AdsML® Framework for E-Commerce Business Standards for Advertising

AdsMLEnvelope 1.1.4
Part 2
Specification & Schema

Document Authors: AdsML Technical Working Group

Document ID: AdsMLEnvelope-1.1.4-SpecP2Schema-AS-5

Document File Name: AdsMLEnvelope-1.1-SpecP2Schema-AS.pdf

Document Status: Approved Specification

Document Date: 30 June 2009

Draft Number: 5

Table of Contents

1 AdsML Standard Documentation

1.1 Document status and copyright

This is the Approved Specification of the AdsML® Envelope 1.1. It is a draft document and may be updated, replaced, or made obsolete by other documents at any time. It is inappropriate to use AdsML Proposed Specifications as reference material or to cite them as other than ″work in progress″.

1.2 Non-Exclusive License Agreement for AdsML Consortium Specifications

USER LICENSE

IMPORTANT: AdsML Consortium specifications and related documents, whether the document be in a paper or electronic format, are made available to you subject to the terms stated below. Please read the following carefully.

1. All AdsML Consortium Copyrightable Works are licensed for use only on the condition that the users agree to this license, and this work has been provided according to such an agreement. Subject to these and other licensing requirements contained herein, you may, on a non-exclusive basis, use the Specification.

2. The AdsML Consortium openly provides this specification for voluntary use by individuals, partnerships, companies, corporations, organizations and any other entity for use at the entity's own risk. This disclaimer, license and release is intended to apply to the AdsML Consortium, its officers, directors, agents, representatives, members, contributors, affiliates, contractors, or coventurers (collectively the AdsML Consortium) acting jointly or severally.

3. This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided

Specification or any portion thereof. Finally, you hereby agree that the AdsML Consortium is not liable for any direct, indirect, special or consequential damages arising from or relating to your acquisition, use, duplication, distribution, or exploitation of the Specification or any portion thereof.

8. This User License is perpetual subject to your conformance to the terms of this User License. The AdsML Consortium may terminate this User License immediately upon your breach of this agreement and, upon such termination you will cease all use duplication, distribution, and/or exploitation in any manner of the Specification.

9. This User License reflects the entire agreement of the parties regarding the subject matter hereof and supercedes all prior agreements or representations regarding such matters, whether written or oral. To the extent any portion or provision of this User License is found to be illegal or unenforceable, then the remaining provisions of this User License will remain in full force and effect and the illegal or unenforceable provision will be construed to give it such effect as it may properly have that is consistent with the intentions of the parties. This User License may only be modified in writing signed by an authorized representative of the AdsML Consortium. This User License will be governed by the law of Darmstadt (Federal Republic of Germany), as such law is applied to contracts made and fully performed in Darmstadt (Federal Republic of Germany). Any disputes arising from or relating to this User License will be resolved in the courts of Darmstadt (Federal Republic of Germany). You consent to the jurisdiction of such courts over you and covenant not to assert before such courts any objection to proceeding in such forums.

10. Except as expressly provided herein, you may not use the name of the AdsML Consortium, or any of its marks, for any purpose without the prior consent of an authorized representative of the owner of such name or mark.

IF YOU DO NOT AGREE TO THESE TERMS PLEASE CEASE ALL USE OF THIS SPECIFICATION NOW. IF YOU HAVE ANY QUESTIONS ABOUT THESE TERMS, PLEASE CONTACT THE SECRETARIAT OF THE ADSML CONSORTIUM.

AS OF THE DATE OF THIS REVISION OF THE SPECIFICATION YOU MAY CONTACT THE AdsML Consortium at www.adsml.org.

1.3 AdsML Code of Conduct

The AdsML Code of Conduct governs AdsML Consortium activities. A reading or reference to the AdsML Code of Conduct begins every AdsML activity, whether a meeting of the AdsML Consortium, AdsML Working Groups, or AdsML conference calls to resolve a technical issue. The AdsML Code of Conduct says:

Trade associations are perfectly lawful organizations. However, since a trade association is, by definition, an organization of competitors, AdsML Consortium members must take precautions to ensure that we do not engage in activities which can be interpreted as violating anti-trust or other unfair competition laws.

For any activity which is deemed to unreasonably restrain trade, AdsML, its members and individual representatives may be subject to severe legal penalties, regardless of our otherwise beneficial objectives. It is important to realize, therefore, that an action that may seem to make "good business sense" can injure competition and therefore be prohibited under the antitrust or unfair competition laws.

To ensure that we conduct all meetings and gatherings in strict compliance with any such laws and agreements in any part of the world, the AdsML Code of Conduct is to be distributed and/or read aloud at all such gatherings.

- There shall be no discussion of rates, fares, surcharges, conditions, terms or prices of services, allocating or sharing of customers, or refusing to deal with a particular supplier or class of suppliers. Neither serious nor flippant remarks about such subjects will be permitted.
- AdsML shall not issue recommendations about any of the above subjects or distribute to its members any publication concerning such matters. No discussions that directly or indirectly fix purchase or selling prices may take place.
- There shall be no discussions of members' marketing, pricing or service plans.
- All AdsML related meetings shall be conducted in accordance with a previously prepared and distributed agenda.
- If you are uncomfortable about the direction that you believe a discussion is heading, you should say so promptly.

Members may have varying views about issues that AdsML deals with. They are encouraged to express themselves in AdsML activities. However, official AdsML communications to the public are the sole responsibility of the AdsML Consortium. To avoid creating confusion among the public, therefore, the Steering Committee must approve press releases and any other forms of official AdsML communications to the public before they are released.

1.4 Document Number and Location

This document, Document Number AdsMLEnvelope-1.1.2-SpecP2Schema-AS-4, is freely available. It is located at the AdsML® website at http://www.adsml-framework.org/.

1.5 Purpose of this document

This document specifies an XML Schema definition of the Advertising Markup Language (AdsML) Envelope Version 1.1. AdsML is an XML-based language used for encoding and routing messages in the AdsML framework. The document exemplifies how an AdsMLEnvelope is created and provides a textual definition of AdsMLEnvelope Version 1.1. The AdsMLEnvelope Specification and the AdsMLEnvelope Schema together provide a normative definition of the AdsMLEnvelope.

1.6 Audience

The intended audience for this document is primarily user and vendor organizations who seek to implement the Advertising Markup Language Version 1.1 Envelope in their workflows, advertising systems, or software products. Those assessing the conformance of vendor products to the AdsMLEnvelope standard may also use the document.

Comments on this specification should be addressed to the AdsML Consortium and to the Technical Working Group of the AdsML Consortium (technical.wg@adsml.org).

1.7 Accompanying documents

This document is the reference guide to the AdsMLEnvelope schema. A companion document, *AdsMLEnvelope – Part 1 - Processing Model, Usage Rules & Guidelines*, provides rules and guidelines for using AdsMLEnvelope messages to address specific business requirements. They are meant to be read together.

Both documents are part of the AdsML Framework, which contains a suite of related documents. Readers of this document are assumed to be familiar with the full range of relevant AdsML documentation. In particular, readers are assumed to have read the *E-Commerce Usage Rules and Guidelines* document. A description of the entire document set can be found in the *ReadMeFirst* html file associated with this release of the Framework.

1.8 Definitions & conventions

1.8.1 Definitions of key words used in the specification

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are used as described in IETF RFC 2119.(S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. Internet Engineering Task Force (IETF), Request for Comments: 2119, March 1997, http://www.ietf.org/rfc/rfc2119.txt)

When any of these words do not appear in upper case as above, then they are being used with their usual English language sense and meaning.

1.8.2 Naming conventions – element, attribute, type, and file names

All element, attribute, and type names follow the '`CamelCase`' convention.

Element and type names begin use upper camel case and begin with capitals (`UpperCamelCase`). For example, `AdsMLEnvelope`, `MessageRef`, and `AdsMLStatusType`.

Attribute names begin using lower camel case and begin with lower case (*`lowerCamelCase`*). For example, *`language`* or *`messageId`*.

File names also follow the camel case convention and use upper camel case for each segment of the file name, plus dashes to separate the segments of the file name. Only the first two digits of the version number are included in the file name. The third digit of the version number (if there is one) and the Draft Number are only shown internally within the document. The full naming conventions for AdsML schema and specification file names are described in the document *AdsML Document Names and Identifiers – Guidelines and Examples*, a copy of which is included in this release of the Framework.

Schema for user-defined extensions to AdsML should use AdsML naming conventions to the extent possible. For example, '`ExampleInstanceFile.xml`', '`ExampleSchemaFile-1.0.xsd`', '`ExampleSchemaFile-1.1.xsd`'.

1.8.3 Typographical conventions

Element and type names are given in Courier New font, size 10. For example, `ResponseType`.

Attribute names are given in italicized Courier New font, size 10. For example, *`transmissionDateTime`*.

When citing examples of values that could be assigned to elements or attributes, the value is given in Courier New font, size 9, so "…the attribute taking the value of '`12`'."

1.8.4 Presentation of element, attribute, and type definitions and worked examples

The elements, attributes, and types of the AdsMLEnvelope standard are presented and explained by a written description, an extract of the appropriate schema fragment from the AdsMLEnvelope Schema, and a worked example.

Individual element and attribute definitions are presented in tabular form. Each is defined by name, occurrence, data type, and description. The occurrence of an element or attribute – whether it is required ('mandatory') or optional and how many times it appears - is indicated by,

- '1-1' – mandatory, **MUST** occur once
- '?' – Optional, **MAY** occur once
- '+' – Mandatory and repeatable, **MUST** occur once and **MAY** appear multiple times
- '*' – Optional and repeatable, **MAY** occur one or multiple times
- 'Choice' – identifies an element as belonging to a mandatory choice where the choice **MUST** occur once
- '+ Choice' – identifies an element as belonging to a mandatory and repeatable choice where the choice **MUST** occur once and **MAY** appear multiple times

To illustrate, an example of the definition of an optional *`id`* attribute of data type 'string' with accompanying description is given below.

Name	**Occurs**	**Type**	**Description**
`id`	?	`xs:string`	Records an optional identifier.

A schema fragment showing the normative schema definition of the elements, attributes, or types described in the table follows the tabular definition.

A worked example showing how the element, attribute, or type(s) appear in an instance document follows the schema fragment.

1.9 Change History

Number	Date	Changes	Author
1.1.4-AS-5	30 June 2009	• Updated to import AdsML Type Library 2.0 PS • Editorial – corrected erratum in version metadata	JC

1.1.3-AS-4	10 October 2007	• Updated to reference AdsML Type Library 2.0 and AdsML Controlled Vocabularies 3.0 • Editorial restructuring of sections 2 and 3	JC
1.1.2-AS-3	1 Oct 2006	Changed to Controlled Vocabulary 3.0. No other changes.	UW
1.1.1-AS-2	1 Oct 2006	AdsML references updated to reflect Registered Trademark status	TS
1.1.0-AS-1	1 June 2006	Refactored schema to use common components from the AdsML Type Library schema. Changes affect new namespace and a few other minor schema structures leading to that 1.1 document instances are not backwards compatible with 1.0 document instances. However, no changes in functionality and processing model. Renamed "AdsML" to "AdsMLEnvelope", as "AdsML" now covers the complete framework of AdsML standards. Overviews of the AdsML approach as well as other general documentation have been moved to other non-standard specific documents.	JC, UW
1.0.0-AS-1	17 May 2004	Approved Specification. Earlier change history removed.	JC, MD, UW

1.10 Acknowledgements

This document is a product of the AdsML Technical Working Group. Primary authorship and editing was performed by,

- Jay Cousins (RivCom Ltd.) - jay.cousins@rivcom.com
- Ulf Wingstedt (CNet Svenska AB) ulf.wingstedt@cnet.se

Portions are based on material written by:

- Marcel Dumont (Rosetta) - marcel@rosetta.nl

Acknowledgements and thanks to other contributors for additional input to this document are listed in Appendix A: Acknowledgement for contributions to this document.

