Provisional Envelope Specification

January 22, 2001

This version:
TSC-ProvisionalEnvelopeSpec-v01.00
Latest version:
01.00
Previous version:
N/A
Editor:
N/A
Authors:
Alan Coleman, Vivant Corp (acoleman@vivant.com)
Kim Bartkus, HR-XML Consortium (kim@hr-xm.org)
Contributors:
Simon Bray, Opus360
Alan Sproat, Jobs.com

Abstract

This document lays out a provisional XML envelope used for data exchange between HR-XML member organizations.

This proposal is intended to allow business partners to implement HR-XML in the absence of either de facto or de jure standards for envelope and transport in the B2B Internet environment.

Status of this Document

This document is a Technical Note and is not to be referenced as a formal recommendation by any party.

The key words "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.

Table of Contents

1 Introduction

This document lays out a provisional XML envelope used for data exchange between HR-XML member organizations.

This proposal is intended to allow business partners to implement HR-XML in the absence of either de facto or de jure standards for envelope and transport in the B2B Internet environment.

There are three layers involved in exchanging an XML payload between two parties:

1. The payload
2. The envelope
3. The transport

The hackneyed analogy of the physical mail system applies. There's a mailman who carries envelopes around, an envelope with routing information, and the actual letter itself.

For our purposes, an XML document conforming to an HR-XML DTD is the payload. The rest of this document discusses our provisional recommendation for the envelope and the transport.

Just as the TCP/IP network protocol has layer-specific headers in its protocol, so should we. The figure below shows the general idea.

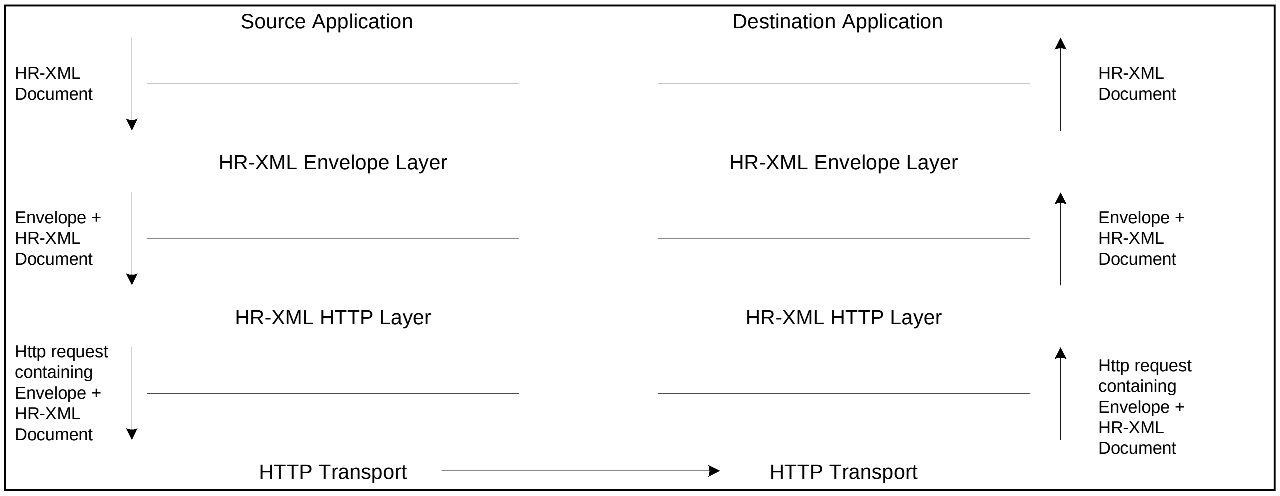


Figure 1. Protocol Stack Diagram for HR-XML Header Processing

1.1 Purpose and Scope

This specification is not intended to be a fully fleshed out proposal for a long-term transport solution. It is intended to be sufficient for use until a leader emerges among the current crop of proposed standards.

1.2 Goals

The provisional envelope has the following design requirements.

- Be robust and complete enough for members to create production-quality integrations.
- Be simple enough for business partners to use easily.
- Avoid dependencies between this envelope/transport mechanism and HR-XML's core content standards.

