AdsML® Framework for E-Commerce Business Standards for Advertising

AdsMLFinancials 1.5
Part 1
Usage Rules & Guidelines

Document Authors: AdsML Technical Working Group

Document ID: AdsMLFinancials-1.5.0-SpecP1Usage-AS-1

Document File Name: AdsMLFinancials-1.5-SpecP1Usage-AS.pdf

Document Status: Approved Specification

Document Date: 15 April 2010

Draft Number: 1

Table of Contents

1 AdsMLFinancials Standard Documentation

1.1 Document status and copyright

This is the Approved Specification of the *AdsMLFinancials 1.5 Part 1 Usage Rules & Guidelines*.

1.2 Non-Exclusive License Agreement for AdsML Consortium Specifications

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 AdsMLFinancials-1.5.0-SpecP1Usage-AS-1, is freely available. It is located at the AdsML website at http://www.adsml.org/.

1.5 Purpose of this document

This document provides rules and guidelines for how to use the messages defined in the AdsMLFinancials standard. AdsMLFinancials is an XML-based language used for encoding and routing financial documents pertaining to advertising transactions, in particular invoices and credits.

1.6 Audience

The intended audience for this document is primarily user and vendor organizations who seek to implement the AdsMLFinancials standard in their workflows, advertising systems, or software products. Those assessing the conformance of vendor products to the 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 provides rules and guidelines for using AdsMLFinancials messages to address specific business requirements. A companion document, *AdsMLFinancials – Part 2 - Specification & Schema*, serves as the reference guide to the AdsMLFinancials schema. They are meant to be read together.

In addition, elements and structures that are used in multiple AdsML schemas are documented in the *AdsML Type Library* specification. AdsMLFinancials makes extensive use of such structures, therefore the *Type Library* specification is an essential reference.

All three 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)

The key word "**DEPRECATED**" is used to indicate that structures are being phased out of the AdsML specifications. Structures marked as **DEPRECATED** will be removed in the next major schema upgrade and should not be used in new implementations.

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 using 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 as detailed above. For example, '`ExampleInstanceFile.xml`', '`ExampleSchemaFile-1.0.xsd`', '`ExampleSchemaFile-1.1.xsd`'.

In some cases, element names mentioned in usage guidelines and narrative text in this document do not include their namespace prefix. For example, the element `adsml-bo:BookingInformation` is often referred to as simply

'BookingInformation'. This simplification is provided in order to make the text easier to read. Element names in code fragments are always shown with their full namespace prefix.

1.8.3 Typographical conventions

Element and type names are given in Courier font as, for example, AdOrder.

Attribute names are given in italicized Courier font as, for example, *messageCode*.

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

1.9 Change History

Version	Date	Changes	Editor
1.5.0 AS 1	15 April 2010	First AS release of version 1.5. Earlier change log removed.	JC, TS
1.0.1 AS 1	30 May 2008	Maintenance release.	UW, TS
1.0 AS 1	10 October 2007	First AS Release. Earlier change log removed.	UW, TS

1.9.1 Changes in version 1.5.0

Version 1.5.0 is a major upgrade to the specification which contains changes that are not backwards compatible with the previous release of AdsML Financials, version 1.0.1. The change delta between these versions is recorded here.

AdsMLFinancials Version 1.5 is compatible with AdsMLBookings Version 2.5 and supports invoicing for interactive (i.e. 'online') advertisements.

The BookingInformation structures have been updated to align with AdsMLBookings Version 2.5 – the variants for .Generic, .Insert, and .NewspaperMagazine have been updated and a new structure for .Interactive has been added. The Item.AdvertisementPublication structure has been extended with more booking information, in line with changes to AdsMLBookings.

Other significant changes to note are,

- The Distribution element has been renamed DistributionTarget and is now used together with a new PlacementTarget to specify what has been ordered and how it is to be distributed by the publisher to its intended audience
- Changes to the AppearanceInformation structure have been made in order to convey the MediaType. See the AdsMLProofOfPublication Version 1.5 Specification & Schema, where AppearanceInformation is defined, for more information.
- The deprecated element AdditionalServices has been removed. It was replaced in Framework 3.0 with the more expressive structure adsml:AdditionalService.

1.10 Acknowledgments

This document is a product of the AdsML Technical Working Group.

Primary authorship and editing was performed by,

- Tony Stewart (RivCom) – tony.stewart@rivcom.com
- Ulf Wingstedt (CNet Svenska AB) ulf.wingstedt@cnet.se

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

1.11 The AdsML Consortium

The documents comprising the AdsML standard were written by the AdsML Technical Working Group, a committee charged with creating the consortium's technical deliverables, and then approved by the entire membership.

More information about the consortium can be found on the consortium's website: www.adsml.org.

2 Introduction

The AdsMLFinancials standard has been developed by the AdsML Consortium in order to provide a global standard for the exchange of financial documents pertaining to advertising transactions, including invoices, credits, debits, claims, statements and payment notifications.

AdsML provides an XML framework, called the "AdsML Framework", for unifying and extending XML advertising standards. Where earlier advertising standards for e-commerce such as IfraAdConnexion or CREST focused on specific parts of the overall advertising process, the AdsML specifications fill in the gaps between such standards and specifications, extend their reach and encourage convergence when they overlap. In this line of effort, the AdsMLFinancials standard has been developed by the AdsML Consortium as the preferred approach to handle financial documents pertaining to advertising transactions.

It is important to note that the AdsML Consortium does not intend to provide a new format for financial documents that will replace all the other formats currently in use in the world. Rather, AdsMLFinancials is designed to co-exist with other financial formats where they are being used, while providing a complete solution for those who need one.

AdsMLFinancials relies on earlier experience and standards that have been embraced and extended in order to support current advertising business requirements. In addition, AdsMLFinancials has been designed with extensibility as an important objective in order to be able to grow with the business and support various business models and future requirements.

An important issue in enabling automatic business message flows is the use of a common well-defined message choreography. One of the main components in the AdsML Framework is a set of business process models and related documentation that includes a definition of common process models for the workflows of selected advertising classes (*AdsML Advertising Component Interactions Analysis).* This first release of AdsMLFinancials supports a subset of the business messages from the Financial Documents group (FD-*).

Note: This release of AdsMLFinancials supports Invoice and Credit messages. Support for other types of financial messages may be added in later releases of the specification.

2.1 Implement only what you need

The AdsML Framework aims to provide advertisers, publishers, broadcasters and their suppliers with a consistent toolkit of standards, messages and transactions that can be used to automate every aspect of the advertising supply chain, in any media, anywhere in the world. This means that even though it is still incomplete, the Framework already contains more standards and message types, and can convey more types of information, than any single organization is likely to need.

In order to implement AdsML-based e-commerce, therefore, trading partners and their vendors (or industry associations acting on their behalf) are expected to review the AdsML Framework and decide:

- Which AdsML standards they will implement within their particular region or business activity
- Within those standards, which business transactions they will support (this determines the types of messages they will exchange)

- Within those messages, which types of information they will include (this determines the optional structures that they will implement)
- Within those types of information, which specific data values they will “control” (this determines their use of controlled vocabularies).

Each AdsML standard defines its mandatory and optional components, and where appropriate, each provides a Configuration Checklist to help users discuss and agree on the features and functionality that they will implement. These implementation decisions can be agreed privately between the trading partners, and/or codified in a formal “profile” which is made publicly available in order to encourage interoperability.

Based on their customers’ implementation decisions, vendors can decide which types of AdsML functionality to implement in their systems. In order to market a system’s AdsML capabilities, a vendor might indicate that it supports specific named Profiles, and/or the vendor might use the relevant Configuration Checklist(s) to describe the supported capabilities.

Further information about these concepts can be found in *AdsML E-Commerce Usage Rules & Guidelines*, in the *Advertising Components Interactions Analysis*, and in the Specification for each standard.

NOTE: Even though you can implement just those portions that you need, all of the standards and features in the AdsML Framework are designed to work together as a cohesive whole, in that they share common technical components and a common approach to advertising e-commerce that makes them "AdsML".

2.2 Use of the AdsML Envelope is optional, but recommended

AdsMLFinancials uses the AdsML business process model as a foundation for its message types. It also imports and reuses controlled vocabularies and the type library from the Framework. However, it is important to note that AdsMLFinancials does not require use of, nor support for, the AdsML Envelope standard. The actual transfer of AdsMLFinancials messages can be performed by arbitrary method and software application, with or without the use of the AdsML Envelope. For instance, an AdsMLFinancials message can be transmitted using other envelopes such as ebXML or BizTalk or directly by SOAP, FTP, HTTP or SMTP services.

But it should nevertheless be noted that as the AdsML Envelope has been particularly developed to support message transfer within the advertising business and it is **RECOMMENDED** for use with the AdsMLFinancials message format.

Please see the *AdsML Framework - Overview* and *AdsML E-commerce Overview* for a more thorough discussion about the AdsML approach to e-commerce.

2.3 Relationship to other advertising standards

2.3.1 Relationship to other standards in the AdsML Framework

AdsMLFinancials focuses on the processes involved in the exchange of financial documents, and is intended to be used in conjunction with AdsML standards covering other areas in the advertising work flow, for example, AdsMLBookings and AdsMLProofOfPublication. However, use of these or any other AdsML standard is not required.

Within the financials area, AdsMLFinancials is designed to extend and embrace functionality previously covered, partly, by older standards. The design of the header and footer structures, as well as those parts of the line items which are media agnostic, was inspired by the Invoice document in Universal Business Language (UBL) version 2.0, and is, to the extent possible, compatible with it. These parts of AdsMLFinancials are not only media agnostic but even domain agnostic, in that they could be used for financial documents in any type of business.

The advertising-specific information in AdsMLFinancials has been pushed down into components of the package, in the `InvoiceLine` and `CreditLine`. AdsMLFinancials provides specific content models targeted for relating a line item to a published advertisement or its original booking, and for describing the details of the published ad as they relate to this invoice. These advertising-specific structures are not based on UBL, but instead, re-use standard AdsML components from elsewhere in the Framework. In addition, AdsML provides a generic structure for representing these types of information, in case it is not possible or desirable to use the advertising-specific components.