1.11 Code of Conduct

The AdsML Code of Conduct governs AdsML® Consortium activities. A reading or reference to the AdsML Code of Conduct begins every AdsML activity, whether a meeting of the AdsML Consortium, AdsML Working Groups, or AdsML conference calls to resolve a technical issue. The AdsML Code of Conduct says:

Trade associations are perfectly lawful organizations. However, since a trade association is, by definition, an organization of competitors, AdsML Consortium members must take precautions to ensure that we do not engage in activities which can be interpreted as violating anti-trust or other unfair competition laws.

For any activity which is deemed to unreasonably restrain trade, AdsML, its members and individual representatives may be subject to severe legal penalties, regardless of our otherwise beneficial objectives. It is important to realize, therefore, that an action that may seem to make "good business sense" can injure competition and therefore be prohibited under the antitrust or unfair competition laws.

To ensure that we conduct all meetings and gatherings in strict compliance with any such laws and agreements in any part of the world, the AdsML Code of Conduct is to be distributed and/or read aloud at all such gatherings.

- There shall be no discussion of rates, fares, surcharges, conditions, terms or prices of services, allocating or sharing of customers, or refusing to deal with a particular supplier or class of suppliers. Neither serious nor flippant remarks about such subjects will be permitted.
- AdsML shall not issue recommendations about any of the above subjects or distribute to its members any publication concerning such matters. No discussions that directly or indirectly fix purchase or selling prices may take place.
- There shall be no discussions of members' marketing, pricing or service plans.
- All AdsML related meetings shall be conducted in accordance with a previously prepared and distributed agenda.
- If you are uncomfortable about the direction that you believe a discussion is heading, you should say so promptly.

Members may have varying views about issues that AdsML deals with. They are encouraged to express themselves in AdsML activities. However, official AdsML communications to the public are the sole responsibility of the AdsML Consortium. To avoid creating confusion among the public, therefore, the Steering Committee must approve press releases and any other forms of official AdsML communications to the public before they are released."

2 AdsMLEnvelope XML Schema – Overview

This section describes the use of XML Schema in the definition of the AdsML Envelope.

2.1 Schema Architecture

AdsMLEnvelope uses a modular schema architecture as defined by the AdsML Framework architecture consisting of the following schemas,

- The **Main Schema** – This schema defines the root element AdsMLEnvelope and all other components used in the standard, either by local definitions or by importing and/or including other schema files.
- The **AdsML Type Library** – This schema defines reusable components from the AdsML Framework.
- The **AdsML Controlled Vocabularies** – This schema defines all controlled vocabularies recommended by the AdsML Consortium.

All structures specific to AdsMLEnvelope are defined in the Main Schema. These structures are all defined in the AdsMLEnvelope namespace.

Where possible, AdsMLEnvelope specific structures have been defined as derivations of general AdsML Framework components defined in the AdsML Type Library that is imported into the Main Schema.

The AdsML Controlled Vocabularies schema provides a set of controlled vocabularies (CVs) that may be used in AdsML messages. The CVs are made available to all document instances through import into the Main Schema.

2.2 Schema Files

The complete set of schema files used in AdsMLEnvelope version 1.1, Approved Specification is:

```
AdsMLEnvelope-1.1-Main-AS.xsd
AdsMLTypeLibrary-2.0-PS.xsd
AdsMLControlledVocabularies-3.0-AS.xsd
```

2.3 AdsMLEnvelope namespaces

The AdsMLEnvelope defines an AdsML namespace,

'`http://www.adsml.org/adsmlenvelope/1.1`'

This is defined as the default namespace of the AdsMLEnvelope Schema. The AdsMLEnvelope Schema specifies this using `targetNamespace` and `xmlns` attributes as illustrated below,

```
<xs:schema targetNamespace="http://www.adsml.org/adsmlenvelope/1.1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.adsml.org/adsmlenvelope/1.1" ... >
```

Components reused from other standards carry their own namespaces that also have to be declared. The following external namespace definitions are also used:

```
adsml="http://www.adsml.org/typelibrary/2.0"

adsml-cv="http://www.adsml.org/controlledvocabularies/3.0"
```

It is **RECOMMENDED** to use namespace prefixes as listed above.
It is **RECOMMENDED** to have the AdsMLEnvelope namespace as the default namespace in AdsMLEnvelope document instances. If, however, a namespace prefix is wanted, then it is **RECOMMENDED** to use "`adsml-en`".

This prefix would be declared in the AdsMLEnvelope instance using an *`xmlns`* attribute as illustrated below,

```
<adsml-en:AdsMLEnvelope
        xmlns:adsml-en="http://www.adsml.org/adsmlenvelope/1.1"
        language="en-us">
...
</adsml-en:AdsMLEnvelope>
```

2.4 Validation and Schema Location

A trading partner **MUST NOT** send any invalid AdsMLEnvelope messages. However, use of XML Schema based validation of production messages in runtime is **OPTIONAL**. Systems are allowed to use any available approach to ensure that their output is valid.
For production messages, a schema location **SHOULD NOT** be given in document instances using the *`xsi:schemaLocation`* attribute. Systems are **REQUIRED** to be able to identify which schema a particular document instance belongs to by reading the mandatory *`adsml:schemaVersion`* attribute.

2.5 Empty values for elements and attributes

For the rules concerning the use of 'null' values in elements and attributes see the section 'Mandatory vs. required, blanks vs. nulls' in the '*AdsML E-commerce Usage Rules & Guidelines*' document.

2.6 Fixed and Default values

All fixed or default values specified for elements or attributes in the schema **MUST** be present in an XML document instance conforming to that schema; schema validation and the post-schema-validation infoset (PSVI) **SHOULD NOT** be relied upon in order to make fixed or default values available for processing.
This restriction is imposed so that a particular mode of validation (XML Schema validation and the PSVI) is not relied upon to ensure that all data content of a message is present in an instance messages. This allows for non-XML Schema validation of an instance.
This constraint is enforced in the schema by specifying attributes that carry fixed values with a 'use' of required, by not specifying default values, and by the policy that element content should not be empty in instances.

3 Content Model Reference

This is a reference section describing elements, attributes and other building blocks of the AdsMLEnvelope XML vocabulary's content model. The `AdsMLEnvelope` element is the root element, i.e. the top node of an AdsMLEnvelope message. Each building block is briefly described with the intention of providing context and background as well as some technical detail about its usage. Particular focus is placed on issues and business rules that are not possible to express using XML Schema. Note that the XML Schema specification includes additional rules. Components from imported external schemas are not described here; please see their specific specification documents. Such components are named with their recommended namespace prefix when discussed in the context of AdsMLEnvelope elements.

Elements and attributes with namespace prefix:	**Are described in the document:**
`adsml:`	*AdsMLTypeLibrary Schema & Specification*

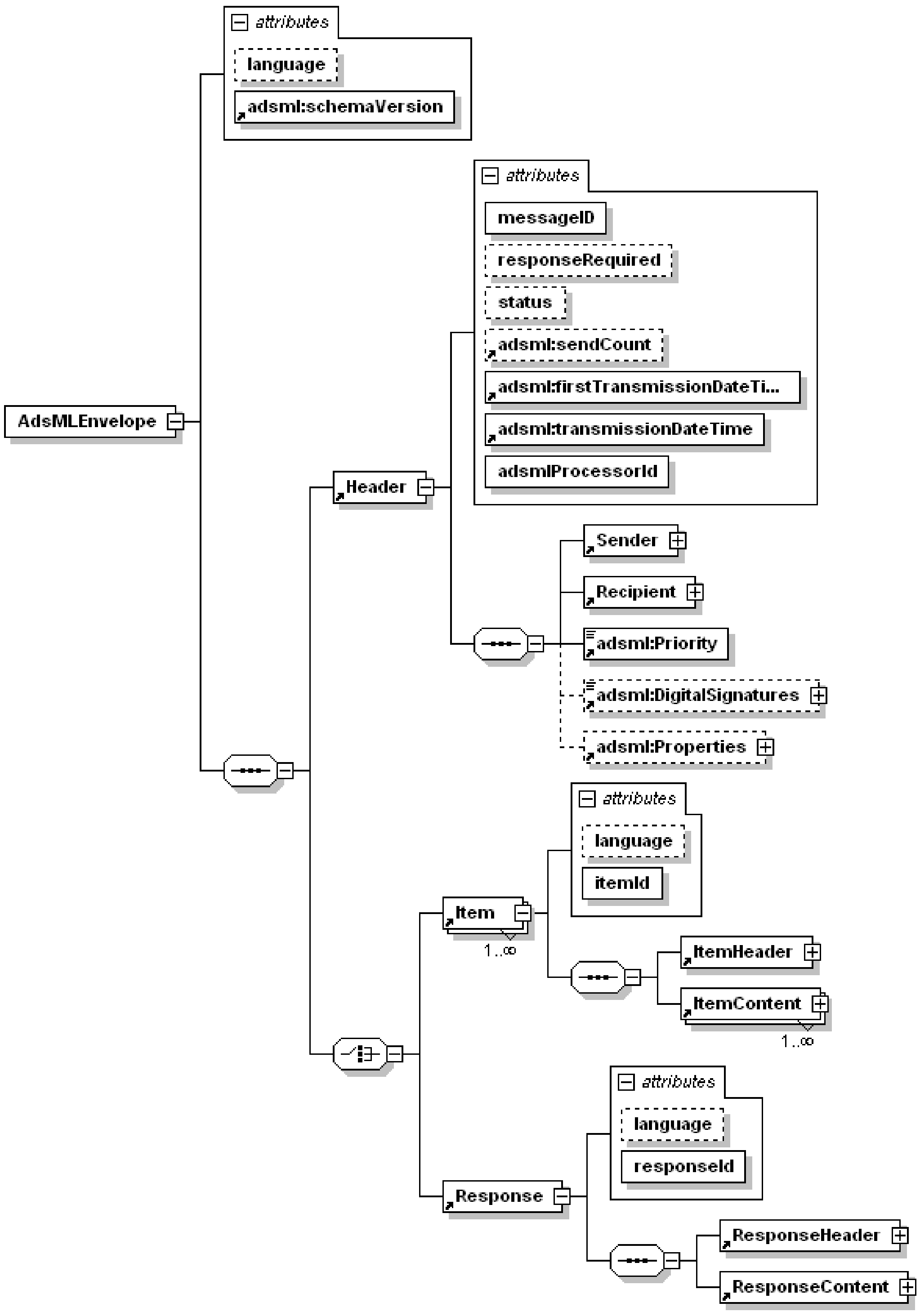
3.1Envelope Content Overview

The AdsMLEnvelope vocabulary defines an AdsML message that is an 'envelope' containing a 'header' and a 'content' layer. The header, defined by the `Header` element, holds administrative information sufficient to identify and route an AdsMLEnvelope message. The content layer contains advertising data, held by the `Item` element, and response data, held by the `Response` element. `Item`s contain operational data; `Response`s contain messaging data relating to the AdsMLEnvelope workflow.

The following sub-sections define the elements, attributes, and types used by the AdsMLEnvelope schema. The AdsMLEnvelope schema reuses structures from the AdsML Type Library, these structures identified by their 'adsml:' prefix. See the '*AdsML Type Library 2.0'* Specification for more information about the AdsML Type Library 'adsml:' namespace structures.

NOTE: The definitions in the formal XML Schema files listed above take precedence over the definitions in this document. Any inconsistency would normally be due to error, typos etc, in this document rather than in the XML Schema files.