1.3 Relationship to Other Specifications

Document Version Status Defined v01.00

ISO Specifications v00.03

W3C RFC 2119 - Key words for use in RFCs to Indicate Requirement Levels

2 Provisional Envelope Design

2.1 Envelope Format

For the envelope to be useful it requires at least the following items:

1. Sender identification/credential
2. Action to be performed
3. The payload

The following items are also needed for it to be sufficiently robust to use in a production context:

1. Recipient ID
2. Sender's transaction ID
3. Status Code/Error Info
4. Manifest
5. Timestamp
6. Version Information

In addition, it should be possible to include more than one message in a single envelope. The DTD/Schema Design section of this document describes these items in detail.

2.2 Payload Packaging

There is always a separation between envelope header information and the payload – the actual content document being sent. The question is, how strictly enforced is this separation?

There are two general approaches to packaging the payload within an envelope. One enforces opacity of the payload within the envelope – applying the tcp/ip style layering approach. The other keeps the two distinct, but does not enforce anything.

In the first approach, the envelope is an XML document, and the payload is a distinct XML document contained within it. The payload document is contained in a PCDATA section, as "arbitrary" content. This forces the envelope to be parsed, the payload to be extracted, and then the payload to be parsed separately.

This hard separation accomplishes several things:

1. The envelope can be parsed and interpreted without knowing anything about the contents.
2. The envelope can be modified (by an intermediate node, for example) with confidence that no inadvertent modifications will be made to the payload.
3. When processing the payload, only payload is available.
4. The envelope XML references the envelope DTD, and the payload XML references the payload DTD, and the two are completely separate.
5. The payload could actually be anything, not just an XML document.

In short, this approach enforces a layered processing model analogous to a network protocol stack.

In the second approach, the payload is treated as an XML element. This removes the hard separation between envelope and payload, with the following consequences:

1. Only one parse is needed. Both envelope and payload are parsed together as a single document.
2. The envelope DTD and the payload DTD must both be available before the document can be parsed.
3. Separation of envelope and payload cannot be enforced. Care must be taken when modifying either one.

For this provisional transport specification, we use a PCData-based payload approach.

2.2.1 PCDATA Payload Approach

The simplified tagging below shows the structure of the documents in this approach. As described above, the algorithm for processing it is:

1. Parse envelope
2. Validate envelope
3. Extract payload
4. Parse/process payload

```
<?xml version="1.0" ?>
   <!DOCTYPE Envelope SYSTEM “http://foo.com /Envelope.dtd">
   <Envelope>
   <HeaderStuff/>
   <Payload><![CDATA[
   <?xml version="1.0" ?>
   <!DOCTYPE Envelope SYSTEM “http://foo.com /SomePayload.dtd">
   <someotherdocument/>
   ]]>
   </Payload>
   </Envelope>
```

One caveat needs to be borne in mind when this approach is used: the payload document MUST NOT contain any PCDATA sections itself. This occurs because the XML parser will interpret the first “]]>” sequence it encounters as the end of the PCDATA section.

2.3 Security

This document takes a very simple approach to security. The envelope contains elements that allow business partners to identify themselves, and we recommend use of SSL as a secure protocol.

This addresses two of the primary security requirements, authentication and encryption.

This document does not try to address non-repudiation.

2.4 Versioning

For versioning to be effective and easy to use, we must address the following requirements:

1. The recipient of an envelope should be able to determine the version of that envelope
2. The recipient should also be able to parse the envelope and its contents regardless of the version

The common approaches to versioning involve two strategies:

- Encoding a version number in the DTD name

- Including a version element or attribute in the DTD

In order to accomplish both of the requirements listed above, we do both of these things.

The only way to ensure that all versions of a DTD remain available is to encode the version number in the name of the DTD. For example, using a naming convention like hrxmlenvelope-v01.00.dtd and hrxmlenvelope-v02.00.dtd allows us to keep all versions of the DTD available in the same repository.

It is difficult for the recipient of a document to retrieve the correct version number if we use only this approach, though (and it's especially difficult if we use the element-safe envelope wrapper approach outlined earlier in this paper).

Therefore, we should also add a version field to the envelope. This specifies the version of the envelope itself, not the version of the payload. The same approach to versioning can be applied to HR-XML payload DTD's if we choose.

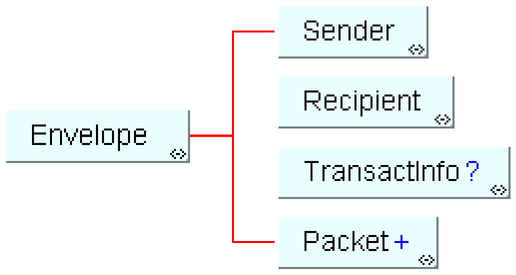
The actual version numbers used will conform to the HR-XML version format recommendation (see the Related Documents section earlier in this document).