2.3.2 Relationship to non-AdsML standards

As noted above, AdsMLFinancials is not intended to replace all the other financial formats currently in use in the world. Rather, AdsMLFinancials is designed to co-exist with other financial formats where they are being used, while providing a complete solution for those who need one.

We have designed AdsMLFinancials so that it can be reused by other standards at either the line item level or the document level. In this manner a financial standard that is capable of incorporating structures from an external standard can import either the structure of an entire AdsMLFinancials document, or just the required AdsMLFinancials line-item structure, and use them as embedded payloads within the other standard's message format.

Further, the design of AdsMLFinancials has been inspired by the Universal Business Language (UBL), a widely used XML e-commerce standard for financial messages. Conceptually, we have done what UBL defines as a "compatible customization". The UBL model has been taken as the "baseline" model for an invoice and then has been restricted to simplify and remove optional content not required in the advertising context. As part of this customization, existing AdsML structures - e.g. for `PartyType` - have been used as-is, or were themselves restricted or extended for use in the invoicing context in order to achieve compatibility with the UBL model. Therefore, although the XML serialization will differ, the underlying model is compatible.

The model has then been extended to add specific AdsML-developed structures required by the advertising context. The logic is to localize specific advertising extensions at the line item level as far as possible, while modeling those structures such that the advertising-specific extensions are available for independent reuse in another standard, should that be required.

3 Business Messages Overview

AdsMLFinancials supports the business process model and message flow as proposed in the *AdsML Advertising Component Interactions Analysis*, a part of the AdsML Framework. In particular, AdsMLFinancials defines the set of business messages that belongs to the financial documents group (FD).

3.1 Invoices and Credits

There are two main classes of business messages defined in the current release of the schema:

- Invoices
- Credits

In addition, the Invoice format provides the ability for an invoice line to be either a charge or a credit, effectively replicating the functionality of a standalone Credit message as a line item on the invoice.

In both of these message classes there is a request-response pair of transactions, although the response functionality is relatively limited and is intended to support acknowledgement, rejection or request for clarification. For example, a publisher submits an invoice or a credit, and the recipient responds by acknowledging receipt of that invoice and perhaps by refusing to accept to pay it.

It is also possible to send a follow-up message conveying the status of an Invoice document that has previously been received. This can be done either in response to an enquiry about the status of that invoice or as a standalone, event-triggered broadcast message.

Future releases of AdsMLFinancials are expected to support messages that provide additional functionality, for example, the ability to lodge a query or claim against the charges on an invoice. For the time being such communications must be conveyed by non-AdsML communications.

The complete list of business messages supported in AdsMLFinancials 1.5 is:

Message Code	Message Name
FD-NV	Invoice
FD-NVR	Invoice Response
FD-NVS	Invoice Status
FD-NVSE	Invoice Status Enquiry
FD-CR	Credit Note
FD-CRR	Credit Note Response

Note: this list of messages reflects the subset of the FD group messages that have been implemented in AdsMLFinancials 1.5.

The usage of other types of financials messages defined in the ACIA will be described when support for those messages is given by the AdsML Financials standard.

For the complete list of FD group messages see the *Advertising Components Interactions Analysis.*

3.1.1 Debits

AdsML Invoice messages are capable of conveying non invoice-related charges, and therefore can serve the function of Debit documents sent by an invoicer to a payer in an advertising workflow. When an invoicing party wishes to send a debit to a paying party, they should send an AdsML Invoice message. Trading partners can agree to use the `DocumentType` code in the invoice header to indicate whether it is a "Debit" rather than an "Invoice".

Note: The AdsML Technical Working Group considered either naming the existing Invoice message "Debit" rather than "Invoice", or creating an additional "Debit" message type, but elected in the end to use the name "Invoice" for all debit-related messages. We made this choice because the majority of debit documents in an advertising workflow are, in fact, invoices, and because most people are more comfortable talking about Invoices than Debits when they discuss the advertising financial workflow.

3.2 Message components

For each business message type supported by AdsMLFinancials, there is a corresponding content model in AdsMLFinancials.

Different messages have different content models. However, there are also many similarities and common content models are reused in several message types.

A new financial document must include a reference key (the `DocumentIdentifier`). The general rule for reference keys is that the originating party provides a reference key with the message and that this key is then used in the response and any further transaction regarding this document.

Both Invoice and Credit Note documents share a common structure consisting of a header, a set of line items, and a footer. Similar header and footer structures are used by both document types. Their line items, however, vary more. For example, while the line items on an invoice can be any combination of `InvoiceLines` (requests for payment) `CreditLines` (acknowledgement of credits that have been given) and `InformationalLines`, the line items on a Credit Notification must all be `CreditLines`.

3.3 Information contents

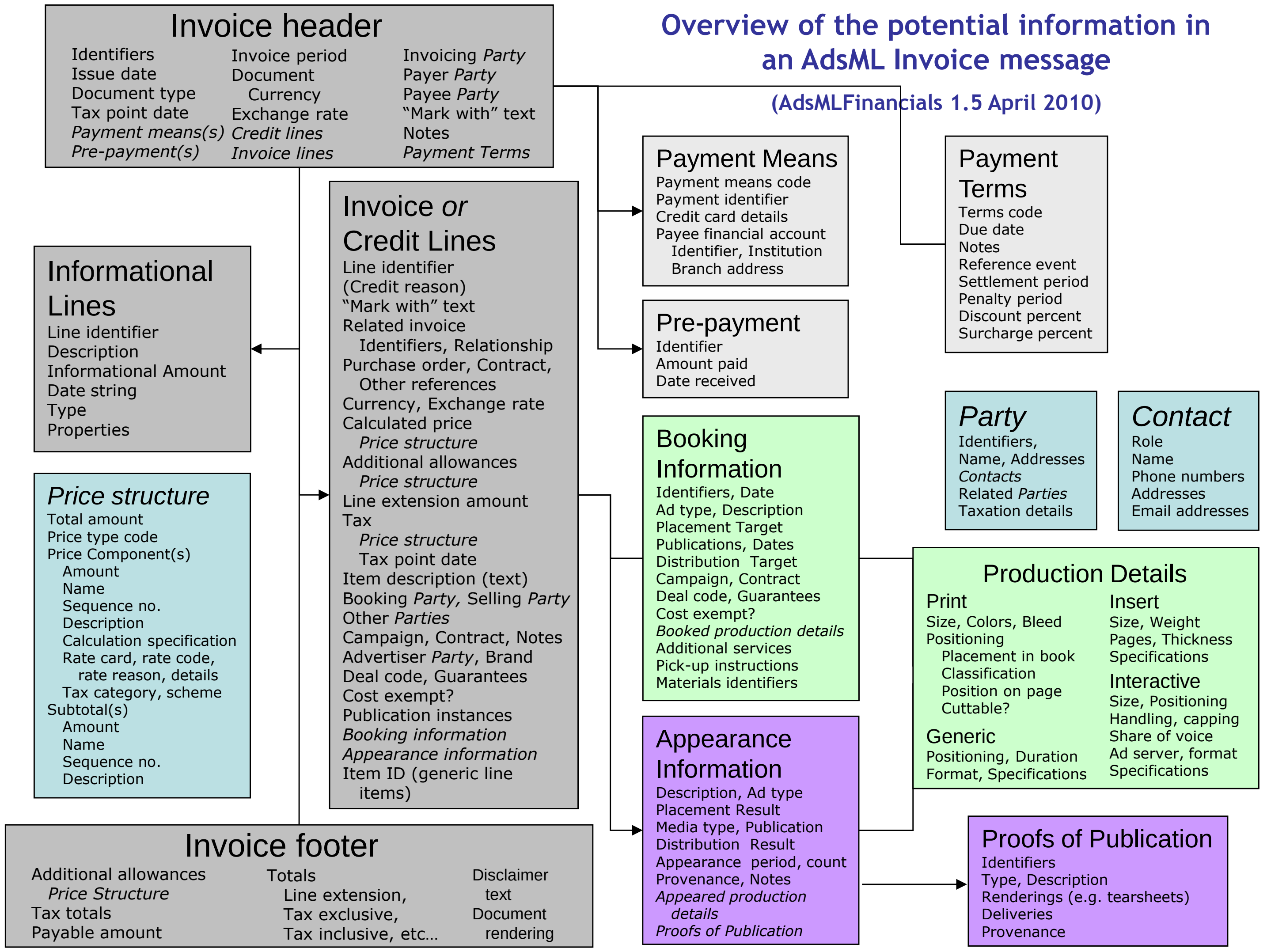
Here are diagrammatic overviews of the potential information in an Invoice or Credit Note message.

These views omit the generic message header as well as many of the smaller details, in order to see the main context-specific information "at a glance". Much of the information is optional, intended for use in specific circumstances – and some of it can *only* be used in those specific circumstances. Therefore a given message instance will not contain all of the information shown here.

Note also that some information can be recorded at both document and line item level. In general, information provided at the document level applies to all line items in the financial document. When the same type of information is also given at the line item level, it should override and replace the document level information for that particular line item. There should be no other implicit relationship between values for the same elements at the two levels.