A graphical illustration of the AdsML Envelope element and attribute tree is given below:



Additional rules and guidelines pertaining to the AdsML architecture and technical approach can be found in *E-Commerce Rules & Guidelines*.

3.2 Element: AdsMLEnvelope

The root element of an AdsMLEnvelope message is the `AdsMLEnvelope` element. The `AdsMLEnvelope`element contains an element sequence of mandatory `Header`, followed by a choice between mandatory and repeatable `Item` element(s) or a single `Response` element. The `AdsMLEnvelope` element has an optional *language* attribute and a required *adsml:schemaVersion* attribute.

See the '*AdsML Type Library 2.0*' Specification for more information about the AdsML Type Library 'adsml:' namespace structures.

Name	Occurs	Type	Description
Header	1-1		The Header element records message identification, message sender and recipient, test status, transmission time, response requirements, priority rating, and any user-defined information about an AdsMLEnvelope message. See Header for more information.
Item	+ Choice		The Item element contains the operational data of an AdsMLEnvelope message. See Item for more information. Depending on the choice taken by the user, an AdsMLEnvelope message **MUST** contain either one or more Item element(s) or a single Response element.
Response	1-1 Choice		The Response element contains any response content messaging data that may be carried by an AdsMLEnvelope message. See Response for more information. Depending on the choice taken by the user, an AdsMLEnvelope message **MUST** contain either a single Response element or one or more Item element(s).
language	?	adsml:LanguageType	The optional *language* attribute **MAY** be used to identify the human language that has been used in the elements and attributes of the AdsMLEnvelope message itself. The language is recorded as a string conformant to *IETF Request for Comment 3066*. The default value of *language* is set to 'en-us', American English.
adsml:schemaVersion	1-1	adsml:SchemaVersionType	The *adsml:schemaVersion* attribute records the version of the schema to which an instance document conforms.

The following XML Schema fragment defines the AdsMLEnvelope element:

```
<xs:element name="AdsMLEnvelope">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="Header"/>
    <xs:choice>
     <xs:element ref="Item" maxOccurs="unbounded"/>
     <xs:element ref="Response"/>
    </xs:choice>
   </xs:sequence>
  <xs:attribute name="language" type="adsml:LanguageType" use="optional"/>
  <xs:attribute ref="adsml:schemaVersion" use="required"/>
 </xs:complexType>
</xs:element>
```

Illustration – example of AdsMLEnvelope element in an instance

To illustrate, the example below shows an `AdsMLEnvelope` root element in an instance. The first *`xmlns`* attribute declares the AdsMLEnvelope namespace. The second *`xmlns`* attribute declares the namespace of the W3C XML Schema instance schema and assigns it the prefix '`xsi`'. The *`language`* attribute has its default value of '`en-us`', indicating that any text used in the instance's elements or attributes will be in American English. The *`adsml:schemaVersion`* attribute identifies the version of the AdsMLEnvelope schema to which the instance conforms. Although no children elements are shown in this example, the `AdsMLEnvelope` element will contain a `Header` element child followed by either `Item` element(s) or a single `Response` element.

```
<?xml version="1.0" encoding="UTF-8"?>
<AdsMLEnvelope xmlns="http://www.adsml.org/adsmlenvelope/1.1"
xmlns:adsml="http://www.adsml.org/typelibrary/2.0"
xmlns:adsml-cv="http://www.adsml.org/controlledvocabularies/3.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.adsml.org/adsmlenvelope/1.1/AdsMLEnvelope-
1.0-AS.xsd" language="en-us" adsml:schemaVersion="1.1.2">
 ...
</AdsMLEnvelope>
```

3.2.1 Element: Header

The `Header` element records identification, message sender and recipient, envelope status, transmission, AdsMLEnvelope schema version, response requirements, priority rating, and user-defined information about an AdsMLEnvelope message.

The `Header` element contains an element sequence of mandatory `Sender`, `Recipient` and `adsml:Priority` elements followed by an optional `adsml:DigitalSignatures` and optional `adsml:Properties` elements. The `Header` element has mandatory *`messageId`*, *`adsml:firstTransmissionDateTime`*, *`adsml:transmissionDateTime`*, and *`adsmlProcessorId`*, and optional *`responseRequired`*, *`status`*, and *`adsml:sendCount`* attributes.

See the '*AdsML Type Library 2.0*' Specification for more information about the AdsML Type Library 'adsml:' namespace structures.

Name	**Occurs**	**Type**	**Description**
`Sender`	1-1		Identifies the sender of the AdsMLEnvelope message.

			See Sender for more information.
Recipient	1-1		Identifies the intended recipient of the AdsMLEnvelope message. See Recipient for more information.
adsml:Priority	1-1	adsml:PriorityType	Records the priority rating that applies to an AdsMLEnvelope message, enabling the message to be prioritized appropriately as it is routed through sender and recipient systems. The priority of a message **MUST NOT** be lower than the priority rating of any Item(s) or Response(s) in the envelope.
adsml:DigitalSignatures	?	xs:any	**MAY** be used to record a signature for the AdsMLEnvelope message using a W3C XML Signature.
adsml:Properties	?		**MAY** be used to record any user-specific properties that the sender and recipient of an AdsMLEnvelope message **MAY** specify for use in the Header element as a name-value pair.
messageId	1-1	adsml:QIDType	Records a globally unique identifier for the AdsMLEnvelope message. The identifier value **MUST** enable a message to be unambiguously identified within the operational context in which it is being used.
responseRequired	?	adsml:BooleanType Default: false	Indicates if the sender of an AdsMLEnvelope message requires the recipient to send a response to the sender upon receipt of the message. If set to 'true', it indicates that the message **MUST** be responded to by sending an message containing an appropriate Response.
status	?	EnvelopeStatusCV Default: Production	Records the status of an AdsMLEnvelope message, indicating if the message is a standard production message containing genuine data or a test message sent for system testing purposes.
adsml:sendCount	?	adsml:PositiveIntegerType Default: 1	Records the number of times that an AdsMLEnvelope message has been sent. Used to identify if a message is being sent for the first time or if it is a repeat transmission. With each resending of a message, the *sendCount* attribute **MUST** be incremented by 1 so that the value of this attribute always reflects the number of times that the message

			has been sent.
adsml:firstTransmissionDateTime	1-1	adsml:DateTimeType	Records the date and time at which an AdsMLEnvelope message was first transmitted. The value of *firstTransmissionDateTime* **MUST** always be that of the message's first transmission and **MUST NOT** be changed or modified during any subsequent transmissions of that message.
adsml:transmissionDateTime	1-1	adsml:DateTimeType	Records the date and time at which an AdsMLEnvelope message was transmitted. The value of *transmissionDateTime* **MUST** always be that of the message's current transmission and **MUST** be updated to reflect the new transmission time in the event of a resend. When an message is being sent for the first time, then the values of the *firstTransmissionDateTime* and *transmissionDateTime* attributes **MUST** be identical.
adsmlProcessorId	1-1	adsml:LongStringType	Records the processor id of the AdsMLEnvelope processor that has sent (and so is the origin of) the message. The value of the *adsmlProcessorId* attribute is recorded as a string.

The following XML Schema fragment defines the `Header` element,

```
<xs:element name="Header">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="Sender"/>
   <xs:element ref="Recipient"/>
   <xs:element ref="adsml:Priority"/>
   <xs:element ref="adsml:DigitalSignatures" minOccurs="0"/>
   <xs:element ref="adsml:Properties" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="messageID" type="adsml:QIDType" use="required"/>
  <xs:attribute name="responseRequired" type="adsml:BooleanType"
use="optional"/>
  <xs:attribute name="status" type="EnvelopeStatusCV" use="optional"/>
  <xs:attribute ref="adsml:sendCount" use="optional" default="1"/>
  <xs:attribute ref="adsml:firstTransmissionDateTime" use="required"/>
  <xs:attribute ref="adsml:transmissionDateTime" use="required"/>
  <xs:attribute name="adsmlProcessorId" type="adsml:LongStringType"
use="required"/>
 </xs:complexType>
</xs:element>
```

3.2.1.1 Element: Sender

The `Sender` element records identification and contact information about the sender of an AdsMLEnvelope message. The `Sender` element contains an element sequence of an optional `adsml:Name` element, mandatory `BusinessEntity` element, and an optional and repeatable `Contact` element. The `Sender` element has an optional *`senderReferenceNo`* attribute.

See the '*AdsML Type Library 2.0*' Specification for more information about the AdsML Type Library 'adsml:' namespace structures.

Name	Occurs	Type	Description
`adsml:Name`	?	adsml:LongStringType	Records a name for the sender.
`BusinessEntity`	1-1		A sender **MUST** be identified by one or more business entity identifier(s) using the `BusinessEntity` element. See BusinessEntity element for definition.
`Contact`	*		Information about whom to contact about the AdsMLEnvelope message, e.g. in case of transmission/message errors, **MAY** be recorded using the `Contact` element. Any contact information so given provides contact information intended for a human being in the receiver's organization in the event that validation fails and human error handling is required. See Contact for more information.
`senderReferenceNo`	?	adsml:LongStringType	Normally, the *`messageId`* of an AdsMLEnvelope message is expected to be used as a reference number when contacting the sender of an message, but the sender may provide an additional reference number as a string using the optional *`senderReferenceNo`* attribute.

The following XML Schema fragment defines the `Sender` element,

```
<xs:element name="Sender">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="adsml:Name" minOccurs="0"/>
   <xs:element ref="BusinessEntity"/>
   <xs:element ref="Contact" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="senderReferenceNo" type="adsml:LongStringType"
use="optional"/>
 </xs:complexType>
</xs:element>
```

3.2.1.1.1 Element: Contact

The `Contact` element records optional name, subject, and address information to be used when contacting people or organizations. It should be noted that the content model is very simple and is not intended for automatic processing of addresses etc but mainly for simple recording of contact information intended for manual handling. `Contact` contains an element sequence of an optional `Role` and `Name` element, and an optional and repeatable `ContentInfo` element. An optional `priority` attribute can be used to identify the priority rating of a contact.

See the '*AdsML Type Library 2.0*' Specification for more information about the AdsML Type Library 'adsml:' namespace structures.

Name	Occurs	Type	Description
`Role`	?	`adsml:ContactRoleRootType`	Records information about the identity or role of the contact. The `Role` can have a controlled vocabulary specified..
`adsml:Name`	?	adsml:LongStringType	Records a name for the contact as a string.
`ContactInfo`	*	`ContactInfoRootType`	Records contact information as an unstructured string, each line optionally classified by the element's *`class`* attribute. Each individual set of contact information is recorded using a separate `ContactInfo` element. The *`class`* attribute is declared as `adsml:CodeRootType` and so a controlled vocabulary can be specified for use in this attribute context. The `ContactInfo` element is declared as `ContactInfoRootType`. See ContactInfoRootType for this type definition.
`Priority`	?	`adsml:PriorityType`	Assigns a priority rating to the `Contact` element. The priority rating is used to identify the sequence in which contacts should be contacted in the event that more than one `Contact` element is present.

The following XML Schema fragment defines the `Contact` element,

```
<xs:element name="Contact" type="ContactType"/>

<xs:complexType name="ContactType">
 <xs:sequence>
  <xs:element name="Role" type="adsml:ContactRoleRootType" minOccurs="0"/>
  <xs:element ref="adsml:Name" minOccurs="0"/>
  <xs:element name="ContactInfo" type="ContactInfoRootType" minOccurs="0"
maxOccurs="unbounded"/>
 </xs:sequence>
 <xs:attribute name="priority" type="adsml:PriorityType" use="optional"/>
</xs:complexType>
```

Instance Example – Contact element

To illustrate, the example below shows an instance where two `Contact` elements are used to record contact address information for the company placing an advertisement, in this case the fictional company 'Northern Car Ads'.