3 DTD/Schema Design

This section describes the DTD structure of the envelope.

3.1 DTD Diagrams

3.1.1 Top Level Structure



The Envelope has four top-level elements – *Sender*, *Recipient*, *TransactInfo* and *Packet*. All but *TransactInfo* are mandatory elements. *Packet* is repeatable.

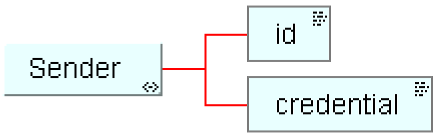
Attributes:

The top-level Envelope element has one attribute, *version*.

version

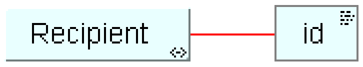
This attribute identifies the version of the envelope. The only valid value at this time is "01.00". Note that this is the version of the envelope itself, not the version of the payload document.

3.1.2 Sender



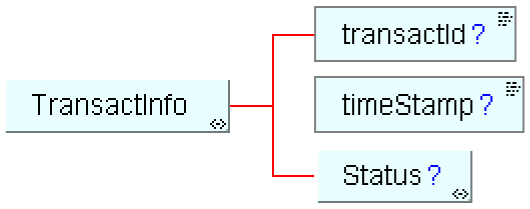
An identifier for the sender is included in *id*, with an associated qualifying password, phrase or other shared secret in *credential*.

3.1.3 Recipient



An identifier for the recipient is included in *id*. This helps to ensure that the data has been sent to the correct destination. The *id* could be either the URL of the recipient process/service or an email address – it is up to the envelope parser to confirm.

3.1.4 TransactInfo



transactId?
An identifier for the transaction can be included in the optional *transactId*. This identifies the envelope transaction and not any of the constituent data, since there could be multiple documents within one envelope. Note that because each envelope has a unique *transactId* and each packet has a unique *packetId*, the combination of *transactId* and *packetId* creates a globally unique identifier for each packet.

timeStamp?
A timestamp can be added in the optional *timestamp*. This date SHOULD conform to ISO standard 8601 format.

Status?

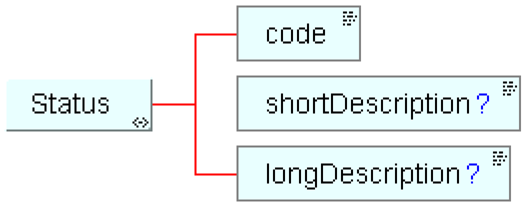
The optional *Status* element may be present when the type is *response*. This indicates the outcome of a requested transaction.

Attributes:

transactType (data | *request* | *response* | *multi)*

This optional attribute signals whether the envelope contains actual data, a request, a response or a combination of these. It helps provide the envelope parser with a heads up on what to expect in the payloads.

3.1.5 Status



The *Status* element contains response information about a transaction or payload. A transaction or packet whose type is *response* may have a Status element in it, which describes the processing status of the referenced transaction or packet ID.

code
This element contains a numeric code indicating success or some variation on failure. See the HR-XML Staffing Exchange Protocol Specification for an initial list of status codes.

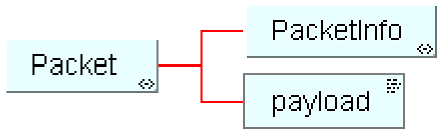
shortDescription?

This element contains a brief description of the error.

longDescription?

This element contains a longer and more complete description of the error. If the short description is not sufficient to pinpoint the cause, this description should be.

3.1.6 Packet



The *Packet* element is used to store metadata for each payload, along with each payload itself. The same functionality could be applied using one payload element and several attributes but this would be unscalable and could quickly become unwieldy.

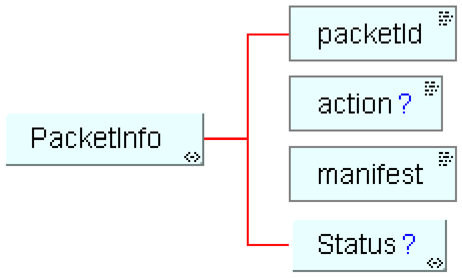
PacketInfo

The PacketInfo element is described below.

payload

The *payload* element contains one HR-XML document inside a CDATA section. This approach allows the payload to be parsed separately from the envelope and avoids any interdependency between envelope and contents when validating envelope documents.

