsrm:MessageID header, the presence or absence of the wsrm:LastMessage indicator, and possibly the body contents that are opaque to the PRESTO protocol.

The PRESTO receiver proxy sends back acknowledgements for all received messages. The exact number of acknowledgement messages sent back and their location in the overall flow will vary among implementations. The acknowledgement ranges in these acknowledgements may vary, as well.

A PRESTO sender proxy may retransmit certain messages as it deems appropriate or as necessary.

Once the PRESTO sender proxy receives an acknowledgement covering the range 1 through 3, it sends a TerminateSequence message to the PRESTO receiver proxy.

The wsa:ReplyTo and the wsrm:AcksTo EPRs MUST not be anonymous and must be reachable by the PRESTO receiver proxy. The PRESTO receiver proxy must send its messages on a separate HTTP channel.

CreateSequence Message:

```
<s12:Envelope xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
              xmlns:s12="http://www.w3.org/2003/05/soap-envelope">
  <s12:Header>
    <wsa:Action s12:mustUnderstand="1">
      http://schemas.xmlsoap.org/ws/2005/02/rm/CreateSequence
    </wsa:Action>
    <wsa:ReplyTo>
      <wsa:Address>
        http://presto-sender/tempaddr/ef83d068-d4eb-412d-a4ab-9a1cf5843e12/a04a8dc5-6c99-4aa6-
9447-c44ea8b854be
      </wsa:Address>
    </wsa:ReplyTo>
    <wsa:MessageID>
      urn:uuid:2a28544b-75d4-4227-ab15-b54cbfc95332
    </wsa:MessageID>
    <wsa:To s12:mustUnderstand="1">
      http://presto-receiver/inbox
    </wsa:To>
  </s12:Header>
  <s12:Body>
    <CreateSequence xmlns="http://schemas.xmlsoap.org/ws/2005/02/rm">
      <AcksTo>
        <wsa:Address>
          http://presto-sender/tempaddr/ef83d068-d4eb-412d-a4ab-9a1cf5843e12/a04a8dc5-6c99-4aa6-
9447-c44ea8b854be
        </wsa:Address>
      </AcksTo>
    </CreateSequence>
  </s12:Body>
</s12:Envelope>
```

CreateSequenceResponse Message:

```
<s12:Envelope xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
              xmlns:s12="http://www.w3.org/2003/05/soap-envelope">
  <s12:Header>
    <wsa:Action s12:mustUnderstand="1">
      http://schemas.xmlsoap.org/ws/2005/02/rm/CreateSequenceResponse
    </wsa:Action>
    <wsa:RelatesTo>
      urn:uuid:2a28544b-75d4-4227-ab15-b54cbfc95332
    </wsa:RelatesTo>
    <wsa:To s12:mustUnderstand="1">
```

```
      http://presto-sender/tempaddr/ef83d068-d4eb-412d-a4ab-9a1cf5843e12/a04a8dc5-6c99-4aa6-9447-
c44ea8b854be
    </wsa:To>
  </s12:Header>
  <s12:Body>
    <CreateSequenceResponse xmlns="http://schemas.xmlsoap.org/ws/2005/02/rm">
      <Identifier>
        urn:uuid:c5fe4fe1-c9c3-447e-af87-0406e049a86b
      </Identifier>
    </CreateSequenceResponse>
  </s12:Body>
</s12:Envelope>
```

Payload Message:

```
<s12:Envelope xmlns:wsrm="http://schemas.xmlsoap.org/ws/2005/02/rm"
              xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
              xmlns:s12="http://www.w3.org/2003/05/soap-envelope">
  <s12:Header>
    <wsrm:AckRequested>
      <wsrm:Identifier>
        urn:uuid:c5fe4fe1-c9c3-447e-af87-0406e049a86b
      </wsrm:Identifier>
    </wsrm:AckRequested>
    <wsrm:Sequence s12:mustUnderstand="1">
      <wsrm:Identifier>
        urn:uuid:c5fe4fe1-c9c3-447e-af87-0406e049a86b
      </wsrm:Identifier>
      <wsrm:MessageNumber>1</wsrm:MessageNumber>
    </wsrm:Sequence>
    <wsa:Action s12:mustUnderstand="1">urn:wsrm:tbd</wsa:Action>
    <wsa:From>
      <wsa:Address>
        http://presto-sender/tempaddr/ef83d068-d4eb-412d-a4ab-9a1cf5843e12/a04a8dc5-6c99-4aa6-
9447-c44ea8b854be
      </wsa:Address>
    </wsa:From>
    <wsa:To s12:mustUnderstand="1">
      http://presto-receiver/inbox
    </wsa:To>
  </s12:Header>
  <s12:Body>

    <!—OPAQUE payload to the PRESTO protocol -->

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

Sequence Acknowledgement Message:

```
<s12:Envelope xmlns:wsrm="http://schemas.xmlsoap.org/ws/2005/02/rm"
              xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
              xmlns:s12="http://www.w3.org/2003/05/soap-envelope">
  <s12:Header>
    <wsrm:SequenceAcknowledgement>
      <wsrm:Identifier>
        urn:uuid:c5fe4fe1-c9c3-447e-af87-0406e049a86b
```

```
      </wsrm:Identifier>
      <wsrm:AcknowledgementRange Lower="1" Upper="1"></wsrm:AcknowledgementRange>
    </wsrm:SequenceAcknowledgement>
    <wsa:Action s12:mustUnderstand="1">
      http://schemas.xmlsoap.org/ws/2005/02/rm/SequenceAcknowledgement
    </wsa:Action>
    <wsa:To s12:mustUnderstand="1">
      http://presto-sender/tempaddr/ef83d068-d4eb-412d-a4ab-9a1cf5843e12/a04a8dc5-6c99-4aa6-9447-
c44ea8b854be
    </wsa:To>
  </s12:Header>
  <s12:Body></s12:Body>
</s12:Envelope>
```

TerminateSequence Message:

```
<s12:Envelope xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
              xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
  <s12:Header>
    <wsa:Action s12:mustUnderstand="1">
      http://schemas.xmlsoap.org/ws/2005/02/rm/TerminateSequence
      </wsa:Action>
      <wsa:From>
        <wsa:Address>
          http://presto-sender/tempaddr/ef83d068-d4eb-412d-a4ab-9a1cf5843e12/a04a8dc5-6c99-4aa6-
9447-c44ea8b854be
        </wsa:Address>
      </wsa:From>
      <wsa:To s12:mustUnderstand="1">
        http://presto-receiver/inbox
      </wsa:To>
    </s12:Header>
  <s12:Body>
    <TerminateSequence xmlns="http://schemas.xmlsoap.org/ws/2005/02/rm">
      <Identifier>
        urn:uuid:c5fe4fe1-c9c3-447e-af87-0406e049a86b
      </Identifier>
    </TerminateSequence>
  </s12:Body>
</s12:Envelope>
```

6.3. Request-Reply Message Exchange, Addressable PRESTO Sender Proxy

This scenario exercises a more complex reliable messaging situation – one that requires a duplex sequence to support a request-reply messaging pattern (as defined in section § 1.1.2 "Request-Reply Message Exchange"): a message is sent from a PRESTO sender proxy to a PRESTO receiver proxy and then a response is sent back in turn from the PRESTO receiver proxy to the PRESTO sender proxy.

The PRESTO sender proxy must be addressable and reachable by the PRESTO receiver proxy to enable this scenario. The PRESTO sender proxy then sends messages to the PRESTO receiver proxy and expects message responses in return.

The PRESTO sender proxy and the PRESTO receiver proxy establish a duplex WS-RM sequence with a duplex CS/CSR handshake, i.e. wsrm:Offer and wsrm:Accept are present.

The PRESTO sender proxy then transmits three PRESTO message requests to the PRESTO receiver, each of which carries a payload in the body. For the illustration purposes, the payload can be simply an UTF-8 encoded string.

The HTTP response is 202 Accepted every time. For brevity, we have omitted the HTTP acknowledgements that are not part of the PRESTO protocol and are simply a consequence of the HTTP binding currently uniquely defined in the PRESTO Technical Reference [PRESTO-Ref].

When a request is processed, the PRESTO receiver proxy replies with a response. Notice that the PRESTO sender proxy will typically wait for the PRESTO receiver proxy to respond before initiating a new request.

When acknowledging messages, the PRESTO sender proxy and the PRESTO receiver proxy may send stand-alone sequence acknowledgements or piggy-back the SequenceAcknowledgement header on a message that is conveniently being transmitted at the same time or close by.

Once the PRESTO sender proxy received an acknowledgement for all its requests and received and acknowledged all the replies, it will send a stand-alone message carrying the wsrm:LastMessage indicator. Alternatively, a PRESTO sender proxy could piggy-back the wsrm:LastMessage indicator on the last request it sends. Once the PRESTO sender proxy receives an acknowledgement acknowledging the entire range of requests, including the last message, it will send a TerminateSequence message to the PRESTO receiver proxy and consider the PRESTO sequence complete.

A PRESTO receiver proxy would at this point typically do the same – send a wsrm:LastMessage when all its replies were acknowledged, then send a TerminateSequence PRESTO sequence when an acknowledgement for the entire range, including its wsrm:LastMessage, is received.

CreateSequence Message:

```
<s12:Envelope xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
              xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
  <s12:Header>
    <wsa:Action s12:mustUnderstand="1">
      http://schemas.xmlsoap.org/ws/2005/02/rm/CreateSequence
    </wsa:Action>
    <wsa:ReplyTo>
      <wsa:Address>
```

```
        http://presto-sender/tempaddr/96f6d890-b7d5-48ad-a853-d6395a3a3d2e/cd154e2c-ef0d-4ddb-
bac9-adcda6bb7906
      </wsa:Address>
    </wsa:ReplyTo>
    <wsa:MessageID>
      urn:uuid:9bddb76a-59cc-4e5b-a792-c6646998bf66
    </wsa:MessageID>
    <wsa:To s12:mustUnderstand="1">
      http://presto-receiver/inbox
    </wsa:To>
  </s12:Header>
  <s12:Body>
    <CreateSequence xmlns="http://schemas.xmlsoap.org/ws/2005/02/rm">
      <AcksTo>
        <wsa:Address>
          http://presto-sender/tempaddr/96f6d890-b7d5-48ad-a853-d6395a3a3d2e/cd154e2c-ef0d-4ddb-
bac9-adcda6bb7906
        </wsa:Address>
      </AcksTo>
      <Offer>
        <Identifier>
          urn:uuid:5f7d5761-d6f7-446f-bea6-989a926ae712
        </Identifier>
      </Offer>
    </CreateSequence>
  </s12:Body>
</s12:Envelope>
```

CreateSequenceResponse Message:

```
<s12:Envelope xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
              xmlns:s12="http://www.w3.org/2003/05/soap-envelope">
  <s12:Header>
    <wsa:Action s12:mustUnderstand="1">
      http://schemas.xmlsoap.org/ws/2005/02/rm/CreateSequenceResponse
    </wsa:Action>
    <wsa:RelatesTo>
      urn:uuid:9bddb76a-59cc-4e5b-a792-c6646998bf66
    </wsa:RelatesTo>
    <wsa:To s12:mustUnderstand="1">
      http://presto-sender/tempaddr/96f6d890-b7d5-48ad-a853-d6395a3a3d2e/cd154e2c-ef0d-4ddb-bac9-
adcda6bb7906
    </wsa:To>
  </s12:Header>
  <s12:Body>
    <CreateSequenceResponse xmlns="http://schemas.xmlsoap.org/ws/2005/02/rm">
      <Identifier>
        urn:uuid:3c0ba104-815c-4d83-b2bb-47670a5f8083
      </Identifier>
      <Accept>
        <AcksTo>
          <wsa:Address>
            http://presto-receiver/inbox
          </wsa:Address>
        </AcksTo>
      </Accept>
    </CreateSequenceResponse>
```

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

Payload Message:

```
<s12:Envelope xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
              xmlns:wsrm="http://schemas.xmlsoap.org/ws/2005/02/rm"
              xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
  <s12:Header>
    <wsrm:AckRequested>
      <wsrm:Identifier>
        urn:uuid:3c0ba104-815c-4d83-b2bb-47670a5f8083
      </wsrm:Identifier>
    </wsrm:AckRequested>
    <wsrm:Sequence s12:mustUnderstand="1">
      <wsrm:Identifier>
        urn:uuid:3c0ba104-815c-4d83-b2bb-47670a5f8083
      </wsrm:Identifier>
      <wsrm:MessageNumber>1</wsrm:MessageNumber>
    </wsrm:Sequence>
    <wsa:Action s12:mustUnderstand="1">urn:wsrm:tbd</wsa:Action>
    <wsa:MessageID>
      urn:uuid:db82b0df-3966-476b-b605-e26d9900d538
    </wsa:MessageID>
    <wsa:ReplyTo>
      <wsa:Address>
        http://presto-sender/tempaddr/96f6d890-b7d5-48ad-a853-d6395a3a3d2e/cd154e2c-ef0d-4ddb-
bac9-adcda6bb7906
      </wsa:Address>
    </wsa:ReplyTo>
    <wsa:From>
      <wsa:Address>
        http://presto-sender/tempaddr/96f6d890-b7d5-48ad-a853-d6395a3a3d2e/cd154e2c-ef0d-4ddb-
bac9-adcda6bb7906
      </wsa:Address>
    </wsa:From>
    <wsa:To s12:mustUnderstand="1">
      http://presto-receiver/inbox
    </wsa:To>
  </s12:Header>
  <s12:Body>