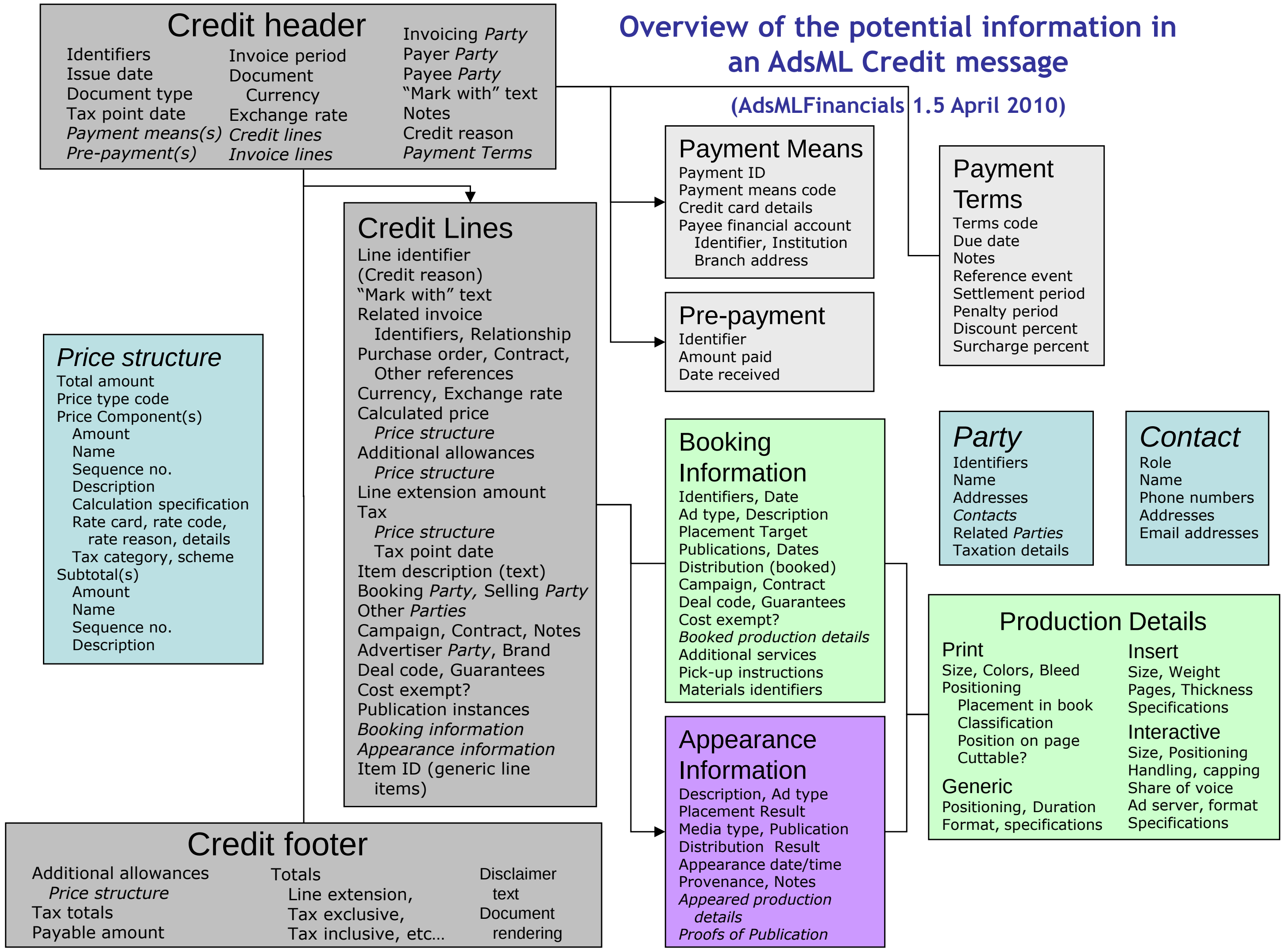
Overview of the potential information in an AdsML Invoice message

(AdsMLFinancials 1.5 April 2010)



Overview of the potential information in an AdsML Credit message

(AdsMLFinancials 1.5 April 2010)



4 Message Choreography

This is a normative section describing the expected message flow between communications partners in a financial documents transaction.

In addition, implementations of the AdsMLFinancials **MUST** support the specifications provided in the *AdsML E-commerce Usage Rules & Guidelines*.

AdsMLFinancials includes two main categories of messages:

- Business messages, i.e. messages such as new bookings, changes, cancellations and status requests that are part of the parties' advertising business.
- Administrative messages, i.e. house-keeping messages for the systems involved in exchange of business messages. Examples are error messages and receipts of received AdsMLFinancials XML files.

4.1 Administrative Messages – Acknowledgment and Error handling

Administrative messages are an integral part of the AdsML Framework. As a general case, for example, the recipient of an AdsML business message is expected to send an administrative response to that message promptly upon receipt of the business message, in order to indicate that the business message was received and to convey any AdsML-level errors that may have been found in it. At the same time, the contents of the business message are forwarded to the appropriate application, from which (in due course) a business response message will be generated.

The rules governing administrative messages and error handling are generic and apply to the entire AdsML Framework. These rules **MUST** be followed when sending and receiving AdsMLFinancials messages. For a description of administrative messages and error handling, please see *AdsML E-commerce Usage Rules & Guidelines*.

4.2 Testing

The rules governing test messages are generic and apply to the entire AdsML Framework. These rules **MUST** be followed when sending and receiving test AdsMLFinancials messages. For a description of test messages, please see *AdsML E-commerce Usage Rules & Guidelines*.

4.3 Response Modes

The preferred messaging model is the Request-Response model as specified in the *AdsML E-commerce Usage Rules & Guidelines*.

However, since legacy applications may have limited ability to provide appropriate responses, it is also possible to use a model where only requests and administrative responses are transmitted, assuming an acceptance on the receiver's side. If a problem occurs such that a message cannot be accepted, it has to be solved manually. This kind of model is called a datagram model. For more information about datagram messaging, see the *AdsML E-commerce Usage Rules & Guidelines.*

As a summary:

1) Implementations of AdsMLFinancials **SHOULD** apply the full Request-Response model
2) If agreed by communication parties, implementations **MAY** use a datagram model (no business level responses required), and if so, they must agree on which direction(s) of datagram messaging they will support.

4.4 Business Messages

Each business message type is identified by a message code that specifies if the message is, for instance, an invoice, a credit, or responses to these messages. AdsMLFinancials supports a subset of business messages as defined in *AdsML Advertising Component Interactions Analysis*, namely messages from the Financials group (FD).

The message type is expressed as a code value for the `messageCode` attribute on business message elements such as `Invoice`. The code values are defined as the code values used in the AdsML Framework.

The sections below give a summary of the messages in each subgroup "Invoices" and "Credits". For more information, see also the reference section for each message element (named as the message name in CamelCase).

4.4.1 Message exchange model

4.4.1.1 Delivery of the financial document

This release of AdsML Financials supports a single, simple message exchange model for financial documents. The publisher or invoicing party packages their financial document and transmits it as an AdsML message to the paying party. The payer responds with an administrative response, and optionally (if request-response choreography is used), a business level response message. For example:

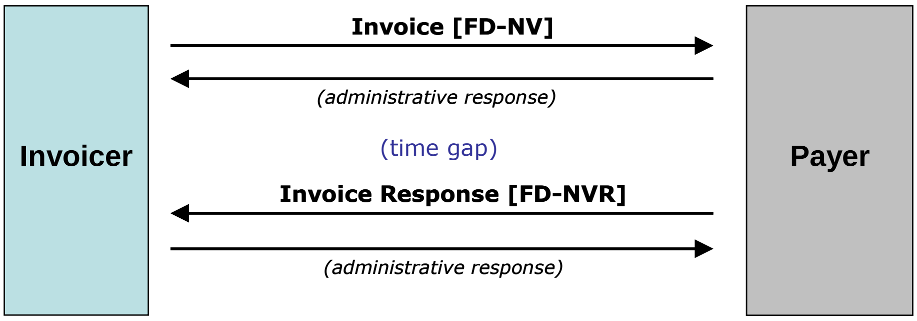

Invoicer
Invoice [FD-NV]
(administrative response)
(time gap)
Invoice Response [FD-NVR]
(administrative response)
Payer

figure: Invoice delivery, request-response mode

The administrative response is a technical acknowledgement that a message was received and indicates whether or not it is valid according to the AdsMLFinancials schema. A business level response, on the other hand, should include information about the business validity of the payload of the message, i.e. if the invoice was accepted, received by the financial system, queued for further processing and order reconciliation, or any other business response.

It should be noted that under no circumstance should it be assumed that receiving an administrative response or a business level Invoice Response from the Payer indicates agreement by the payer to pay the invoice, unless that interpretation has been explicitly agreed in advance by the trading partners.

Once a financial document has been delivered and accepted, it is not possible to send a changed version of the same financial document, or to "cancel" it. Such real-world requirements are accomplished by sending a subsequent financial document (either an invoice or a credit) which references the previous document and accomplishes the desired goal.

NOTE: Future versions of AdsMLFinancials are planned to include additional messages that can be used after an Invoice message has been transmitted, in order to keep track of the ongoing invoice and payment process including reconciliations, payments, claims etc.

4.4.1.2 Providing status information

Following the initial delivery of an invoice, it is possible for the document's recipient to deliver updates to the sender about the status of that invoice. For example, a payer might initially accept an invoice into its system in a "pending" state, and then later send a status update to the invoicer which indicates, for example, that there are questions about the invoice that will need to be resolved before it can be approved for payment. This is done by sending an Invoice Status message:

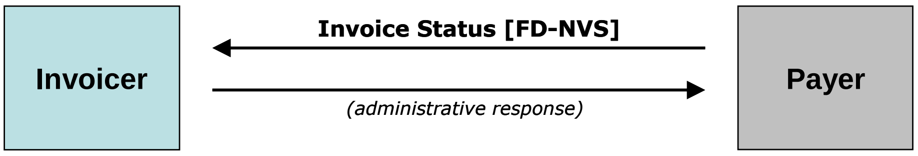


figure: Delivery of status information about an invoice

An invoicing party may also wish to explicitly request status information about a previously-delivered financial document. This is done by sending a Status Enquiry message referencing the original document, upon receipt of which the Payer is expected to reply with a Status message that answers the enquiry:

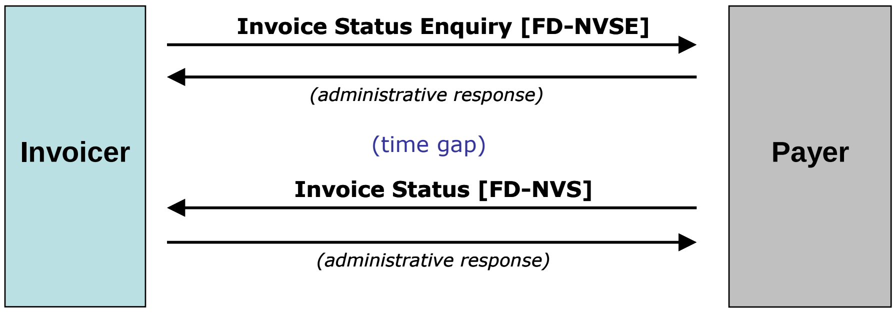

Invoicer
Invoice Status Enquiry [FD-NVSE]
(administrative response)
(time gap)
Invoice Status [FD-NVS]
(administrative response)
Payer

figure: A Status Enquiry followed by a response to the enquiry

The only available response to a Status message is an Administrative Response message indicating receipt of the message.