3.1.7 PacketInfo



The *PacketInfo* element describes the contents of a packet.

packetId
This element is mandatory inside the *PacketInfo* element and uniquely identifies the contents of the payload. *packetId* should be unique within each envelope, and the combination of *packetId* and *transactId* constitute a globally unique identifier for each packet.

action?

The optional action element signals the action that should be performed on the payload. The various HR-XML content specifications define the actions needed for their document exchanges.

manifest
The contents of the payload are identified by the mandatory manifest element. This allows the envelope parser to route the payload accordingly without parsing it. One approach is to use the URL of the payload document's DTD.

Status?

This optional element indicates the success/failure status on a response message.

Attributes:

packetType (data | *request* | *response)*

This optional attribute describes the content of the packet. A *packetType* of *data*, the default, describes the packet as containing an XML document. A value of *response* signifies that a S*tatus* element should also be present and that the payload can and probably will be empty. A value of *request* should coexist only with an *action* attribute with a value meaning *get* or *find*.

4 Implementation Considerations

4.1 HTTP Transport

We currently recommend http using simulated form transmission as a transport. Simulated http form submission is the simplest for most business partners to implement.

The following is an example of a simulated http form transmission:

```
POST /foo/bar/somethingorother HTTP/1.0
Content-type: application/x-www-form-urlencoded
Content-length: bignumber

HRXMLDoc=%3Cxml… whole mess here %3C%2Fhrxmlenvelope%3E
```

This approach has the advantage that a fairly standard form-processing engine can forward this to the appropriate code, but it also requires the xml document to be url-encoded, which is a bit of a pain.

4.2 HRXMLDoc Form Field

The name of the http form field in which the HR-XML document is passed is "HRXMLDoc". So http/html form processing software should simply retrieve the value of this form field in order to get the document a partner sent.

This field will contain an XML document that conforms to the envelope described above. The envelope will contain one of the HR-XML standard XML documents as its payload.

4.3 Security

As described earlier in this document, SSL or similar protocol should be used to ensure that sensitive data is encrypted.

5 Open/Closed Issues

To be supplied.

6 References

To be supplied.

7 Appendix A Document Version History

Version	Date	Description
N/A	May 6 2000	Initial version
N/A	July 5 2000	Incorporate Opus extensions
N/A	Sept. 22 2000	Miscellaneous updates after TSC review.
V01.00	Oct. 6, 2000	Initial Draft Specification

8 Appendix B Reference Examples

8.1 Example Request Envelope:

```
<?xml version = "1.0"?>
<!DOCTYPE Envelope SYSTEM "Envelope-v01-00.dtd">
<Envelope version = "01.00">
	<Sender>
		<id>joebob@briggs.com</id>

		<credential>drive-in</credential>
	</Sender>
	<Recipient>
		<id>ethyl@mertz.com</id>
	</Recipient>
	<TransactInfo transactType = "request">
		<transactId>0037</transactId>
		<timeStamp>2000-10-09T14:14:11Z</timeStamp>
	</TransactInfo>
	<Packet>
		<PacketInfo packetType = "request">
			<packetId>1</packetId>
			<action>dothathrxmlthang</action>
			<manifest>whizzyhrxmlthingy.dtd</manifest>
		</PacketInfo>
		<payload><![CDATA[
<?xml version="1.0"?>
<!DOCTYPE whizzyhrxmlthingy SYSTEM "whizzyhrxmlthingy.dtd">
<WhizzyHRXMLThingy>
```

```
          etc etc
          HR-XML content goes here
</WhizzyHRXMLThingy>
]]></payload>
          </Packet>
</Envelope>
```

8.2 Example Response Envelope

This section shows a possible response envelope for the request above.

```
<?xml version="1.0" ?>
<!DOCTYPE Envelope SYSTEM "Envelope-v01-00.dtd">
          <Envelope version="01.00">
          <Sender>
                    <id>ethyl@mertz.com</id>
                    <credential>lilricky</credential>
          </Sender>
          <Recipient>
                    <id>joebob@briggs.com</id>
          </Recipient>
          <TransactInfo transactType="response">
                    <transactId>0037</transactId>
                    <timeStamp>2000-10-09T14:15:23Z</timeStamp>
          </TransactInfo>
          <Packet>
                    <PacketInfo packetType="response">
                    <packetId>1</packetId>
                    <manifest />
                    <Status>
                              <code>200</code>
                              <shortDescription>Success</shortDescription>
                    </Status>
                    </PacketInfo>
          <payload />
          </Packet>
</Envelope>
```