The first `Contact` element identifies a direct contact at Northern Car and is identified as the primary contact point by virtue of the priority rating of '`1`' given to the `priority` attribute. The direct contact is identified as having the name '`John Hepplesw aite`'. John's role is identified by the `Role` element. The user has decided to use a specific value and so has specified the `AdsMLContactRoleCV` controlled vocabulary for use in the `Role` element, as shown by the `xsi:type` attribute of the `Role` element. The `Role` element has been given the data value of '`BuyerNameOrDepartment`' is given, identifying the buyer of advertising. The contact address is given by the `ContactInfo` element with the `class` attribute of '`PostalAddress`'. Further `ContactInfo` elements of `class` '`CellPhone`' and '`DirectOfficeNumber`' provide direct telephone numbers for reaching John. Note how the user has not specified a controlled vocabulary for use in the `class` attribute context and is recording the contact information type using its default root type of `adsml:CodeRootType`, which is a plain string.

The second `Contact` element identifies a general contact number at Northern Car and is identified as the secondary contact point by virtue of the priority rating of '`2`' given to the `priority` attribute. The second `Contact` element is identified by name as being '`Northern Car Ads`' and contains a single `ContactInfo` element with the `class` of '`MainSwitchboard`' that provides the main telephone number for Northern Car Ads.

```
<Contact priority="1">
 <Role xsi:type="AdsMLContactRoleCV">BuyerNameOrDepartment</Role>
 <adsml:Name>John Hepplesw aite</adsml:Name>
 <ContactInfo class="PostalAddress">Northern Car Ads, Harrogate Avenue,
Yorkshire.</ContactInfo>
 <ContactInfo class="CellPhone">04790123001</ContactInfo>
 <ContactInfo class="DirectOfficeNumber">04560123012</ContactInfo>
</Contact>
<Contact priority="2">
 <adsml:Name>Northern Car Ads</adsml:Name>
 <ContactInfo class="MainSwitchboard>04560123001</ContactInfo>
</Contact>
```

3.2.1.2 Element: Recipient

The `Recipient` element records the name and at least one organization identifier for the intended recipient of an AdsMLEnvelope message. The `Recipient` element contains an element sequence of an optional `Name` element and a mandatory `BusinessEntity` element.

See the '*AdsML Type Library 2.0*' Specification for more information about the AdsML Type Library 'adsml:' namespace structures.

Name	**Occurs**	**Type**	**Description**
`adsml:Name`	?	adsml:LongStringType	Optionally used to record a name for the recipient. The recipient **SHOULD** always be identified by name using the `adsml:Name` element.
`BusinessEntity`	+		The intended recipient of the AdsMLEnvelope message **MUST** be identified by one or more business entity identifier(s) using the `BusinessEntity` element. On receiving the message, the recipient uses these identifier(s) to verify that the message has reached the correct destination. See BusinessEntity element for a definition.

The following XML Schema fragment defines the Recipient element:

```
<xs:element name="Recipient">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="adsml:Name" minOccurs="0"/>
   <xs:element ref="BusinessEntity" maxOccurs="unbounded"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

Instance example – example of Header element in an AdsMLEnvelope instance

To illustrate, the example below shows a `Header` element in an AdsMLEnvelope message instance. The message type, response requirements, identification and transmission information are given by the attributes of the `Header` element. The *`messageId`* attribute records the globally unique identifier for the message.

The message is a test message, shown by the value of '`EnvelopeTest`' given to the *`status`* attribute. As response to a test message is mandatory, the *`responseRequired`* attribute is set to '`true`'.

The identical values of the *`adsml:transmissionDateTime`* and *`adsml:firstTransmissionDateTime`* attributes show this to be the first transmission of the message, confirmed by the default value of '`1`' given to the *`adsml:sendCount`* attribute. The *`adsmlProcessorId`* attribute identifies the AdsMLEnvelope Processor that sent the message as the processor with the id of '`ADSML-PR-ID-6745`'.

The `Sender` element identifies the sender of the message as having the name '`Northern Car`' and uniquely identifies this business entity by using its VAT number of '`123 846 082 11`'; the sender has assigned their own reference number of '`ref001`' to the message using the *`senderReferenceNo`* attribute.

Two sets of contact information are provided by `Contact` elements. The primary contact, identified by the `Contact` element whose *`priority`* attribute has the value '`1`', is the Sales and Marketing Manager with the name George Carlsson, who can be reached at the postal address of Northern Car and by his mobile and direct office telephone numbers. A second contact address provides the main switchboard telephone number for Northern Car, information identified as the secondary contact point by the value of '`2`' given to the `Contact` element's *`priority`* attribute.

The `Recipient` element identifies the intended recipient of the message as '`National Automobile Advertising Ltd.`', which is also uniquely identified using its VAT number, '`903 776 081 61`'. The `adsml:Priority` element identifies the message as having the highest priority rating, shown by its value of '`1`'.

```
...
<Header messageID="adsml.org:2004-01-01:001" responseRequired="true"
status="EnvelopeTest" adsml:sendCount="1"
adsml:firstTransmissionDateTime="2003-08-29T09:30:47-05:00"
adsml:transmissionDateTime="2003-08-29T09:30:47-05:00"
adsmlProcessorId="ADSML-PR-ID-6745">
 <Sender senderReferenceNo="ref001">
  <adsml:Name>Northern Car</adsml:Name>
   <BusinessEntity>
    <BusinessEntityId>
     <BusinessEntityIdClass>VAT</BusinessEntityIdClass>
     <BusinessEntityIdValue>123 846 082 11</BusinessEntityIdValue>
     </BusinessEntityId>
    </BusinessEntity>
    <Contact priority="1">
     <Role>Sales and Marketing Manager</Role>
      <adsml:Name>George Carlsson</adsml:Name>
      <ContactInfo class="PostalAddress">Northern Car, Harrogate Avenue, 
Yorkshire.</ContactInfo>
      <ContactInfo class="CellPhone">04790123001</ContactInfo>
     <ContactInfo class="DirectOfficeNumber">04560123012</ContactInfo>
    </Contact>
    <Contact priority="2">
     <adsml:Name>Northern Car</adsml:Name>
     <ContactInfo class="MainSwitchboard">04560123001</ContactInfo>
    </Contact>
   </Sender>
   <Recipient>
   <adsml:Name>National Automobile Advertising Ltd.</adsml:Name>
   <BusinessEntity>
    <BusinessEntityId>
     <BusinessEntityIdClass>VAT</BusinessEntityIdClass>
     <BusinessEntityIdValue>903 776 081 61</BusinessEntityIdValue>
     </BusinessEntityId>
    </BusinessEntity>
  </Recipient>
 <adsml:Priority>5</adsml:Priority>
</Header>
...
```

3.3 Element: Item

The `Item` element contains the operational data – the actual advertising data - inside `Item` element(s). The `Item` element contains an element sequence of mandatory `ItemHeader`, followed by mandatory and repeatable `ItemContent` element(s). An `Item` **MUST** only contain more than one `ItemContent` if those `ItemContent`(s) are duplicate versions of one another – that is, they contain alternative representations of the same data. It is an implicit business rule and business logic that the `ItemHeader` data applies to all `ItemContent`. The `Item` element has an optional *`language`* attribute and a required *`itemId`* element.

See the '*AdsML Type Library 2.0*' Specification for more information about the AdsML Type Library 'adsml:' namespace structures.

Name	Occurs	Type	Description
`ItemHeader`	1-1		The `ItemHeader` child element contains information about the `Item` sufficient to route the `Item` and to perform initial processing of the `Item` and its `ItemContent` children upon receipt. See ItemHeader element for definition.
`ItemContent`	+		The `ItemContent` child element(s) contains the operational data of the `Item`. A single `Item` **MUST** only contain more than one `ItemContent` children elements if the `ItemContent` child elements contain duplicate versions of the same operational data. `ItemContent` elements are only considered to contain duplicate versions of

			the same operational data if the `ItemContent` elements contain alternate representations of that data and so share the same `ItemHeader` data. For example, two `ItemContent` elements carry the same advertisement but in different encoding formats, with one in IfraAdConnexion and one in AdsMLBookings, are considered to be duplicate versions of the same operational data. Where duplicate data is present, `ItemContent` elements are differentiated from one another by comparison of the `ContentHeader` data within each `ItemContent` element in order to identify the correct `ItemContent`. See ItemContent element for definition.
`language`	?	`adsml:LanguageType`	If present, identifies the human language used in the `Item` element subtree should this be different from the language globally specified for the whole message by the `AdsMLEnvelope` element's *`language`* attribute. The language is recorded as a string conformant to *IETF Request for Comment 3066*.
`itemId`	1-1	adsml:QIDType	Records a globally unique identifier for the `Item`.

The following XML Schema fragment defines the `Item` element:

```
<xs:element name="Item">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="ItemHeader"/>
   <xs:element ref="ItemContent" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="language" type="adsml:LanguageType" use="optional"/>
  <xs:attribute name="itemId" type="adsml:QIDType" use="required"/>
 </xs:complexType>
</xs:element>
```

Instance Example – Item element

To illustrate, the example below shows an `Item` element containing an `ItemHeader` element and a single `ItemContent` element. The *`itemId`* attribute records a unique identifier for the `Item` element.

```
<Item language="en-us" itemId="adsml.org:2004-01-01:001">
 <ItemHeader>
  ...
 </ItemHeader>
 <ItemContent>
 ...
 </ItemContent>
</Item>
```

3.3.1 Element: ItemHeader

The ItemHeader child element contains messaging data that is used to route an Item to its intended destination according to the priority assigned to that Item. The data in the ItemHeader **MUST** apply to all of the ItemContent element(s) contained by that Item. This is consistent with the constraint that an Item can only contain more than one ItemContent element if the ItemContent elements contained are duplicates of the same data, that data only differing in its representation.

The ItemHeader element contains an element sequence of a mandatory and repeatable ItemType element, a mandatory MessageClass element, a mandatory adsml:Priority element, a mandatory To element, a mandatory Destination element, an mandatory LastProcessedBy element, a mandatory ItemHistory element, an optional adsml:DigitalSignatures element, and an optional adsml:Properties element.

See the '*AdsML Type Library 2.0*' Specification for more information about the AdsML Type Library 'adsml:' namespace structures.

Name	Occurs	Type	Description
ItemType	+	adsml:BusinessMessageRootType	Records the type of Item that an Item has been classified as. An Item **MAY** be classified as more than one ItemType to allow for circumstances where the particular format in which operational data is being exchanged allows information that applies to one or more business objects to be held in a single document and so the content of an Item can be classified as being of more than one ItemType. The ItemType element records Item type using BusinessMessageRootType. As with all contexts where a root type is used, the BusinessMessageRootType can be substituted with a controlled vocabulary of specific values if required.
MessageClass	1-1	adsML:MessageClassCV	Records the message class of the message carried as the Item content of an Item, enabling it to be identified as carrying a standard business transaction, an administrative error, or an administrative acknowledgement message. See MessageClass element for definition.

adsml:Priority	1-1	adsml:PriorityType	Records the priority rating that applies to an Item, enabling the Item to be prioritized appropriately as it is routed through sender and recipient systems. Note that the priority rating applies to the routing of the Item. Scheduling information that applied to the Item 's content such as due dates and deadlines would be recorded within the operational data carried by the Item. If a user wishes to specify further scheduling information at the ItemHeader level, then they can do so by defining properties to record this using the adsml:Properties element.
To	1-1	BusinessEntityType	Records business entity identifier(s) for the organization to which an Item is being sent, and enables an AdsMLEnvelope processor to correctly route an Item during processing. The To destination may be the organization that is the final destination of an Item, or may be an intermediate routing point. If the To element identifies the final destination of the Item, then the value of To will be identical to that of the Destination element. If the To element identifies an intermediary destination point, then the value of To will record a unique identifier or identifier(s) for that intermediary point. See BusinessEntityType for type definition.
Destination	1-1	BusinessEntityType	Records organization identifier(s) for the organization that is the intended destination of an Item, identifying the organization that is the final consumer of the Item's content. The Destination element is declared as BusinessEntityType. See BusinessEntityType for type definition.
LastProcessedBy	1-1	BusinessEntityType	Records organization identifier(s) for the organization that last processed the operational data content of an Item in some way – for example, a pre-flighting service checking that item content conforms to technical specifications. The LastProcessedBy element is declared as BusinessEntityType. See BusinessEntityType for type definition.
ItemHistory	1-1		Records a history 'stack' of the activities that an Item has passed through during routing. See ItemHistory element for definition.

adsml:DigitalSignatures	?	xs:any	**MAY** be used to record a signature for an Item using a W3C XML Signature.
adsml:Properties	?		**MAY** be used to record any user-specific properties that the sender and recipient of an AdsMLEnvelope message **MAY** specify for use in the ItemHeader element as a name-value pair.

The following XML Schema fragment defines the ItemHeader element:

```
<xs:element name="ItemHeader">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="ItemType" maxOccurs="unbounded"/>
   <xs:element ref="MessageClass"/>
   <xs:element ref="adsml:Priority"/>
   <xs:element ref="To"/>
   <xs:element ref="Destination"/>
   <xs:element ref="LastProcessedBy"/>
   <xs:element ref="ItemHistory"/>
   <xs:element ref="adsml:DigitalSignatures" minOccurs="0"/>
   <xs:element ref="adsml:Properties" minOccurs="0"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

3.3.1.1 Element: ItemHistory

The ItemHistory element records a history 'stack' of the activities that an Item has passed through during routing. The ItemHistory element contains a mandatory and repeatable Activity element, with each activity recorded in a separate Activity element.

Name	Occurs	Type	Description
Activity	+		The first Activity element **MUST** always record the original activity that the Item underwent. For each additional activity that an Item has undergone, an additional Activity element **MUST** be added to the stack to record the activity.

The following XML Schema fragment defines the ItemHistory element,

```
<xs:element name="ItemHistory">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="Activity" maxOccurs="unbounded"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

The Activity element contains a mandatory Action element followed by a mandatory PerformedBy element. The *adsml:timeStamp* attribute of Activity records the time stamp of when the activity took place.

See the '*AdsML Type Library 2.0*' Specification for more information about the AdsML Type Library 'adsml:' namespace structures.

Name	Occurs	Type	Description
`Action`	1-1	[`ItemActionRootType`]	Identifies the type of action that an `Item` underwent during an activity. The `Action` element records item action using `ItemActionRootType` As with all contexts where a root type is used, the `ItemActionRootType` can be substituted with a controlled vocabulary of specific values if required. AdsML recommends the `AdsMLItemActionType` controlled vocabulary defined in the AdsML Type Library **SHOULD** be used to record `Item` action type. See ItemActionRootType for definition of `ItemActionRootType`.
`PerformedBy`	1-1	`BusinessEntityType`	Identifies the organization that performed the action during the activity. The `PerformedBy` element is declared as `BusinessEntityType`. See BusinessEntityType for type definition.
`adsml:timeStamp`	1-1	`adsml:DateTimeType`	Records the time at which the activity took place.

The following XML Schema fragment defines the `Activity` element:

```
<xs:element name="Activity">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="Action"/>
   <xs:element ref="PerformedBy"/>
  </xs:sequence>
  <xs:attribute ref="adsml:timeStamp" use="required"/>
 </xs:complexType>
</xs:element>

<xs:element name="Action" type="ItemActionRootType"/>

<xs:element name="PerformedBy" type="BusinessEntityType"/>
```

Instance Example – ItemHeader element

To illustrate, the example below shows an `ItemHeader` with `ItemType`, `To`, `Destination`, `LastProcessedBy`, and `ItemHistory` children elements. The optional `adsml:Properties` element is not present, so no user-defined properties have been specified.

The `ItemType` element is specified as taking a value defined in the `AdsMLBusinessMessageType` controlled vocabulary, the value of '`AM-PO`' identifying the `ItemContent` as containing an `Item` of type 'Production Order'. The `MessageClass` element takes its default value of '`BusinessTransaction`' as defined in the `AdsMLMessageClassType` controlled vocabulary, identifying the

`ItemContent` as containing an `Item` that is a standard business transaction. In this example, the transaction is the simple transmission of a production order message. The `Priority` element identifies the `Item` as having the highest priority rating, shown by its value of '`1`'. The `To` element identifies the destination to which the `Item` is being routed, and is identical to the value of the `Destination` element, indicating that there are no intermediate routing steps in the `Item`'s transmission. The `LastProcessedBy` element identifies the organization that last processed the `Item` to be the one with the identifier of '`123 846 082 11`'. The `ItemHistory` element identifies the actions that have been performed on the `Item`. In this instance, the history 'stack' contains a single `Activity` element. The time the activity occurred is identified by the *`timeStamp`* attribute of the `Activity` element. Within the `Activity` element, the `Action` element uses a value defined in the `AdsMLItemActionType` controlled vocabulary, the value of '`Create`' identifying the `Action` as having been the initial creation of the `Item`. The `PerformedBy` element identifies the business entity that performed the action as having the business entity identifier of '`123 846 082 11`', showing that the business entity that last 'touched' the `Item` is in this case the same business entity that created the `Item`.

```
...
<ItemHeader>
 <ItemType xsi:type="adsml:AdsMLBusinessMessageCV">AM-PO</ItemType>
  <MessageClass>BusinessTransaction</MessageClass>
  <adsml:Priority>1</adsml:Priority>
  <To>
   <BusinessEntityId>
    <BusinessEntityIdClass>VAT</BusinessEntityIdClass>
    <BusinessEntityIdValue>903 776 082 61</BusinessEntityIdValue>
    </BusinessEntityId>
   </To>
   <Destination>
    <BusinessEntityId>
     <BusinessEntityIdClass>VAT</BusinessEntityIdClass>
      <BusinessEntityIdValue>903 776 082 61</BusinessEntityIdValue>
     </BusinessEntityId>
    </Destination>
    <LastProcessedBy>
     <BusinessEntityId>
      <BusinessEntityIdClass>VAT</BusinessEntityIdClass>
      <BusinessEntityIdValue>123 846 082 11</BusinessEntityIdValue>
     </BusinessEntityId>
    </LastProcessedBy>
    <ItemHistory>
     <Activity adsml:timeStamp="2003-08-29T09:28:47-05:00">
     <Action xsi:type="ItemActionCV">Create</Action>
     <PerformedBy>
      <BusinessEntityId>
      <BusinessEntityIdClass>VAT</BusinessEntityIdClass>
     <BusinessEntityIdValue>123 846 082 11</BusinessEntityIdValue>
    </BusinessEntityId>
   </PerformedBy>
  </Activity>
 </ItemHistory>
</ItemHeader>
...
```

3.3.2 Element: ItemContent

The `ItemContent` element contains an element sequence of a mandatory `ContentHeader` element followed by a mandatory `ContentData` element. Identification is provided for the `ItemContent` element by a required *`itemContentId`* attribute.

See the '*AdsML Type Library 2.0*' Specification for more information about the AdsML Type Library 'adsml:' namespace structures.

Name	Occurs	Type	Description
`ContentHeader`	1-1		Records the format, encoding, encryption method, and user-defined information that applies to the data carried by the `ItemContent` element's ContentData child. See ContentHeader element for definition.
`ContentData`	1-1		Contains the operational data carried by the `ItemContent` element. See ContentData element for definition.
`itemContentId`	1-1	`adsml:QIDType`	Records a globally unique identifier for the `ItemContent` element.

The following XML Schema fragment defines the `ItemContent` element:

```
<xs:element name="ItemContent">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="ContentHeader"/>
   <xs:element ref="adsml:ContentData"/>
  </xs:sequence>
  <xs:attribute name="itemContentId" type="adsml:QIDType" use="required"/>
 </xs:complexType>
</xs:element>
```

3.3.2.1 Element: ContentHeader

The `ContentHeader` element records metadata about the data contained inside its sibling `ContentData` element. `ContentHeader` contains an element sequence of an optional `adsml:MIMEType` element, a mandatory `adsml:Format` element, and optional `adsml:FormatProfile` `adsml:ContentDataEncoding`, `adsml:EncryptionMethod`, `adsml:ContentSizeInBytes`, and `adsml:Properties` elements.

Note that if a user wants to record a file name for the data contained in the `ContentData` element (for instance, to provide a file name for data extracted from the `ContentData` element and stored in a system), then this should be done using the `Properties` element in the `ContentHeader`.

See the '*AdsML Type Library 2.0*' Specification for more information about the AdsML Type Library 'adsml:' namespace structures.

Name	Occurs	Type	Description
`adsml:MIMEType`	?	`adsml:ShortStringType`	Records the MIME type of the content file containing the operational data.
`adsml:Format`	1-1	`adsml:FormatRootType`	Records the format or XML vocabulary used to represent the operational data contained inside the `ContentData` element. If

			applicable, the version of the format can be recorded using an optional `version` attribute. The `Format` element records format using `FormatRootType`. As with all contexts where a root type is used, the `FormatRootType` can be substituted with a controlled vocabulary of specific values if required. AdsML recommends the `AdsMLFormatType` controlled vocabulary **SHOULD** be used to record format type.
`adsml:FormatProfile`	?	`adsml:ShortStringType`	The `adsml:FormatProfile` element is used to identify the specific profile or subset of the format identified by the `adsml:Format` element.
`adsml:ContentDataEncoding`	?	`adsml:EncodingRootType`	Records the encoding of operational data contained inside the `ContentData` element. The `adsml:ContentDataEncoding` element records encoding using `adsml:EncodingRootType`. As with all contexts where a root type is used, the `adsml:EncodingRootType` can be substituted with a controlled vocabulary of specific values if required. AdsML recommends the `AdsMLEncodingCV` controlled vocabulary **SHOULD** be used to record encoding type.
`adsml:EncryptionMethod`	?	`adsml:EncryptionMethodRootType`	Records the encryption method used to encrypt the operational data contained inside the `ContentData` element. The `EncryptionMethod` element records encryption method using `EncryptionMethodRootType`, which is an adsml:ShortTokenType data type. As with all contexts where a root type is used, the `EncryptionMethodRootType` can be substituted with a controlled vocabulary of specific values if required. AdsML recommends the `AdsMLEncryptionMethodCV` controlled vocabulary **SHOULD** be used to record encryption method type.
`adsml:ContentSizeInBytes`	?	`adsml:PositiveIntegerType`	Records the size of the content file in bytes.
`adsml:Properties`	?		**MAY** be used to record any user-specific properties that the sender and recipient of an AdsMLEnvelope message **MAY** specify for

			use in the ContentHeader element as a name-value pair.

The following XML Schema fragment defines the ContentHeader element:

```
<xs:element name="ContentHeader">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="adsml:MIMEType" minOccurs="0"/>
   <xs:element ref="adsml:Format"/>
   <xs:element ref="adsml:FormatProfile" minOccurs="0"/>
   <xs:element ref="adsml:ContentDataEncoding" minOccurs="0"/>
   <xs:element ref="adsml:EncryptionMethod" minOccurs="0"/>
   <xs:element ref="adsml:ContentSizeInBytes" minOccurs="0"/>
   <xs:element ref="adsml:Properties" minOccurs="0"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

3.3.2.2 Element: adsml:ContentData

The adsml:ContentData element contains the operational data of an AdsMLEnvelope message. Operational data of any format can be contained and AdsML specifies no restriction on the data, except that any data carried **MUST** conform to the constraints applicable to character data in XML documents as defined by the XML 1.0 specification[1].