4.4.1.3 Relationship to Proof of Publication messages

In many cases, an Invoice (FD-NV) message will be preceded, accompanied or followed by a form of proof of publication, for example, a digital or hard-copy tearsheet. There may be many proofs of publication relating to a single invoice, because an invoice can represent many publication events. The proof of publication information can also be delivered in an AdsML message. Proof of Publication (PO-PB) messages are always sent using the datagram model, which means that there is no business-significant response to them.

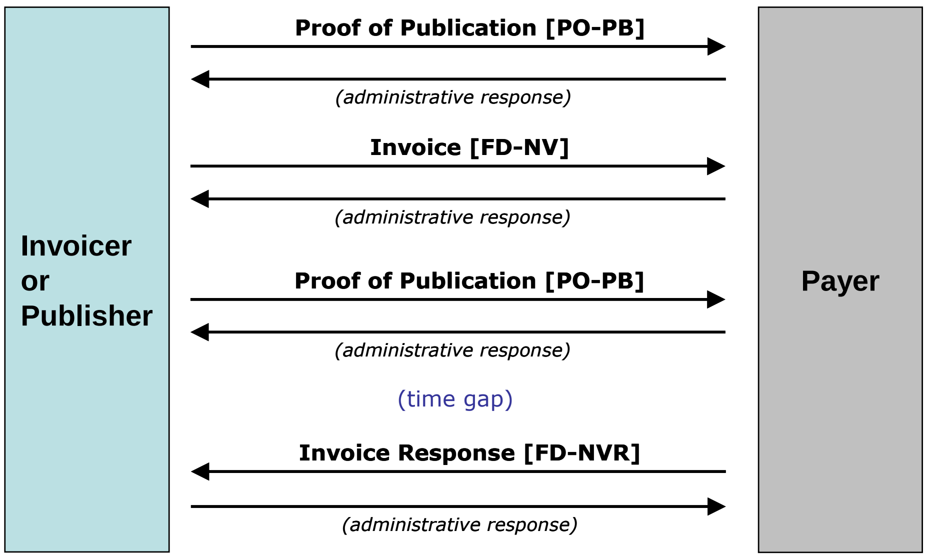

Invoicer or Publisher
Proof of Publication [PO-PB]
(administrative response)
Invoice [FD-NV]
(administrative response)
Proof of Publication [PO-PB]
(administrative response)
(time gap)
Invoice Response [FD-NVR]
(administrative response)
Payer

figure: An invoice and two related proofs of publication, in no particular sequence, followed by the invoice response

Once the paying party has received the invoice and any relevant proofs of publication, it can proceed to approve the invoice for payment.

4.4.2 Invoice Messages

An invoice is a request for payment, or an acknowledgement of payment previously received, that is sent by an invoicing party to a paying party, and that typically includes pricing and taxation information and a description of the products and/or services whose provision is the basis for the requested payment.

In order to support business practices that are common in some regions, an AdsML invoice can also contain line items that serve as embedded credit notes, in that they reflect credits which the sender of the invoice has applied to the recipient's account.

4.4.2.1 Datagram messaging from invoicer to payer

1) The invoicing party sends an Invoice (FD-NV) message to the Paying Party. Once the invoicer has received an Administrative Response from the payer, the payer is assumed to have received the invoice. (Note that this does not mean the payer has agreed to pay the invoice.)

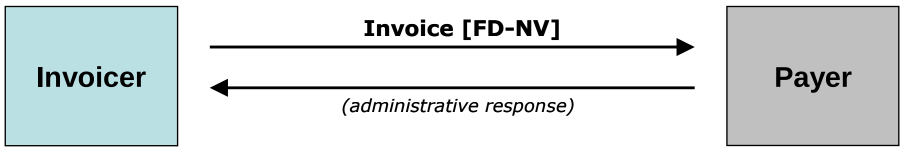


figure: Invoice delivery in datagram mode

4.4.2.2 Datagram messaging from payer to invoicer

1) A payer **MAY** send an Invoice Status (FD-NV) message spontaneously at any time during the invoice's lifecycle.

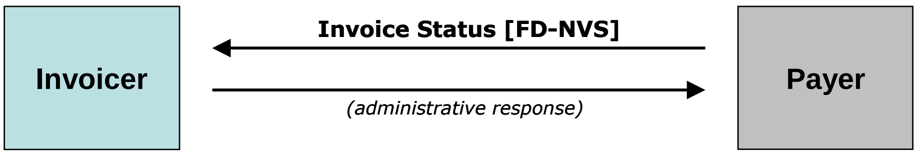


figure: Delivery of status information

4.4.2.3 Request-Response messaging model

1) The invoicing party sends an Invoice message (FD-NV) to the Paying Party, which **MUST** result in an Invoice Response (FD-NVR) message that either confirms or denies receipt of the invoice at the business level. The sequence is as follows:
   a. Immediately upon receipt of the Invoice, the payer sends an automated administrative response message to the invoicer, indicating that the message has been received and is technically ok.
   b. Subsequently, when the business information in the invoice has been reviewed, the payer sends an Invoice Response (FD-NVR) message to the invoicer. This message includes the business level response to the original invoice message, such as whether the Invoice was successfully loaded into the payer’s financial system, or conversely, that a business-level issue has arisen which will impact processing.
   c. Immediately upon receipt of the Invoice Response, the invoicer sends an automated administrative response message to the payer, indicating that the Invoice Response message has been received.
2) Both parties can now act on the information that was contained in the Invoice Response business message. But note that an Invoice Response (FD-NVR) message does not necessarily mean that the payer has agreed to pay the invoice.

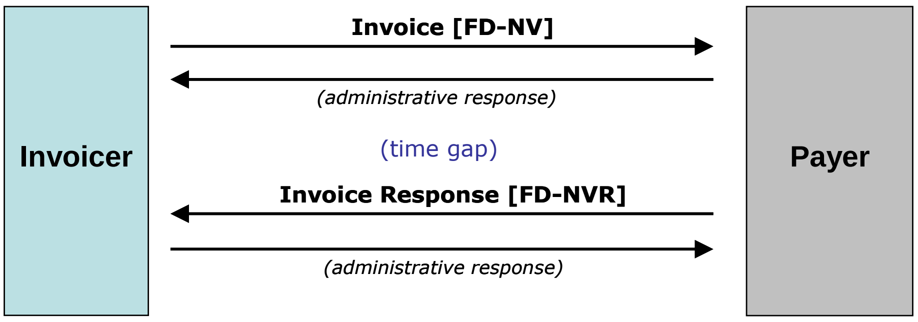

Invoicer
Invoice [FD-NV]
(administrative response)
(time gap)
Invoice Response [FD-NVR]
(administrative response)
Payer

figure: Invoice delivery, request-response mode

3) An Invoice Status Enquiry (FD-NVSE) **MUST** result in an Invoice Status (FD-NVS) message response.
4) If an Invoice Status message is a response to an Invoice Status Enquiry, it **MUST** reference the Invoice Status Enquiry’s message identifier and invoice identifier.

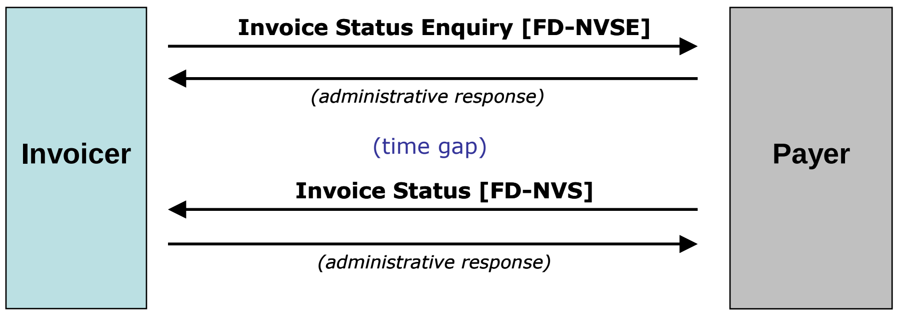

Invoicer
Invoice Status Enquiry [FD-NVSE]
(administrative response)
(time gap)
Invoice Status [FD-NVS]
(administrative response)
Payer

figure: Invoice status enquiry and response.

5) An Invoice Status (FD-NVS) **MAY** be issued by the payer without a previous Invoice Status Enquiry (FD-NVSE) having been received.

4.4.3 Credit Messages

A credit is a formal acknowledgement that is sent by an invoicer of advertising to a paying party to describe an adjustment that the invoicer has made to its internal record of its financial relationship with the payer – that is, its understanding of how much money is owed by the payer to the invoicer.

A Credit, as its name indicates, **MUST** describe an adjustment that benefits the message recipient. Usually the credit confirms an agreement that the parties have reached via non-AdsML communications. Usually it relates to an invoice (or a part of an invoice) that was previously sent by the invoicer to the paying party.

4.4.3.1 Datagram messaging from invoicer to payer

1) The invoicing party sends a Credit (FD-CR) message to the Paying Party. Once the invoicer has received an Administrative Response from the payer, the payer is assumed to have received the credit.