    <!—OPAQUE payload to the PRESTO protocol -->

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

Response Message:

```
<s12:Envelope xmlns:wsrm="http://schemas.xmlsoap.org/ws/2005/02/rm"
              xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
              xmlns:s12="http://www.w3.org/2003/05/soap-envelope">
  <s12:Header>
    <wsrm:SequenceAcknowledgement>
      <wsrm:Identifier>
        urn:uuid:3c0ba104-815c-4d83-b2bb-47670a5f8083
      </wsrm:Identifier>
```

```
      <wsrm:AcknowledgementRange Lower="1" Upper="1"></wsrm:AcknowledgementRange>
    </wsrm:SequenceAcknowledgement>
    <wsrm:AckRequested>
      <wsrm:Identifier>
        urn:uuid:5f7d5761-d6f7-446f-bea6-989a926ae712
      </wsrm:Identifier>
    </wsrm:AckRequested>
    <wsrm:Sequence s12:mustUnderstand="1">
      <wsrm:Identifier>
        urn:uuid:5f7d5761-d6f7-446f-bea6-989a926ae712
      </wsrm:Identifier>
      <wsrm:MessageNumber>1</wsrm:MessageNumber>
    </wsrm:Sequence>
    <wsa:Action s12:mustUnderstand="1">
      urn:wsrm:EchoStringResponse
    </wsa:Action>
    <wsa:RelatesTo>
      urn:uuid:db82b0df-3966-476b-b605-e26d9900d538
    </wsa:RelatesTo>
    <wsa:To s12:mustUnderstand="1">
      http://presto-sender/tempaddr/96f6d890-b7d5-48ad-a853-d6395a3a3d2e/cd154e2c-ef0d-4ddb-bac9-
adcda6bb7906
    </wsa:To>
  </s12:Header>
  <s12:Body>

    <!—OPAQUE message response to the PRESTO protocol -->

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

PRESTO sender proxy's Acknowledgement Message:

```
<s12:Envelope xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
              xmlns:wsrm="http://schemas.xmlsoap.org/ws/2005/02/rm"
              xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
  <s12:Header>
    <wsrm:SequenceAcknowledgement>
      <wsrm:Identifier>
        urn:uuid:5f7d5761-d6f7-446f-bea6-989a926ae712
      </wsrm:Identifier>
      <wsrm:AcknowledgementRange Lower="1" Upper="1"></wsrm:AcknowledgementRange>
    </wsrm:SequenceAcknowledgement>
    <wsa:Action s12:mustUnderstand="1">
      http://schemas.xmlsoap.org/ws/2005/02/rm/SequenceAcknowledgement
    </wsa:Action>
    <wsa:From>
      <wsa:Address>
        http://presto-sender/tempaddr/96f6d890-b7d5-48ad-a853-d6395a3a3d2e/cd154e2c-ef0d-4ddb-
bac9-adcda6bb7906
      </wsa:Address>
    </wsa:From>
    <wsa:To s12:mustUnderstand="1">
      http://presto-receiver/inbox
    </wsa:To>
  </s12:Header>
```

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

PRESTO sender proxy's LastMessage Message:

```
<s12:Envelope xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
              xmlns:wsrm="http://schemas.xmlsoap.org/ws/2005/02/rm"
              xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
  <s12:Header>
    <wsrm:SequenceAcknowledgement>
      <wsrm:Identifier>
        urn:uuid:5f7d5761-d6f7-446f-bea6-989a926ae712
      </wsrm:Identifier>
      <wsrm:AcknowledgementRange Lower="1" Upper="3"></wsrm:AcknowledgementRange>
    </wsrm:SequenceAcknowledgement>
    <wsrm:Sequence s12:mustUnderstand="1">
      <wsrm:Identifier>
        urn:uuid:3c0ba104-815c-4d83-b2bb-47670a5f8083
      </wsrm:Identifier>
      <wsrm:MessageNumber>4</wsrm:MessageNumber>
      <wsrm:LastMessage></wsrm:LastMessage>
    </wsrm:Sequence>
    <wsa:Action s12:mustUnderstand="1">
      http://schemas.xmlsoap.org/ws/2005/02/rm/LastMessage
    </wsa:Action>
    <wsa:From>
      <wsa:Address>
        http://presto-sender/tempaddr/96f6d890-b7d5-48ad-a853-d6395a3a3d2e/cd154e2c-ef0d-4ddb-
bac9-adcda6bb7906
      </wsa:Address>
    </wsa:From>
    <wsa:To s12:mustUnderstand="1">
      http://presto-receiver/inbox
    </wsa:To>
  </s12:Header>
  <s12:Body></s12:Body>
</s12:Envelope>
```

PRESTO receiver proxy's TerminateSequence Message:

```
<s12:Envelope xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
              xmlns:s12="http://www.w3.org/2003/05/soap-envelope">
  <s12:Header>
    <wsa:Action s12:mustUnderstand="1">
      http://schemas.xmlsoap.org/ws/2005/02/rm/TerminateSequence
    </wsa:Action>
    <wsa:To s12:mustUnderstand="1">
      http://presto-sender/tempaddr/96f6d890-b7d5-48ad-a853-d6395a3a3d2e/cd154e2c-ef0d-4ddb-bac9-
adcda6bb7906
    </wsa:To>
  </s12:Header>
  <s12:Body>
    <TerminateSequence xmlns="http://schemas.xmlsoap.org/ws/2005/02/rm">
      <Identifier>
        urn:uuid:5f7d5761-d6f7-446f-bea6-989a926ae712
      </Identifier>
    </TerminateSequence>
```

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

6.4. Composing with WS-Security

As stated by the PRESTO Technical Reference [PRESTO-Ref], in order to properly preserve the integrity messages, the (opaque) body and all relevant headers need to be included in the signature.

Specifically, the message information headers for reliable messaging MUST be signed with the body in order to "bind" the two together. The wsrm:Sequence header MUST be signed with the body and the wsrm:SequenceAcknowledgement header MAY be signed independently because a reply independent of the message is not a security concern.

7. Security

This section describes the use of the security specifications (PRESTO protocol endorses the WS-I Basic Security Profile (BSP) for message level security (http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html)). 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.

The PRESTO Technical Reference [PRESTORef] has two levels of security: message signature and message encryption.

The following chapters illustrate the application of the two different levels of message security that MAY be applied on the PRESTO use cases of this document. This section doesn't include the specific headers depending on the specifications used.

7.1. Message Signature

In order to achieve security, the PRESTO sender proxy and the PRESTO receiver proxy must have a dedicated public/private key and must have each other's certificate in their key store.

In a signed exchange, all outgoing messages are signed with the private key of the source, which can be whether a PRESTO sender or a PRESTO receiver proxy.

7.1.1. X509 Mutual Authentication, Sign Only

The PRESTO sender proxy uses his private key to sign the outgoing message body and some message header elements. The header elements to sign may vary according to the specifications used and are described in the different chapters "Composing with WS-Security" and in the PRESTO Technical Reference [PRESTORef], some header elements must be signed with the body.

The PRESTO receiver proxy receives the signed message and verifies the signature in order to ensure the integrity of the received data.

Symmetrically the same rule MUST be applied when the PRESTO receiver proxy sends a response message to the PRESTO sender proxy. In this case the receiver proxy signs the messages using his own private key.

7.2. Message Encryption

In order to achieve security, the PRESTO sender proxy and the PRESTO receiver proxy must have a dedicated public/private key and must have each other's certificate in their key store.

In an encrypted exchange, all outgoing messages are signed with the private key of the source and encrypted with the public key (from X509 certificate) of the target, where source and target can be whether a PRESTO sender or a PRESTO receiver proxy.

7.2.1. X509 Mutual Authentication, Sign then Encrypt

The PRESTO sender proxy uses his private key to sign the outgoing message body and some message header elements. The header elements to sign may vary according to the specifications used and are described in the different chapters "Composing with WS-Security" and in the PRESTO Technical Reference [PRESTORef], some header elements must be signed with the body.

After singing, the PRESTO sender proxy encrypts the message body using the PRESTO receiver public key (from X509 certificate).

When receiving the message, The PRESTO decrypts the message then verifies the signature in order to ensure the integrity of the received data.

Symmetrically the same rule MUST be applied when the PRESTO receiver proxy sends a response message to the PRESTO sender proxy. In this case the receiver proxy signs the messages using his private key and encrypts using the PRESTO sender proxy's public key (from 509 certificate).

Different algorithms MAY be used for encryption, this example uses AES256.

8. References

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

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

[PRESTO-Proxy]
"PRESTO Proxy Specifications," June 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 A – Glossary

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.

Principal

Any system entity that can be granted security rights or that makes assertions about security or identity.

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.