9 Appendix C Envelope DTD

Below is the actual envelope DTD.

```
<!---->
<!--The Envelope has four top-level elements: Sender, Recipient, TransactInfo and Packet. All but
TransactInfo are mandatory elements. Packet is repeatable.-->
<!ELEMENT Envelope  (Sender , Recipient , TransactInfo? , Packet+ )>

<!--This attribute identifies the version of the envelope.  The only valid value at this time is '01.00'.  Note that
this is the version of the envelope itself, not the version of the payload document.-->
<!ATTLIST Envelope version CDATA  #FIXED "01.00">

<!--An identifier for the sender is included in id, with an associated qualifying password, phrase or other
shared secret in credential.-->
<!ELEMENT Sender  (id , credential )>

<!--An identifier for the recipient is included in id. This helps to ensure that the data has been sent to the
correct destination. The id could be either the URL of the recipient process/service or an email address.-->
<!ELEMENT Recipient  (id )>

<!--The Packet element is used to store metadata for each payload, along with each payload itself. -->
<!ELEMENT Packet  (PacketInfo , payload )>

<!--TransactInfo identifies the client transaction for this exchange of documents, if there is a transaction
associated with it.  This element enables the coordinated tracking of transactions between business
partners.-->
<!ELEMENT TransactInfo  (transactId? , timeStamp? , Status? )>

<!--This optional attribute signals whether the envelope contains actual data, a request, a response or a
combination of these. It helps provide the envelope parser with a heads up on what to expect in the
payloads.-->
<!ATTLIST TransactInfo transactType  (data | request | response | multi )  "data">

<!--The PacketInfo element describes the contents of a packet.-->
<!ELEMENT PacketInfo  (packetId , action? , manifest , Status? )>

<!--This optional attribute describes the content of the packet. A packetType of data, the default, describes
the packet as containing an XML document. A value of response signifies that a Status element should also
be present and that the payload can and probably will be empty. A value of request should coexist only with
an action attribute that requests a service.-->
<!ATTLIST PacketInfo packetType  (data | request | response )  "data">

<!--The Status element contains response information about a transaction or payload.  A transaction or
packet whose type is response may have a Status element in it, which describes the processing status of
the referenced transaction or packet ID.-->
<!ELEMENT Status  (code , shortDescription? , longDescription? )>

<!--The optional action element signals the action that should be performed on the payload.  The various
HR-XML content specifications define the actions needed for their document exchanges.-->
<!ELEMENT action  (#PCDATA )>

<!--This element contains a numeric code indicating success or some variation on failure.  See the HR-XML
Staffing Exchange Protocol Specification for an initial list of status codes.-->
<!ELEMENT code  (#PCDATA )>

<!--A secret shared between sender and receiver for authentication purposes-->
<!ELEMENT credential  (#PCDATA )>

<!--The identifier of a user-->
<!ELEMENT id  (#PCDATA )>

<!--This element contains a longer and more complete description of the error.  If the short description is not
sufficient to pinpoint the cause, this description should be.-->
<!ELEMENT longDescription  (#PCDATA )>

<!--The contents of the payload are identified by the mandatory manifest element. This allows the envelope
```

```
parser to route the payload accordingly without parsing it. One approach is to use the URL of the payload
document's DTD.-->
<!ELEMENT manifest  (#PCDATA )>

<!--This element is mandatory inside the PacketInfo element and uniquely identifies the contents of the
payload. packetId should be unique within each envelope, and the combination of packetId and transactId
constitute a globally unique identifier for each packet.-->
<!ELEMENT packetId  (#PCDATA )>

<!--This element contains a brief description of the error.-->
<!ELEMENT shortDescription  (#PCDATA )>

<!--This element should be in ISO standard format.-->
<!ELEMENT timeStamp  (#PCDATA )>

<!--An identifier for the transaction can be included in the optional transactId. This identifies the envelope
transaction and not any of the constituent data, since there could be multiple documents within one
envelope.  Note that because each envelope has a unique transactId and each packet has a unique
packetId, the combination of transactId and packetId creates a globally unique identifier for each packet.-->
<!ELEMENT transactId  (#PCDATA )>

<!--The payload element contains one HR-XML document inside a CDATA section. This approach allows
the payload to be parsed separately from the envelope and avoids any interdependency between envelope
and contents when validating envelope documents.-->
<!ELEMENT payload  (#PCDATA )>

<!ENTITY % version "">
```