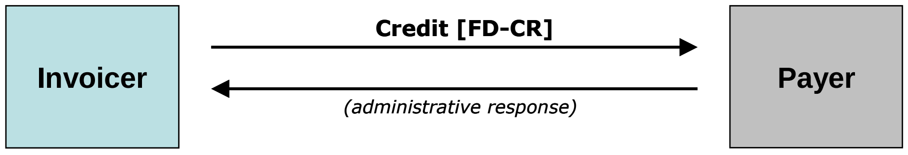


figure: Credit delivery, datagram mode

4.4.3.2 Request-Response messaging model

1) The invoicing party sends a Credit (FD-CR) message to the Paying Party, which **MUST** result in a Credit Response (FD-CRR) message containing a business level response. The sequence is as follows:

   a. Immediately upon receipt of the credit, the payer sends an automated administrative response message to the invoicer, indicating that the message has been received and is technically ok.

   b. Subsequently, when the credit has been reviewed, the payer sends a Credit Response (FD-CRR) message to the invoicer. This message indicates for instance whether the Credit was successfully loaded into the payer's system, or conversely, that a business-level issue has arisen which will impact processing.

   c. Immediately upon receipt of the Credit Response, the invoicer sends an automated administrative response message to the payer, indicating that the Credit Response message has been received.

2) Both parties can now act on the information that was contained in the Credit Response (FD-CRR) business message. But note that a Credit Response indicating that the credit has been successfully received does not necessarily mean that the payer has agreed to accept its terms.

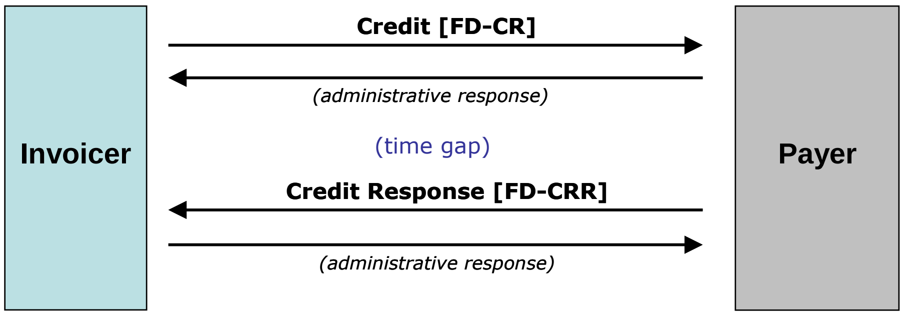


figure: Credit delivery, request-response mode

4.4.4 Message References – document and message identifiers

The AdsMLFinancials standard supports an asynchronous messaging model. For a general discussion, please see *AdsML E-commerce Rules & Guidelines*.

An invoice (FD-NV) message normally initiates an order reconciliation and payment process. During the life cycle of an invoice, it may be referenced by another invoice, by a debit or credit, or by a payment notification. In order to maintain the relationship between request-response message pairs, as well as between a series of messages regarding the same invoice, a stable identifier **MUST** be used during the complete suite of possible messages, i.e. the *document identifier* below. Response messages **MUST** use the same identifier as was used in the initiating message. The same is also true for other financial messages such as the Credit.

An implication of the above is that an advertising financial system **MUST** be able to store this identifier with the financial document in its internal data storage.

In general, each message has the following identifiers:

- *The document identifier*. An identifier first issued by the party that generates a financial document. The document identifier is the primary identifier for a financial document and **MUST NOT** change during the life of that document. Its structure **MUST** conform to the AdsML QID format, and it **MUST** be included in any transmitted message relating to that document, both responses and requests. The document identifier is called `DocumentIdentifier`.
- *The invoicing party's reference*. A reference identifier issued by the invoicing party in an invoice or credit transaction. The invoicer's reference conveys the internal identifier that the invoicer uses to reference this Invoice or Credit. The invoicer's reference is optional; however, once provided, its value **MUST** remain stable throughout the life of the Invoice or Credit. The invoicer's reference element is called `InvoicersReference` and is part of the `AuxiliaryDocumentReferences` structure.
- *The message ID*. A unique identifier for the business message. Each message ID **MUST** conform to the AdsML QID format and **MUST** be different from any other message ID. The message ID appears in the business message and is called *`messageID`*.

NOTE: As discussed below, the term "invoicing party" is also used to describe the sender of a Credit or Debit document. In those contexts the *invoicing party's reference* is the reference identifier that was assigned by the sender of the document.

Response messages need to identify the message they respond to as well as the message class of that message:

- *The "in response to" message ID*. A mandatory ID referencing the message that a response is about.

The ID appears in responses as the attribute *`inResponseToMessageID`*.

Please see the section on "Globally Unique Identifiers" in *E-Commerce Usage Rules & Guidelines* for information regarding how identifiers may be expressed using the AdsML QID type.

5 Usage of Business Messages

5.1 Relationship of message flow to the “invoicer” and “payer”

An Invoice, credit or debit message is always sent by an invoicing party to a paying party. These roles are used consistently throughout this document, and are embedded in various element names in the message format.

In an Invoice document the roles are self explanatory. In the context of a Credit, the “invoicing party” is the party that issued the credit and the “paying party” is the party that receives (and benefits from) it.

The most common situation is for the Invoicer to be either a publisher/broadcaster or a sales agency acting on its behalf, and the Payer to be a media buyer (which may be an advertiser acting as its own media buyer). However, the roles of “invoicer” and “payer” may be played by any party in the advertising workflow. In particular, a media buyer or advertiser may send an invoice, credit or debit message to a media seller or publisher, if such an action is required according to the current circumstances. In this case the message sender is acting as an “invoicer” and the recipient is acting as a “payer”, even if those are not their customary roles in the rest of the workflow.

Trading partners **MUST** agree in advance on which financial messages may flow between them, and in which directions.

5.2 Line items

All of the AdsML financial documents share a common structure consisting of a header, a set of one or more line items, and a footer.

The headers and footers of all the financial documents are quite similar. The header defines the type of document, the parties, the dates, the currency and exchange rate, and (if appropriate) the payment means and terms. The footer contains numeric totals for the document as a whole. Between these are the line items.

The overall document structure, and the design of the line items themselves, is designed to be quite flexible. This flexibility is found in the varied types of line items that can be included in a given financial document, and in the ability within a given line item to describe information in both simple and complex ways.

5.2.1 Types of line items

The line item structures vary according to the nature of the information that the line item is intended to convey. At the time of this writing there are line item structures for:

- **invoicing** information (representing a request for payment that the invoicer is making of the payer; also used to convey debits)
- **credit** information (representing a credit that the invoicer has applied to the payer’s account)
- **informational** lines (representing non-machine-processable information that the sender wishes to make available to the recipient, essentially an annotation structure that can convey any textual information the sender wishes to display to human beings on the receiving side)

Invoice and informational line items are used only in Invoice messages (FD-NV). Credit line items can be used in both Invoices (FD-NV) and Credits (FC-CR).

5.2.2 Line item flexibility

AdsML line items are designed to be extremely flexible in terms of the types and numbers of transactions that a single line item can represent.

An AdsML **invoice line** item can relate to a range of possible events including:

- The publication of a single advertisement including all costs
- A subset of the charges relating to the publication of an advertisement (for example, production charges)
- The publication of multiple advertisements that were included in the same booking
- The publication of all the advertisements in a given time period, for example an entire month of activity, which may represent either partial or total fulfillment of many different bookings
- Contractually-triggered chargeable events that are not tied to the publication of specific advertisements
- Arbitrary transactions that have nothing to do with advertising activities (although it should be noted that the AdsML line item structures are not optimized for such uses).

Similarly, an AdsML **credit line** can relate to a range of possible transactions including:

- A prior invoice
- A line-item on a prior invoice
- Any other type of financial document
- Contractually-triggered creditable events that are not tied to the publication of specific advertisements
- Arbitrary transactions that have nothing to do with advertising activities (although it should be noted that the AdsML line item structures are not optimized for such uses).

An AdsML **informational line** is essentially a non-machine-processable annotation element that can convey any textual information the sender wishes to display to human beings on the receiving side, for example: Balance Forward, Payment Received, Subtotals and Aging Receivables. It is meant to provide a bridge between the current human-oriented workflow and the e-commerce workflow that AdsML aims to enable.

5.2.2.1 Usage rules and guidelines

1. It is **RECOMMENDED** that an Invoice message (FD-NV) **SHOULD** contain at least one `InvoiceLine`.
2. It is **RECOMMENDED** that the `LineExtensionAmount` of an `InvoiceLine` **SHOULD** contain either a zero or positive value, so that the line item as a whole represents a debit to the Payer's account.
3. It is **RECOMMENDED** that the `LineExtensionAmount` of a `CreditLine` **SHOULD** contain either a zero or negative value, so that the line item as a whole represents a credit to the Payer's account.

4. An `InformationalLine` **MUST NOT** convey information that forms a material part of the current transaction. Such information **MUST** be conveyed in `InvoiceLines` and/or `CreditLines`.

5.2.3 Structured information in line items (including booking details)

This section discusses the ability of an AdsML Invoice or Credit line item to provide structured, machine-processable information which the recipient of the message can extract and use to drive automated reconciliation and payment approval. While such automated processing is a primary goal of AdsML (and indeed of all e-commerce), it is recognized that many systems are not capable of transmitting or receiving the detailed structures that would be required to accomplish this. Therefore, AdsMLFinancials supports several levels of structural formality in its line items, of which the three most common are described below.

5.2.3.1 Amount and description

The simplest (most informal) approach is to populate an AdsML Invoice or Credit line with just an amount, a name, and an optional textual description.

The description can be as simple or complicated as necessary. It can contain important cross-references such as booking, placement or order numbers. However, because these numbers will be in-line in the text, in most cases they will not easily be machine processable by the recipient.

This approach is analogous to sending a PDF of a current invoice. It automates the transmission of the information and makes it more accessible to the recipient, but it will require a human being or customized tools to process it.

5.2.3.1.1 Usage rules and guidelines

AdsMLFinancials provides a choice between two structures for describing the nature of a line item: `Item.Generic` and `Item.AdvertisementPublication`. Each of them contains an `adsml:Description` element, and therefore the "amount and description" approach can be implemented by use of either structure.