The adsml:ContentData allows mixed content of any type as long as the content does not invalidate the well-formedness of the surrounding AdsMLEnvelope message.

In the event that an XML document is contained as content then the ContentData element allows elements from any namespace to appear without validation.

See the '*AdsML Type Library 2.0*' Specification for more information about the adsml:ContentData structure.

Instance Example – ItemContent element

To illustrate, the example below shows an ItemContent with ContentHeader and adsml:ContentData children elements. The *itemContentId* attributes records a unique identifier for the ItemContent element.

Within ContentHeader the required adsml:Format child element is present, but the other optional elements are absent. The adsml:Format element is specified as taking a value defined in the AdsMLFormatCV controlled vocabulary, the value of 'AdConnexion' identifying the ItemContent as containing content data represented in the IfraAdConnexion format; the *version* attribute of Format identifies the IfraAdConnexion version as '2.0'.

```
...
<ItemContent itemContentId="adsml.org:2004-01-01:001">
 <ContentHeader>
  <adsml:Format adsml:version="2.0" xsi:type="adsml-
cv:AdsMLFormatTypeCV">IfraAdConnexion</adsml:Format>
```

[1] For allowable text and character content in XML see the following sections of the XML specification: Section 2.2 Characters, Section 2.4 Character Data and Markup, and Section 2.7 CDATA sections.

```
  </ContentHeader>
  <adsml:ContentData>
   <!-- Content goes here -->
  </adsml:ContentData>
 </ItemContent>
...
```

3.4 Element: Response

Responses to AdsMLEnvelope messages are sent when errors occur, when the sender requests acknowledgment of received messages, and when running in test mode. This section describes the elements used for responses.

For further information about response choreography in the AdsMLEnvelope standard , see the section '*Response Choreography*' in the accompanying document '*Envelope Processing Model, Usage Rules & Guidelines*'. (See Section 1.7 Accompanying documents above.).

The `Response` element contains an element sequence of a mandatory `ResponseHeader` element followed by a mandatory `ResponseContent` element. The `Response` element has a mandatory *`responseId`* attribute and optional *`language`* attribute.

See the '*AdsML Type Library 2.0*' Specification for more information about the `adsml:ContentData` structure.

Name	**Occurs**	**Type**	**Description**
`ResponseHeader`	1-1		The `ResponseHeader` element contains information about the `Response` sufficient to route the response and perform initial processing upon receipt. See ResponseHeader element for definition.
`ResponseContent`	1-1		The `ResponseContent` element contains the response messaging data carried by the `Response`. See ResponseContent element for definition.
`language`	?	`adsml:LanguageType`	If present, identifies the human language used in the `Response` element subtree should this be different from the language globally specified for the message by the `AdsMLEnvelope` element's *`language`* attribute. The language is recorded as a string conformant to *IETF Request for Comment 3066*.
`responseId`	1-1	`adsml:QIDType`	Records a globally unique identifier for the `Response` element.

The following XML Schema fragment defines the `Response` element,

```
<xs:element name="Response">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="ResponseHeader"/>
   <xs:element ref="ResponseContent"/>
  </xs:sequence>
  <xs:attribute name="language" type="adsml:LanguageType" use="optional"/>
  <xs:attribute name="responseId" type="adsml:QIDType" use="required"/>
 </xs:complexType>
</xs:element>
```

3.4.1 Element: ResponseHeader

The `ResponseHeader` element contains a sequence of mandatory `ResponseType`, `MessageRef`, `Priority`, `Destination`, and `LastProcessedBy` elements followed by an optional `Properties` element.

See the '*AdsML Type Library 2.0*' Specification for more information about the `adsml:ContentData` structure.

Name	Occurs	Type	Description
ResponseType	1-1	AdsMLResponseType	Records the type (or 'classification') of a response. See ResponseType element for definition.
MessageRef	1-1		The `MessageRef` element identifies the AdsMLEnvelope message to which the response relates. See MesssageRef element for definition.
adsml:Priority	1-1	AdsMLPriorityType	Records the priority rating that applies to an `Response`, enabling the `Response` to be prioritized appropriately as it is routed through sender and recipient systems. The priority of a response **MUST** always be identical to the priority of the envelope to which the `Response` is responding.
Destination	1-1	BusinessEntityType	Records organization identifier(s) for the organization that is the intended destination of an `Response` and the intended recipient of the `Response`'s content. The `Destination` element is declared as `BusinessEntityType`.
LastProcessedBy	1-1	BusinessEntityType	Records organization identifier(s) for the organization that created (and so 'last processed') the content of an `Response`. The `LastProcessedBy` element is declared as `BusinessEntityType`.
adsml:Properties	?		**MAY** be used to record any user-specific properties that the sender and recipient of an AdsMLEnvelope message **MAY** specify for use in the `Response` element as a name-value

			pair.

The following XML Schema fragment defines the ResponseHeader element,

```
<xs:element name="ResponseHeader">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="ResponseType"/>
   <xs:element ref="MessageRef"/>
   <xs:element ref="adsml:Priority"/>
   <xs:element ref="Destination"/>
   <xs:element ref="LastProcessedBy"/>
   <xs:element ref="adsml:Properties" minOccurs="0"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

3.4.1.1 Element: ResponseType

The ResponseType element records the type (or 'classification') of a response and identifies whether the response is acknowledging receipt of a valid AdsMLEnvelope message ('MessageOK') or contains information about an error in an message ('MessageErrorDetected'). ResponseType takes values from the controlled vocabulary EnvelopeResponseCV. See EnvelopeResponseCV.

```
<xs:element name="ResponseType" type="EnvelopeResponseCV"/>
```

3.4.1.2 Element: MessageRef

The MessageRef element identifies the AdsMLEnvelope message to which a response relates. The message identification information is recorded using a required *messageId* attribute and optional *sendCountNo*, *status*, and *transmissionDateTime* attributes.

See the '*AdsML Type Library 2.0*' Specification for more information about the adsml:ContentData structure.

Name	Occurs	Type	Description
messageId	1-1	adsml:QIDType	Records the unique identifier of the message being referred to. Its value **MUST** be equal to the value of the *messageId* of the referenced AdsMLEnvelope message.
sendCountNo	?	adsml:PositiverIntegerType Default: 1	Records the send count number of the referenced AdsMLEnvelope message. In cases where the referenced message has a *sendCount* larger than its default value of '1', the optional *sendCountNo* attribute **SHOULD** be used to record that number.
status	?	EnvelopeStatusCV Default: Production	Records the status of the message that the response is about. It can be used to differentiate between responses to production or test messages. When used, its value **MUST** correspond to the value of the *status* attribute in the Header element of the referenced message. It is

			given the value 'Production' if the response is to a message of 'Production' status. It is given the value of 'ResponseToEnvelopeTest' if the response is to a message of 'EnvelopeTest' status. The *status* attribute records its value using the EnvelopeStatusCV controlled vocabulary and has a default value of 'Production'. See EnvelopeStatusCV.
adsml:transmissionDateTime	?	adsml:DateTimeType	Records the date and time at which the referenced message was sent. When used, its value **MUST** be equal to the value of the *adsml:transmissionDateTime* attribute in the Header element of the referenced message.

The following XML Schema fragment defines the MessageRef element:

```
<xs:element name="MessageRef">
 <xs:complexType>
  <xs:attribute name="messageID" type="adsml:QIDType" use="required"/>
  <xs:attribute name="sendCountNo" type="adsml:PositiveIntegerType"
use="optional" default="1"/>
  <xs:attribute name="status" type="EnvelopeStatusCV" use="optional"
default="Production"/>
  <xs:attribute ref="adsml:transmissionDateTime" use="optional"/>
</xs:complexType>
</xs:element>
```

Instance Example – Response element with no errors

To illustrate, the example below shows a Response element containing a ResponseHeader and ResponseContent elements.

The ResponseHeader element contains ResponseType, MessageRef, and adsml:Priority element children. The optional adsml:Properties element is not present, so no user-defined properties have been specified for use in the response.

The ResponseType element is given the value of 'MessageOK', confirming that the AdsMLEnvelope message to which the response refers was successfully received.

The empty MessageRef element records its values using attributes. The *messageID* attribute identifies the message to which the response refers by repeating the value of that message's Header element *messageID* attribute, in this case 'adsml.org:2006-01-10:001'. The *sendCountNo* attribute repeats the send count number of the referenced message, in this case '1'. The *status* attribute repeats the status of the referenced message, in this case 'EnvelopeTest'. (Consequently, the message sending this response would have a Header whose *status* attribute would take the value 'ResponseToEnvelopeTest'.). The *adsml:transmissionDateTime* attribute repeats the date and time at which the referenced message was sent, in this case '2006-01-10T17:06:47-05:00'.

The adsml:Priority element child repeats the priority of the referenced message was sent, in this case a priority rating of '1'. The Destination element identifies the business entity to which the Response is being sent as the organization with the identifier of 'ag61'. The LastProcessedBy element identifies the organization that created and so 'last processed' the `Response` to be the business entity identified by the identifier '`ag12`'.

The `ResponseContent` element is empty in this example as there are no errors to report and so no response content to send.

```
<Response responseId="adsml.org:2004-01-01:0011">
 <ResponseHeader>
  <ResponseType>MessageOK</ResponseType>
  <MessageRef messageID="adsml.org:2006-01-10:001" sendCountNo="1"
status="EnvelopeTest" adsml:transmissionDateTime="2006-01-10T17:06:47-
05:00"/>
  <adsml:Priority>1</adsml:Priority>
  <Destination>
   <BusinessEntityId>
    <BusinessEntityIdClass>AB</BusinessEntityIdClass>
    <BusinessEntityIdValue>ag61</BusinessEntityIdValue>
   </BusinessEntityId>
  </Destination>
  <LastProcessedBy>
   <BusinessEntityId>
    <BusinessEntityIdClass>AB</BusinessEntityIdClass>
    <BusinessEntityIdValue>ag12</BusinessEntityIdValue>
   </BusinessEntityId>
  </LastProcessedBy>
 </ResponseHeader>
 <ResponseContent/>
</Response>
```

3.4.2 Element: ResponseContent

The `ResponseContent` element contains the details of the response by a choice of either an optional `MessageError` element or optional `OriginalMessage` and `OriginalMessageHeader` elements, followed by an optional `Properties` element.

In cases where the response is just an acknowledgement of a successfully received message, the `ResponseContent` is empty. (The acknowledgement is provided in the `ResponseHeader` element by assigning the mandatory `ResponseType` element the value of '`Message OK`'.)

When the response is due to an error, `MessageError` children element(s) **SHOULD** be used to transmit details about the error(s) being reported.

If present, the `OriginalMessage` element **MAY** be used to contain the original AdsMLEnvelope message that the response content refers to. The `OriginalMessageHeader` element **MAY** be used to record any encoding applied to the enclosed original message. When AdsMLEnvelope is being used in the "Send and Forget" mode then the `OriginalMessage` element **MUST** be present so that the recipient is able to identify to which message the response applies. For further information about response choreography, see the section '*Response Choreography*' in the accompanying document '*Envelope Processing Model, Usage Rules & Guidelines*'. (See Section 1.7 Accompanying documents above.).

The optional `adsml:Properties` element **MAY** be used for additional data in the response, both for error and acknowledgement response types.

Name	Occurs	Type	Description
`MessageE`	?		A container for a message error report and the original message to

rror	Choice		which that error report refers. See MessageError element definition.
OriginalMessageHeader	? Choice		A container for recording the encoding of the original message content file. See OriginalMessageHeader element definition.
OriginalMessage	? Choice		A container for the original AdsMLEnvelope message that the response refers to. See OriginalMessage element definition.
adsml:Properties	?		**MAY** be used to record additional data that the sender and recipient of an AdsMLEnvelope message **MAY** specify for use in the `ResponseContent` element.

The following XML Schema fragment defines the `ResponseContent` element,

```
<xs:element name="ResponseContent">
 <xs:complexType>
  <xs:sequence>
   <xs:choice>
    <xs:element ref="MessageError" minOccurs="0"/>
    <xs:sequence>
     <xs:element ref="OriginalMessageHeader" minOccurs="0"/>
     <xs:element ref="OriginalMessage" minOccurs="0"/>
    </xs:sequence>
   </xs:choice>
   <xs:element ref="adsml:Properties" minOccurs="0"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

3.4.2.1 Element: MessageError

The `MessageError` element records the description and location of the error(s) in an AdsML message using a sequence of a mandatory and repeatable `ErrorDetails` element, mandatory `OriginalMessage` and `OriginalMessageHeader` elements, and an optional `SuccessfullyRoutedItems` element.