It is **RECOMMENDED** that if a line item relates to the publication of one or more advertisements in any medium, then `Item.AdvertisementPublication` **SHOULD** be used. If a line item relates to some other type of charge, then `Item.Generic` **SHOULD** be used instead.

5.2.3.2 Amount and description <u>plus</u> formal booking identifiers

This approach builds on the previous one, in that it primarily consists of an amount and textual description, but in addition the sender formally identifies the specific booking in question, and optionally identifies specific placement group(s), placement(s) and schedule entry(ies) from that booking. In this way, although the chargeable items are still expressed as a text string, the recipient's system can automatically retrieve the relevant document or information from their systems in order to facilitate the invoice reconciliation process.

5.2.3.2.1 Usage rules and guidelines

An AdsML invoice or credit line item **MAY** be formally associated with one and only one booking, although there is no limit to the number of placement groups and placements within that booking that can be referenced. The references can consist of AdsML QIDs and/or non-AdsML identifiers such as the buyer's and seller's booking or placement IDs. The association with a Booking is accomplished by populating the `BookingReference` element in the top level of `Item.AdvertisementPublication`. The associations with sub-parts of that booking are accomplished by populating one or more `AdvertisementPublicationInstance` elements, each of which contains a `BookingInformation` structure in which a Placement Group, Placement and Schedule Entry Identifier can be explicitly identified.

Because only one booking can be formally identified, parties wishing to provide structured information that associates an invoice or credit line item with more than one booking **MUST** split their information into multiple line items in such a way that each line item relates to all or part of only one booking.

5.2.3.3 Amount, description, identifiers <u>plus</u> structured details about the booking and publication of one or more advertisements

This final approach builds on the prior one by adding the ability to convey fully structured details about one or more advertisements whose publication triggered this line item.

An invoice or credit line can contain many sets of structured information, one for each instance of the publication or broadcast of an advertisement that is related to that line item. This arrangement allows for the invoicing of package buys, in which the publication of multiple advertisements (possibly in different publications or media) is invoiced at a single package price.

For each instance of a published advertisement, two types of information can be provided:

- A copy of relevant **booking information**, such as the publication name and ID, appearance date(s), targeting, size, and color as they were originally booked. This information describes how the advertisement was *intended* to appear at time of booking, which may not be the same as what actually happened. It is conveyed in `AdvertisementPublicationInstance/BookingInformation` inside `Item.AdvertisementPublication`, using some of the same structures that are found in AdsMLBookings.
- **Appearance information** which describes when, where and how the advertisement instance was *actually* published, and optionally contains or references one or more proofs (tearsheets) for that appearance. The structured information about when, where and how the advertisement was published is conveyed in `AdvertisementPublicationInstance/AppearanceInformation` inside `Item.AdvertisementPublication`. Each tearsheet relating to this publication event is identified and either conveyed or referenced in `AdvertisementPublicationInstance/ProofOfPublicationInformation`. All of these structures are imported from the AdsMLProofOfPublication standard, and also re-use structures from AdsMLBookings.

Note that regardless of whether or not `AppearanceInformation` is included in an invoice line, the invoicer may always include costs that are not directly associated with the publication of the advertisement. And in the case that the advertisement was never published, for example because of a cancellation, it is also possible to omit `AppearanceInformation` while invoicing for any charges that were incurred prior to or as a result of the cancellation.

An outline of the structured advertising information that can be included in a line item is shown below. In this outline an indented item is *contained in* its immediate parent, which is the first out-dented item above it, while items with the same level of indentation are *siblings* that share the same parent:

- Invoice or Credit Line (1…many)
    - Booking Identifier (0…1)
    - Advertisement Publication Instance (0…many)
        - Booking Information (0…1)
        - Appearance Information (0…1)
        - Proof of Publication Information (0…many)
            - Proof of Publication Identifier (0…1)
            - Proof of Publication Instance (e.g. tearsheet) (0…1)

The above outline conveys the following rules:

- A financial document must contain at least one invoice or credit line item (according to its type), but it can contain an unlimited number of them.
- A maximum of one Booking Identifier can be provided for a given invoice or credit line item. Therefore, if a Booking Identifier is provided, all of the activity described in the line item must have been triggered by that booking.
- An invoice or credit line item can contain an unlimited number of Advertisement Publication Instances. Each Advertisement Publication Instance describes in granular detail a single instance of the publication of an advertisement. Depending on the medium involved, this might correspond to a single "insertion", "appearance", "delivery", "broadcast" or "flighting" (etc.) of the ad in question.
- Each Advertisement Publication Instance can contain a set of Booking Information, which consists of details copied from the original booking. (See below for more details.)
- Each Advertisement Publication Instance can contain a set of Appearance Information, which describes when, where, how and to whom the ad was actually delivered.
- Each Advertisement Publication Instance can specify an unlimited number of Proofs of Publication. This supports scenarios in which more than one type of proof information must be delivered to the paying party.
- Each Proof of Publication contains an optional Identifier (and optional auxiliary identifiers) and an optional Proof. In the print world, the Proof is typically a tearsheet. In other media it might be omitted, or it might be a different type of physical item.

5.2.3.3.1 Usage rules and guidelines for booking and appearance information

In order to provide structured information in a line item about the publication of one or more advertisements, all of those advertisements **MUST** have been ordered in the same Booking.

Each instance of the publication of an advertisement is described by the inclusion of an `AdvertisementPublicationInstance` structure inside `Item.AdvertisementPublication`.

NOTE: the definition of an "instance" of publication will vary from one media to another. For example, in the print media, if a single invoice line describes the publication of an advertisement on both a Tuesday and a Thursday, then there will be two `AdvertisementPublicationInstance` elements inside the line item, one providing details for the ad that ran on Tuesday, and the other providing details for the ad that ran on Thursday. But in interactive media a line item is more likely to correspond to a time period such as a day, week or month, and that entire period might be described in a single `AdvertisementPublicationInstance`.

Each `AdvertisementPublicationInstance` can contain a copy of the `BookingInformation` associated with the publication of the advertisement, and also an optional `AppearanceInformation` structure that describes when, where and how the advertisement actually appeared.

5.2.3.3.1.1 Booking information

1. Trading partners wishing to convey structured booking information **SHOULD** populate the `BookingInformation` element as fully as reasonably possible.

   NOTE: In some cases the above guideline will result in multiple copies of the same `BookingInformation` being provided in a single line item. For example, if the ad that was published on a Tuesday and a Thursday was originally booked in a single Placement, then the `BookingInformation` in the two `AdvertisementPublicationInstance` elements will be identical, or nearly so. This de-normalization is deliberate, as it allows each copy of `AdvertisementPublicationInstance` to be as complete and self-contained as possible. On the other hand, if the appearances that are being invoiced in a single line item were booked in different Placements, then the contents of their `BookingInformation` elements may be quite different from each other.

2. Booking information, if provided, **MUST** be a copy of information from the booking that triggered the publication of this advertisement, and **SHOULD** consist of information that was previously made available to the buyer of advertising via an Ad Order Response (AD-OR) or Ad Order Status (AD-OS) message.

5.2.3.3.1.2 Appearance information

1. `AppearanceInformation`, if provided, **MUST** convey the invoice preparer's best understanding of when, where and how the ad actually appeared (or will appear).

2. If the invoice is prepared in advance of publication, any provided `AppearanceInformation` **SHOULD** contain only information that the publisher is highly confident will remain accurate when the advertisement is published. Information which might easily change during the publication process **MUST NOT** be explicitly included in `AppearanceInformation`.

3. When both `BookingInformation` and `AppearanceInformation` are included in an `AdvertisementPublicationInstance`, Trading partners should use their discretion when deciding how much of the `AppearanceInformation` structure to populate. It is possible to populate

only those elements of Appearance Information that are not clearly implied by the Booking Information. The benefit of this approach is reduced message size and complexity. The drawback is that when some or all of the Appearance information is omitted, it is difficult for the recipient to know whether that omission implies the ad "ran as booked" or whether the missing information was simply "unknown".

5.2.3.3.2 Usage rules and guidelines for proof of publication references and tearsheets

An `AdvertisementPublicationInstance` can contain explicit references to one or more external copies of the published ad (e.g. tearsheets), and/or can convey one or more digital copies of the ad (e-tearsheets) in-line within the line item. Information about each tearsheet is conveyed in an instance of the `ProofOfPublicationInformation` structure. The identifiers are carried in `ProofOfPublicationReference` and/or `AuxiliaryProofOfPublicationReferences`, while the tearsheet itself can be conveyed in `TearSheet` if desired.

The ability to convey an *in-line* tearsheet in an Invoice line item is meant to support scenarios in the classified advertisement workflow in which the tearsheet consists of a lightweight representation of the published text of the advertisement. However, it **MAY** be used for any digital tearsheet if the trading parties so agree.

The ability to convey or reference *multiple* tearsheets is provided in order to support scenarios in which two or more tearsheets relate to exactly the same instance of a published advertisement. There are two common causes for this:

1. Situations in which multiple pieces of artwork are associated with a single advertisement instance, for example, A/B splits in print ads.
2. Cases where multiple types of tearsheets are provided for the same advertisement instance, for example, a digital tearsheet of the ad in context (a single page) and also a voucher copy of the entire publication.

If more than one proof of publication reference or in-line tearsheet is included in an `AdvertisementPublicationInstance`, they **MUST** all relate to a single publication instance of a single advertisement to which a single set of fully populated Appearance metadata (including date, publication, production details and distribution pattern) would apply. They **MUST NOT** be used to convey proof of publication information about different advertisements, or varying appearances of the same advertisement, or advertisements which were distributed to different distribution targets.

Parties wishing to convey proof of publication references or in-line tearsheets relating to the publication of more than one appearance of an advertisement, or the publication of multiple advertisements, **MUST** split their information into multiple `AdvertisementPublicationInstances` so that each `AdvertisementPublicationInstance` relates to just one instance of a published advertisement.

Note: The above guidelines were written using language that applies primarily to print advertisements. However, the underlying concepts are meant to apply in a general sense to all media, even though the specifics may not be accurate in every case.

5.3 Pricing and taxes

In order to accommodate the wide range of possible invoicing and pricing scenarios, AdsMLFinancials provides an extremely rich set of payment and pricing structures. However, most of these structures are optional. While it is possible to present a great deal of structured pricing information, it is also possible for an invoice line item to consist of only an identifier, a textual description and a total amount. (And even the textual description is optional.) Invoicing parties should customize their use of AdsMLFinancials to reflect the level of detail that they wish to convey.

5.3.1 Relationship of price structures to those used in AdsMLBookings

Although users of AdsMLFinancials are not required to use AdsMLBookings, those who do will find many of the same pricing structures in both of them, although with different names. For example, the structure by which a type of pricing information is broken down into a `TotalPrice` amount, an optional `DescriptionLine`, and an optional set of `PriceComponents` is used extensively in both standards to convey a "stack" of pricing information of a given type. In booking messages this structure is used for Total Booking Price, Placement Group Price and Placement Price, while in invoices and credits it is used for Calculated Price and Additional Allowance Charges at the line item level, and for Additional Allowance Charges in the document footer.

The re-use of pricing structures across various AdsML standards is meant to make it easier for developers to implement them in software: once a developer has written code to populate one of them, it is a straightforward task to reuse that code for the other instances of the same type of structure.

This structural similarity can be deceptive, however, because the way that pricing information is meant to be conveyed in AdsMLFinancials is quite different from the equivalent cases in AdsMLBookings.

In AdsMLBookings, all of the pricing information for a given section of the booking, including charges, allowances, discounts and taxes, is meant to be conveyed in a single pricing structure in that section of the booking. In a Placement, for example, all of this information sits inside a single `PlacementPrice` structure. The breakdown of calculated prices, allowances and taxes is expressed in either the placement price description or the stack of sub-components that sits beneath it.

In AdsMLFinancials, however, each line item contains separate structures for the calculated price, allowances and taxes. The same information which, in AdsMLBookings, would be conveyed in `PlacementPrice` is allocated to three structures in an AdsMLFinancials line item.

This difference in approach and complexity reflects the fact that a booking message is primarily a document about the nature, timing and desired distribution for the publication of an advertisement, which only secondarily, and optionally, discusses its price. Most booking messages convey relatively simple pricing information, or in some cases, none at all. Only in the classified ad workflow, where many bookings have been pre-paid by credit card, and in a few specific countries, is it common to see pricing information in a booking that is complete to the level of detail that one would find on an invoice.

An AdsML financial document, conversely, is primarily about a financial transaction and only secondarily about the other booking-related information that it may contain. It is critical that a financial document be able to describe the financial transaction in sufficient detail to satisfy both the directly involved trading partners and other financial stakeholders such as auditors and tax authorities. The same pricing information which may have been relatively lightly presented in a booking message must be massaged into a different (and more detailed) format when presented on an invoice. The payment and pricing structures in AdsMLFinancials are designed to enable this.

5.3.1.1 Specifying pricing details

As mentioned above, details about the prices expressed in a financial document are conveyed using one or more `adsml:PriceComponents`. Usage rules and guidelines for `adsml:PriceComponent` can be found in the AdsML Type Library specification. Note however that the *`scheduleEntryReference`* attribute that is part of the `adsml:PriceComponent` **MUST NOT** be used in AdsMLFinancials.

5.3.2 Taxation Rules and Structures

In AdsMLFinancials, totals and subtotals for taxes can be expressed at both line and document levels. An overview of usage contexts is shown below, with reused structures formatted in bold:

- *Document level (header)*
  - *Financial parties*
    - `PartyTaxScheme`
      - **`TaxScheme`**
- *Line item level*
  - *Line item prices (charges and additional allowances)*
    - *Price Component*
      - **`TaxCategory`**
        - **`TaxScheme`**
  - `Tax` (adsml:`TaxTotalType`)
    - **`TaxSubTotal`**
      - **`TaxCategory`**
        - **`TaxScheme`**
- *Document level (footer)*
  - `AdditionalAllowanceCharge`
    - *Price Component*
      - **`TaxCategory`**
        - **`TaxScheme`**
  - `TaxTotal`
    - **`TaxSubTotal`**
      - **`TaxCategory`**
        - **`TaxScheme`**

The structures used in AdsML for taxation information are based on the corresponding UBL 2.0 structures, although sometimes simplified with elements removed. As is the case with UBL, AdsML does not provide structures for detailed tax reporting purposes. Instead, it provides structures to identify a tax regime and convey the information on which tax is based. These aim to be generic and are not based on any specific tax regime.

Further details including rules and guidelines for populating these structures can be found in *AdsML Type Library Specification*.

5.3.3 Positive vs. negative monetary amounts

5.3.3.1 Overview

In an AdsML Financial document, positive amounts reflect debits (amounts owed by the payer to the invoicer) and negative amounts reflect credits (reductions in the amount owed by the payer to the invoicer). This is the same regardless of the location of the amount in question: in the document header, in an invoice or credit line item, in an informational line, or in the document footer.

Therefore, most of the amounts in a typical invoice will be positive numbers (except for any allowances or discounts on that invoice), while most of the amounts in a typical credit will be negative numbers.

Note that this rule also applies to the `PrepaidPayment/PaidAmount` element in the document header, which records any pre-payments that have been received. Pre-payments reduce the amount owed by the payer to the invoicer, and therefore should be expressed as negative numbers just like any other credit in the document.

5.3.3.2 Usage rules and guidelines

Any monetary amounts in a financial document that increase the amount owed by the payer to the invoicer **MUST** be expressed as positive numbers.

Any monetary amounts in a financial document that decrease the amount owed by the payer to the invoicer **MUST** be expressed as negative numbers.

5.3.4 Line item pricing and taxes

5.3.4.1 Overview

At the line item level, a line item is required to contain a `LineExtensionAmount` (the total price for the goods and services represented by that line item excluding any taxes). In addition, the sender can optionally convey two distinct types of pricing information which explain how the line item total was derived:

- **Calculated price** – the price for the line item before applying additional surcharges/discounts and taxes. The calculated price is conveyed in the `CalculatedPrice` element.
- **Additional Allowances and Charges** – any surcharges or discounts that should be applied on top of the calculated price. Additional allowances and charges are conveyed in the `AdditionalAllowanceCharge` element.

For each of these two types of pricing information the sender can provide a `TotalPrice` element containing a mandatory `Amount` and an optional `DescriptionLine` (to convey a textual description explaining that total), an optional `PriceType` code (to categorize the total price in a machine processable way), and an optional set of `PriceComponents` (to convey machine-processable charges and/or allowances that comprise the total). The two Total Prices are then added together to generate the `LineExtensionAmount`, which is the total price for that line item as a whole, net of any taxes.

Any taxes that were applied as a result of calculated prices, charges or allowances expressed in the line item can be detailed in one or more instances of the `TaxTotal` structure. A separate `TaxTotal` structure is provided per tax type, i.e. per `TaxScheme`, and within each `TaxTotal` structure a separate

`TaxSubTotal` can be provided for each applicable `TaxCategory`. For example, if both regional and national taxes applied to the line item, two `TaxTotal` structures would be provided: one for the regional taxes and one for the national taxes. Further, if more than one type of regional taxes applied, they would be described as `TaxCategories` in two or more `TaxSubTotals` inside the regional `TaxTotal` structure.

See also the *AdsML Type Library Specification* for further details including rules and guidelines for populating taxation structures.

In addition to the `LineExtensionAmount`, an optional `LineItemCurrencyExtensionAmount` is provided that contains the value of the `LineExtensionAmount` expressed in the line item's locally declared currency. This element supports scenarios in which a different currency has been used for the line item than for the financial document as a whole. See the discussion of currencies and exchange rates, below, for more information.

5.3.4.2 Examples

An extremely simple line item contains just a line extension amount and a textual description:

```
<InvoiceLine>
 <ID>1</ID>
 <LineExtensionAmount>1150.00</LineExtensionAmount>
 <Item.AdvertisementPublication>
   <adsml:Description># NEWSPAPERS : 35</adsml:Description>
 </Item.AdvertisementPublication>
</InvoiceLine>
```

A slightly more sophisticated line item might divide the total into charges and allowances:

```
<InvoiceLine>
 <ID>1</ID>
 <CalculatedPrice>
  <adsml:TotalPrice>
   <adsml:Amount>1000.00</adsml:Amount>
   <adsml:DescriptionLine>Base Price</adsml:DescriptionLine>
  </adsml:TotalPrice >
</CalculatedPrice>
<AdditionalAllowanceCharge>
  <adsml:TotalPrice >
   <adsml:Amount>150.00</adsml:Amount>
   <adsml:DescriptionLine>Agency Commission</adsml:DescriptionLine>
  </adsml:TotalPrice >
 </AdditionalAllowanceCharge>
 <LineExtensionAmount>1150.00</LineExtensionAmount>
 <Item.AdvertisementPublication>
   <adsml:Description># NEWSPAPERS : 35</adsml:Description>
 </Item.AdvertisementPublication>
</InvoiceLine>
```