Each error being reported is recorded in a single `ErrorDetails` element. The `OriginalMessage` element holds the original AdsMLEnvelope message with the error(s) that are being reported. The `SuccessfullyRoutedItems` element **MUST** be present if any `Item` content in the message has been routed. If the `SuccessfullyRoutedItems` element is not present, then it **MUST** be assumed that the AdsMLEnvelope Processor has routed no `Item`s.

Name	Occurs	Type	Description
ErrorDetails	+		Records the details of a validation error. A separate `ErrorDetails` element is used to record each error being reported. See ErrorDetails element for definition.

OriginalMessageHeader	1-1		A container for recording the encoding of the original message content file. See OriginalMessageHeader element definition.
OriginalMessage	1-1		The OriginalMessage element contains the original AdsMLEnvelope message that the response content refers to. See OriginalMessage element definition.
SuccessfullyRoutedItems	?		Present if the error was not catastrophic and indicates which of the message's Item(s) were successfully processed. See SuccessfullyRoutedItems element for definition.

The following XML Schema fragment defines the MessageError element,

```
<xs:element name="MessageError">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="ErrorDetails" maxOccurs="unbounded"/>
   <xs:element ref="OriginalMessageHeader"/>
   <xs:element ref="OriginalMessage"/>
   <xs:element ref="SuccessfullyRoutedItems" minOccurs="0"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

3.4.2.2 Element: ErrorDetails

The ErrorDetails element records the description and location of the error using a sequence of an optional and repeatable ErrorDescription and a mandatory ErrorLocation element. The error type and the reporting application may be recorded using optional *type* and *detectedBy* attributes. Note that if the *type* attribute is given a value of 'CatastrophicEnvelopeError', then this **MUST** mean that no Items in the envelope were routed.

Name	Occurs	Type	Description
ErrorDescription	*	adsml:StringType	Records a textual description of the validation error. ErrorDescription can appear multiple times to allow error descriptions to be recorded in different human languages should this be required. See ErrorDescription element for definition.
ErrorLocation	1-1		Contains a set of pointers into the original AdsMLEnvelope message that can be used to identify the location of an error. See ErrorLocation element for definition.

type	?	EnvelopeErrorCV	Identifies the type of the error using a code defined by EnvelopeErrorCV. See EnvelopeErrorCV.
detectedBy	?	adsml:LongStringType	Records the name or signature (such as the software name and version, e.g. 'msxml4SP1') of the validating application.

The following XML Schema fragment defines the ErrorDetails element,

```
<xs:element name="ErrorDetails">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="ErrorDescription" minOccurs="0" maxOccurs="unbounded"/>
   <xs:element ref="ErrorLocation"/>
  </xs:sequence>
  <xs:attribute name="type" type="EnvelopeErrorCV" use="optional"/>
  <xs:attribute name="detectedBy" type="adsml:LongStringType"
use="optional"/>
 </xs:complexType>
</xs:element>
```

3.4.2.2.1 Element: ErrorDescription

The ErrorDescription element provides a textual description of the fault that is being reported as a string. ErrorDescription has optional *errorCode* and *language* attributes.

Name	Occurs	Type	Description
errorCode	?	adsml:LongStringType	**MAY** be used to provide an error code either in place of an error description, or as the code classifying the error description.
language	?	adsml:LanguageType	**MAY** be used to identify the human language that has been used to record the error description. The language is recorded as a string conformant to *IETF Request for Comment 3066*.

The following XML Schema fragment defines the ErrorDescription element:

```
<xs:element name="ErrorDescription">
 <xs:complexType>
  <xs:simpleContent>
   <xs:extension base="adsml:StringType">
    <xs:attribute name="errorCode" type="adsml:LongStringType"
use="optional"/>
    <xs:attribute name="language" type="adsml:LanguageType" use="optional"
default="en-us"/>
   </xs:extension>
  </xs:simpleContent>
 </xs:complexType>
</xs:element>
```

3.4.2.2.2 Element: ErrorLocation

The ErrorLocation element contains a set of optional attributes that can be used to give the location of the error within the AdsMLEnvelope message.

Name	Occurs	Type	Description

positionInFile	?	adsml:PositiveIntegerType	Records the character number counted from the beginning of the file.
lineInFile	?	adsml:PositiveIntegerType	Records the line number counted from the beginning of the file.
positionInLine	?	adsml:PositiveIntegerType	Records the character number in the line given by the *lineInFile* attribute.
documentNode	?	adsml:StringType	Records an XPath expression pointing to the specific node where the error is located. The XPath expression should begin with the root AdsMLEnvelope element, for example such as 'documentNode="AdsMLEnvelope/Item[2]"'
itemInEnvelope	?	adsml:QIDType	When the error is particular to a single Item the *itemInEnvelope* attribute is assigned the value of the *itemId* attribute of the Item in question.

The following XML Schema fragment defines the ErrorLocation element,

```
<xs:element name="ErrorLocation">
 <xs:complexType>
  <xs:attribute name="positionInFile" type="adsml:PositiveIntegerType"
use="optional"/>
  <xs:attribute name="lineInFile" type="adsml:PositiveIntegerType"
use="optional"/>
  <xs:attribute name="positionInLine" type="adsml:PositiveIntegerType"
use="optional"/>
  <xs:attribute name="documentNode" type="adsml:StringType" use="optional"/>
  <xs:attribute name="itemInEnvelope" type="adsml:QIDType" use="optional"/>
 </xs:complexType>
</xs:element>
```

3.4.2.2.3 Element: SuccessfullyRoutedItems

The SuccessfullyRoutedItems element is present when an error has not been catastrophic and so processing of the AdsMLEnvelope message could continue. SuccessfullyRoutedItems identifies the individual Items in the message that have been successfully routed using a sequence of mandatory and repeatable SuccessfullyRoutedItem element(s).

Name	Occurs	Type	Description
SuccessfullyRoutedItem	+	adsml:QIDType	Records the globally unique identifier of a successfully routed Item, repeating the value of that Item's *itemId* attribute.

The following XML Schema fragment defines the SuccessfullyRoutedItems element:

```
<xs:element name="SuccessfullyRoutedItems">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="SuccessfullyRoutedItem" maxOccurs="unbounded"/>
  </xs:sequence>
 </xs:complexType>
```

```
</xs:element>

<xs:element name="SuccessfullyRoutedItem" type="adsml:QIDType"/>
```

3.4.2.3 Element: OriginalMessageHeader

The `OriginalMessageHeader` element records the details of any encoding that may have been applied to the enclosed original AdsMLEnvelope message to which the `Response` refers, using an optional `adsml:ContentDataEncoding` followed by an optional `adsml:Properties` element.

Name	Occurs	Type	Description
adsml:ContentDataEncoding	?	adsml:EncodingRootType	Records the encoding of the enclosed original message. The `adsml:ContentDataEncoding` element records encoding using `adsml:EncodingRootType`. As with all contexts where a root type is used, the `adsml:EncodingRootType` can be substituted with a controlled vocabulary of specific values if required. AdsML recommends the `AdsMLEncodingCV` controlled vocabulary **SHOULD** be used to record encoding type.
adsml:Properties	?		**MAY** be used to record additional data that the sender and recipient of an AdsMLEnvelope message **MAY** specify for use in the `ResponseContent` element.

The following XML Schema fragment defines the `OriginalMessageHeader` element:

```
<xs:element name="OriginalMessageHeader">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="adsml:ContentDataEncoding" minOccurs="0"/>
   <xs:element ref="adsml:Properties" minOccurs="0"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>

<xs:element name="OriginalMessage" type="adsml:StringType"/>
```

3.4.2.4 Element: OriginalMessage

The `OriginalMessage` element contains the complete original AdsMLEnvelope message to which the `Response` refers. The original message **MAY** be encoded or contained inside a CDATA section. Note that if the original message itself contains any CDATA sections then the contained CDATA section **MUST** be removed, or have the characters closing that CDATA section commented out, or the original message be encoded to ensure that no CDATA closing markers are embedded inside the outermost containing CDATA section with the result that the outermost containing CDATA section ends prematurely and the contained message is not properly nested inside the outermost CDATA section. Note that if the original message is encoded then the `OriginalMessageHeader` element **MUST** be present and **MUST** identify the encoding that has been applied to the original message.

When AdsMLEnvelope is being used in the "Send and Forget" mode then the `OriginalMessage` element **MUST** be present. For further information about response choreography, see the section '*Response Choreography*' in the accompanying document '*Envelope Processing Model, Usage Rules & Guidelines*'. (See Section 1.7 Accompanying documents above.).

The following XML Schema fragment defines the `OriginalMessage` element,

```
<xs:element name="OriginalMessage" type="adsml:StringType"/>
```

Instance Example – Response element with errors

To illustrate, the example below shows a `Response` element containing an error report. The error is non-fatal – described as an unknown format type not agreed in the PPA. In this example, the AdsMLEnvelope Processor that received the message has reported an error that breaches the PPA terms governing the AdsMLEnvelope information exchange between the business entities involved.

The `ResponseHeader` element contains `ResponseType`, `MessageRef`, `adsml:Priority`, `Destination`, and `LastProcessedBy` element children. The optional `adsml:Properties` element is not present, so no user-defined properties have been specified for use in the response.

The `ResponseType` element is given the value of '`MessageErrorDetected`', confirming that the AdsMLEnvelope message to which the response refers was not successfully received. The empty `MessageRef` element identifies the faulty message using its attributes. The *`messageID`* attribute identifies the message to which the response refers as '`adsml.org:2004-01-01:{C65E819C-5585-11D7-ACA0-00B0D022396B}`'. The *`sendCountNo`* identifies the faulty message as being the first sending as *`sendCountNo`* has a value of '`1`'. The *`status`* attribute identifies the status of the referenced message as '`Production`' and so containing operational data. The *`adsml:transmissionDateTime`* attribute identifies the referenced message as having been sent as '`2003-08-29T09:30:47-05:00`'. The `adsml:Priority` element child repeats the priority of the referenced message was sent, in this case a priority rating of '`1`'.

The `Destination` element identifies the intended destination of the `Response` as the organization with the identifier '`ag18`', the identifier labeled as assigned by the buyer using the '`AB`' code from the `AdsMLOrganizationIDClassCV`. The `LastProcessedBy` element identifies the creator of the `Response` as the organization with the identifier '`ag19`', an identifier also qualified as assigned by '`AB`'.

The `ResponseContent` element contains a single `MessageError` in which are nested `ErrorDetails`, `OriginalMessageHeader`, `OriginalMessage`, and `SuccessfullyRoutedItems` element children. The optional `adsml:Properties` element is not present, so no user-defined properties have been specified for use in the response.

The `MessageError` element identifies the error as being of error type 'PPA' and having been detected by the application identified '`ADSML-PR-ID-6744`' – in this example, the unique identifier of the AdsMLEnvelope Processor that received the message. The error is described as '`Unknown Format type not agreed in PPA.`', meaning that the message contained a format type that was not specified as allowed in the associated PPA. (Note that the language of the description has not been identified by a *`language`* attribute on `ErrorDescription`, and so the description language is identified by that attribute's default value of '`en-us`' - American English.)

The `ErrorLocation` element identifies the location in the message where the error was reported using its attributes. The *`documentNode`* attribute identifies the node where the error is located by an XPath expression of '`AdsMLEnvelope/Item[1]/ItemContent/Format`'. The *`itemInEnvelope`* attribute identifies the `Item` in which the error is located as the `Item` with the identifier of '`adsml.org:2004-01-01:801`'. The *`lineInFile`* attribute identifies the line number of the error location as being '59'. The *`positionInFile`* attribute records the character number count from the beginning of the file to the error location, recording it as '2065'. The *`positionInLine`* attribute records the character number count of the error location in the line identified by the *`lineInFile`* attribute as '`49`'.

The `OriginalMessageHeader` element is empty, signifying that no encoding has been applied to the enclosed original message. In this example, the `OriginalMessage` element contains the faulty message nested inside a CDATA section and so no encoding has been applied to the enclosed message.

The `SuccessfullyRoutedItems` element identifies the `Item`s that have been successfully routed. In this case two `Item`s have been routed, the `Item`s identified as those with the *`itemId`* values of '`adsml.org:2004-01-01:802`' and '`adsml.org:2004-01-01:803`'.

```
<Response responseId="adsml.org:2004-01-01:001021">
 <ResponseHeader>
  <ResponseType>MessageErrorDetected</ResponseType>
  <MessageRef messageID="adsml.org:2004-01-01:{C65E819C-5585-11D7-ACA0-
00B0D022396B}" sendCountNo="1" status="Production"
adsml:transmissionDateTime="2003-08-29T09:30:47-05:00"/>
  <adsml:Priority>1</adsml:Priority>
  <Destination>
   <BusinessEntityId>
    <BusinessEntityIdClass xsi:type="adsml-
cv:AdsMLOrganizationIDClassCV">AB</BusinessEntityIdClass>
    <BusinessEntityIdValue>ag18</BusinessEntityIdValue>
   </BusinessEntityId>
  </Destination>
  <LastProcessedBy>
   <BusinessEntityId>
     <BusinessEntityIdClass xsi:type="adsml-
cv:AdsMLOrganizationIDClassCV">AB</BusinessEntityIdClass>
    <BusinessEntityIdValue>ag19</BusinessEntityIdValue>
   </BusinessEntityId>
  </LastProcessedBy>
 </ResponseHeader>
 <ResponseContent>
  <MessageError>
   <ErrorDetails type="PPA" detectedBy="ADSML-PR-ID-6744">
    <ErrorDescription>Unknown Format type not agreed in
PPA.</ErrorDescription>
    <ErrorLocation documentNode="AdsMLEnvelope/Item[1]/ItemContent/Format"
itemInEnvelope="adsml.org:2004-01-01:801" lineInFile="59"
positionInFile="2065" positionInLine="49"/>
   </ErrorDetails>
   <OriginalMessageHeader/>
   <OriginalMessage><![CDATA[<?xml version="1.0" encoding="UTF-
8"?><AdsMLEnvelope xmlns="http://www.adsml.org/adsmlenvelope/v1.1"
language="en-us"><Header messageId="adsml.org:2004-01-01:806891"
responseRequired="false" status="Production" sendCount="1"
firstTransmissionDateTime="2003-08-29T09:30:47-05:00"
transmissionDateTime="2003-08-29T09:30:47-05:00" adsmlProcessorId="ADSML-PR-
ID-545">...</AdsMLEnvelope>]]></OriginalMessage>
   <SuccessfullyRoutedItems>
    <SuccessfullyRoutedItem>adsml.org:2004-01-
01:802</SuccessfullyRoutedItem>
    <SuccessfullyRoutedItem>adsml.org:2004-01-
01:803</SuccessfullyRoutedItem>
   </SuccessfullyRoutedItems>
  </MessageError>
 </ResponseContent>
</Response>
```

3.5 Common definitions

Common definitions are AdsMLEnvelope defined element or type definitions used in more than one context within the AdsMLEnvelope Schema.

3.5.1 Element: BusinessEntity

The `BusinessEntity` element is used to record the standard organization identifier(s) that trading partners have agreed (in their TPA and PPAs) to use as identifiers when exchanging data. Each identifier is recorded as a plain string and the type of organizational identifier used given by a required *`class`* attribute. The `BusinessEntity` element is declared as `BusinessEntityType`. See BusinessEntityType for the definition of the `BusinessEntityType`.

```
<xs:element name="BusinessEntity" type="BusinessEntityType"/>
```

Instance Example – BusinessEntity element

For an example of an instance where the `BusinessEntity` element is used to record organization identifier information, see 'Instance Example – Header element'.

3.5.1.1 Type: BusinessEntityType

The `BusinessEntityType` provides a content model used to record one or more organizational identifiers for a business entity using a sequence of mandatory and repeatable `BusinessEntityId` element(s). Each `BusinessEntityId` element records an organizational identifier by type and value using a sequence of `BusinessEntityIdClass` and `BusinessEntityIdValue` elements.

Name	Occurs	Type	Description
`BusinessEntityIdClass`	1-1	`adsml:IDLabelRootType`	Records the type of organizational identifier used to identify the business entity as a string. The `BusinessEntityIdClass` element is declared as `adsml:IDLabelRootType` and so a controlled vocabulary can be specified for use in this attribute context.
`BusinessEntityIdValue`	1-1	`adsml:LongStringType`	Records the organizational identifier used to identify the business entity as a string.

The following XML Schema fragment defines the `BusinessEntityType` element,

```
<xs:element name="BusinessEntityId" type="BusinessEntityIdType"/>

<xs:complexType name="BusinessEntityIdType">
 <xs:sequence>
  <xs:element name="BusinessEntityIdClass" type="adsml:IDLabelRootType"/>
  <xs:element name="BusinessEntityIdValue" type="adsml:LongStringType"/>
 </xs:sequence>
</xs:complexType>
```

Instance Example – BusinessEntity element and BusinessEntityType

For an example of an instance where the `BusinessEntityType` is used to record organization identifier information, see, 'Instance Example – Header element'.

3.5.1.2 Element: MessageClass

Defines the message class of the message carried as the Item content of an AdsMLEnvelope `Item`. This enables an `Item` to be identified as carrying a standard business transaction, an administrative error response, or an administrative acknowledgement response message.

When sending an administrative response message, the `Item` **MUST** contain `ItemType` sibling element(s) with identical values as those found in the original message that is being responded to. In error situations where the type of the business message received cannot be decided, the `ItemType` **MUST** have the value `'ZZ-ERR'`.

The message class is recorded using enumerated values of '`BusinessTransaction`', '`MessageReceivedAcknowledgement`', or '`TechnicalError`' as required. '`BusinessTransaction`' is the default value. See MessageClassCV.

```
<xs:element name="MessageClass" type="adsml:MessageClassCV"/>
```

For an example of an instance where the `MessageClass` element is used to record message classification, see 'Instance Example – ItemHeader element'.

4 AdsMLEnvelope Controlled Vocabularies

The following controlled vocabularies can be substituted and used for root type contexts in the AdsMLEnvelope. The AdsML Controlled Vocabularies are defined in the AdsML Controlled Vocabulary schema and are the official AdsML controlled vocabularies recommended for use in AdsML messages.

Controlled vocabularies specific to the AdsMLEnvelope are defined in the envelope main schema. All controlled vocabularies defiend in the AdsML Controlled Vocabularies schema are given an 'adsml-cv:' prefix, controlled vocabularies specific to the AdsMLEnvelope are defined in the AdsMLEnvelope namespace.

The following AdsML controlled vocabularies are used in the AdsMLEnvelope,

Controlled Vocabulary	**Usage context**
`adsml-cv:AdsMLOrganizationIDClassCV`	`BusinessEntityIdClass` element (optional)
`ContactInfoTypeCV`	`ContactInfo` element context (optional). See `adsml-cv:AdsMLContactInfoClassCV`
`adsml-cv:AdsMLContactInfoClassCV`	*`class`* attribute context of the `ContactInfo` element context (optional). (Used through the `ContactInfoTypeCV`)
`adsml-cv:AdsMLContactRoleCV`	`Role` element context (optional)
`EnvelopeErrorCV`	*`type`* attribute of the `ErrorDetails` element context (mandatory). See the AdsMLEnvelope Part 1 Processing Model for more information about error handling.
`adsml-cv:AdsMLEncodingCV`	`Encoding` element context (optional)
`adsml-cv:AdsMLEncryptionMethodCV`	`EncryptionMethod` element context (optional)
`adsml-cv:AdsMLFormatTypeCV`	`Format` element context (optional)
`ItemActionCV`	`Action` element context (optional).
`adsml:MessageClassCV`	`MessageClass` element context (mandatory)
`adsml:PriorityType`	`Priority` element child of `Header`, `ItemHeader`, `ResponseHeader` element contexts (mandatory) *`priority`* attribute of `Contact` element context (mandatory)
`EnvelopeResponseCV`	`ResponseType` element context (mandatory)

`EnvelopeStatusCV`	*`status`* attribute of `Header` and `MessageRef` element contexts (mandatory)

See the '*AdsMLEnvelope Part 1 Processing Model*' f or more information about usage of CVs defined in AdsMLEnvelope. For external CVs, see the AdsML Controlled Vocabularies and Type Library documentation for further information.

4.1 The root types

The AdsMLEnvelope specifies the following root types for use in element and attribute contexts,

Root type	**Usage context**
`ContactInfoRootType`	`ContactInfo` element context
`ItemActionRootType`	`Action` element context

All other root types used in the AdsML Envelope are defined in the AdsML type Library. See the The AdsML Controlled Vocabularies documentation for further information (Section 1.7 Accompanying documents).

4.1.1 ContactInfoRootType

Root type for the `ContactInfo` element context. `ContactInfoRootType` records contact information as a string of `adsml:StringType` data type.

The optional *`class`* attribute **MAY** be used to classify the type of contact information being recorded. The *`class`* attribute is declared as `adsml:CodeRootType` and so can be restricted to take specific values if required.

```
<xs:complexType name="ContactInfoRootType" block="extension">
 <xs:simpleContent>
  <xs:extension base="adsml:StringType">
   <xs:attribute name="class" type="adsml:CodeRootType"/>
  </xs:extension>
 </xs:simpleContent>
</xs:complexType>
```

4.1.2 ItemActionRootType

Root type for the `Action` element context. `ItemActionRootType` records the type of action that took place as a token of `ShortTokenType` data type.

```
<xs:simpleType name="ItemActionRootType">
 <xs:restriction base="adsml:ShortTokenType"/>
</xs:simpleType>
```

5 Appendix A: Acknowledgement for contributions to this document

Acknowledgement and thanks for contributions to this document are also due to,

- The AdsML Technical Working Group
- Reviewers of the Last Call Working Draft,
- Israel Viente (Vio Worldwide Limited) israel_viente@il.vio.com
- Martin Bryan. (IS-Thought) martin@is-thought.co.uk
- David Allen. (Former Managing Director of IPTC) mdavidallen@onetel.net.uk