If taxes are applied at the line item level they can be explicitly expressed in the `Tax` element:

```
<InvoiceLine>
 <ID>1</ID>
 <CalculatedPrice>
  <adsml:TotalPrice>
   <adsml:Amount>1000.00</adsml:Amount>
  </adsml:TotalPrice>
</CalculatedPrice>
<AdditionalAllowanceCharge>
  <adsml:TotalPrice>
   <adsml:Amount>150.00</adsml:Amount>
   <adsml:DescriptionLine>Agency Commission</adsml:DescriptionLine>
  </adsml:TotalPrice>
 </AdditionalAllowanceCharge>
 <LineExtensionAmount>1150.00</LineExtensionAmount>
 <adsml:TaxTotal>
   <adsml:TaxAmount>195.50</adsml:TaxAmount>
    <adsml:TaxSubTotal>
     <adsml:TaxAmount>195.50</adsml:TaxAmount>
     <adsml:TaxCategory>
       <adsml:ID>
         <adsml:CodeValue>SR</adsml:CodeValue>
         <adsml:Description>Standard Rate</adsml:Description>
       </adsml:ID>
       <adsml:TaxScheme>
         <adsml:ID>
           <adsml:CodeList>TaxSchemeCodes</adsml:CodeList>
           <adsml:CodeValue>VAT</adsml:CodeValue>
           <adsml:Description>Value Added Tax</adsml:Description>
         </adsml:ID>
       </adsml:TaxScheme>
     </adsml:TaxCategory>
    </adsml:TaxSubTotal>
</adsml:TaxTotal>
</InvoiceLine>
```

Trading partners wishing to communicate the pricing basis in a machine-processable format can use Price Components to indicate exactly how the `CalculatedPrice` and `AdditionalAllowanceCharge` were derived. In this example only a `CalculatedPrice` is provided, but it contains a rich underlying structure which is also available for allowances:

```
<InvoiceLine>
 <ID>1</ID>
 <CalculatedPrice>
  <adsml:TotalPrice>
   <adsml:Amount>1100.00</adsml:Amount>
  </adsml:TotalPrice>
  <adsml:PriceComponent adsml:sequenceNo="1">
   <adsml:PriceComponentName>
     <adsml:CodeValue>Base Price</adsml:CodeValue>
    </adsml:PriceComponentName>
   <adsml:Amount>1000.00</adsml:Amount>
  </adsml:PriceComponent>
  <adsml:PriceComponent adsml:sequenceNo="2">
   <adsml:PriceComponentName>
     <adsml:CodeValue>Color Charge</adsml:CodeValue>
    </adsml:PriceComponentName>
   <adsml:Amount>100.00</adsml:Amount>
  </adsml:PriceComponent>
</CalculatedPrice>
 <LineExtensionAmount>1100.00</LineExtensionAmount>
</InvoiceLine>
```

A price component can include a `CalculationSpecification` to indicate how it was derived. This is especially useful when the component is based on a percentage, as in the following fragment:

```
<AdditionalAllowanceCharge>
 <adsml:TotalPrice>
  <adsml:Amount>-1328.00</adsml:Amount>
 </adsml:TotalPrice>
 <adsml:PriceComponent sequenceNo="1">
  <adsml:PriceComponentName>
   <adsml:CodeValue>Discount</adsml:CodeValue>
  </adsml:PriceComponentName>
  <adsml:Amount>-1328.00</adsml:Amount>
  <adsml:CalculationSpecification>
   <adsml:Percent>-0.5</adsml:Percent>
   <adsml:BasePrice>265602.00</adsml:BasePrice>
  </adsml:CalculationSpecification>
 </adsml:PriceComponent>
</AdditionalAllowanceCharge>
```

Each price component may also be associated with one or more `adsml:TaxCategory` elements that describe the taxes which, in the opinion of the invoicer, apply to that component.

5.3.4.3 Usage rules and guidelines

1. If a Calculated Price and/or Additional Allowances Charges are provided in a line item, then the `LineExtensionAmount` **MUST** be the sum of their Totals.
2. The Calculated Price (and the stack of price components beneath it) can contain internal surcharges or discounts. For example, the Calculated Price for a print ad might include a rate card (base) price, an additional color charge, and a production charge. These three together would add up to the Calculated Price. Depending on the desired level of detail, these could be conveyed either as formal `PriceComponents` of the Calculated Price, or simply described in its `DescriptionLine`.
3. The Calculated Price **SHOULD NOT** include any taxes.
4. The Additional Allowances Charge total (and the stack of price components beneath it) is intended to convey surcharges or discounts that are applied after (or "on top of") the Calculated price. The decision as to which type of price component should be included in the Calculated Price vs. the Allowances/Charges category will vary from one organization to another, and should be based on how the invoicing party wishes to present its information to its customers.
5. A given chargeable item or discount **MUST NOT** appear in multiple pricing structures in the same line item. For example, if a surcharge is included as a component of the Calculated Price, then it **MUST NOT** also be included in the Allowances section, and vice versa. (Note that this rule does not prevent the re-use of a pricing component *name* to represent more than one chargeable item or discount. For example, a line item may contain multiple instances of "Color Surcharge" price components, provided that each of them represents a different actual item or discount. In this case, it is possible that some of these "Color Surcharges" might appear in the Calculated Price stack and others might be treated as Additional Allowance Charges without violating the above rule.)
6. Additional Allowance Charge **SHOULD NOT** include any taxes.

7. The Tax Total price stack, if used, **SHOULD** contain all taxes that apply to this line item, including taxes based on its Calculated Price components and Additional Allowance Charge components. Tax details can be conveyed in the `adsml:TaxSubTotal` structure.
8. In each `adsml:TaxTotal` element, all instances of `adsml:TaxScheme` **MUST** be identical. This means that for every tax scheme where tax amounts are recorded, a separate instance of the `adsml:TaxTotal` element **MUST** be provided.

5.3.5 Document total pricing and taxes

5.3.5.1 Overview

The pricing information in the footer of an invoice or credit provides summary/total pricing for that document as a whole. The following types of summary pricing information are available:

- **Additional allowance charges** (`AdditionalAllowanceCharge`) – charges or discounts applying to the document as a whole, which therefore were not included in any of the line items themselves. Note that this does not include any taxes.
- **Tax total** (`adsml:TaxTotal`) – an optional breakdown of the taxes on the invoice, by type (i.e. Total VAT, Total State Tax, Total Federal Tax, etc.) This includes both taxes that were included in individual line items and any additional taxes that were added at the document level.
- **Line item extension total** (`LineExtensionAmount` inside `LegalMonetaryTotal`) – the sum of the line item extensions, therefore including any allowances and/or charges that were expressed at the line item level, but not taxes. Optional.
- **Total amount, excluding tax** (`TaxExclusiveAmount` inside `LegalMonetaryTotal`) – An optional sum of line extensions, plus any additional allowances and charges that were added at the document level, but not adjusted for any prepayments. I.e. this is the sum of the line item extensions plus document-level allowances and charges.
- **Total amount including tax** (`TaxInclusiveAmount` inside `LegalMonetaryTotal`) – The calculated total for this invoice as a whole, including all calculated charges, allowances, surcharges and taxes, but exclusive of rounding. Note that this amount **MUST NOT** be adjusted to reflect any pre-payments that may have been recorded in the document header.
- **Allowance total** (`AllowanceTotalAmount` inside `LegalMonetaryTotal`) – The total of all the allowances that are expressed in either a `CalculatedPrice` or `AdditionalAllowanceCharge` structure anywhere in the document, including both the line items and the document footer, where "allowance" is defined as any allowance, discount or price adjustment that benefits the Payer. In most cases this will be the total of all `PriceComponents` in a `CalculatedPrice` or `AdditionalAllowanceCharge` whose `Amount` is < 0, but it is also possible to include in the `AllowanceTotalAmount` a price reduction that was not expressed as a separate price component. For example, if a price of $100 shown in the invoice was reduced from a rate which is normally $125, it would be possible to include the amount of that reduction ($25) in the Allowance total.

- **Charges total** (`ChargeTotalAmount` inside `LegalMonetaryTotal`) – The total of all the charges that are expressed in either a `CalculatedPrice` or `AdditionalAllowanceCharge` structure anywhere in the document, including both the line items and the document footer, where "charge" is defined as any charge, surcharge or price adjustment that benefits the Invoicer. In most cases this will be the total of all `PriceComponents` in a `CalculatedPrice` or `AdditionalAllowanceCharge` whose `Amount` is > 0, but it is also possible to include in `ChargeTotalAmount` a charge whose full amount was not expressed as a separate price component. For example, if a price of $100 shown in the invoice was reduced from a rate which is normally $125, it would be possible to include the normal price ($125) in the Charges total.
- **Prepaid amount** (`PrepaidAmount` inside `LegalMonetaryTotal`) – The total of all prepayments that have been applied to this document. The details of each prepayment **SHOULD** be recorded as instances of `PrepaidPayment` in the document header, in which case `PrepaidAmount` is the sum of those prepayments.
- **Payable rounding amount** (`PayableRoundingAmount` inside `LegalMonetaryTotal`) – An adjustment applied by the invoicer at the document footer level to convert the calculated total price of the document (including any taxes and prepayments) into an acceptable payable amount. The `PayableRoundingAmount` is calculated as `PayableAmount` *minus* (`TaxInclusiveAmount` *plus* `PrepaidAmount`).
- **Payable amount** (`PayableAmount` inside `LegalMonetaryTotal`) – The mandatory total amount-to-pay of the financial document, including all charges, discounts and prepayments, and if appropriate, rounded to a suitable degree of precision.

5.3.5.2 Examples

The only required pricing information in a financial document is a `LineExtensionAmount` in each transactional line item and the `PayableAmount` in the footer:

```
<Invoice>
 ...
 <InvoiceLine>
  ...
  <LineExtensionAmount>6577.20</LineExtensionAmount>
 </InvoiceLine>
 <LegalMonetaryTotal>
  <PayableAmount>6577.20</PayableAmount >
 </LegalMonetaryTotal>
</Invoice>
```

If taxes are involved, it is possible to populate more of the `LegalMonetaryTotal` structure, for example as to show tax-exclusive and inclusive subtotals for the document:

```
<LegalMonetaryTotal>
 <LineExtensionAmount>6577.20</LineExtensionAmount>
 <TaxExclusiveAmount>6677.20</TaxExclusiveAmount>
 <TaxInclusiveAmount>6777.20</TaxInclusiveAmount>
 <PayableAmount>6577.20</PayableAmount >
</LegalMonetaryTotal>
```

The example above implies that a document level additional charge of `'100'` was added to reach a `TaxExclusiveAmount` of `'6677.20'` from the `LineExtensionAmount` of `'6577.20'`. In addition, the same value of `'100'` was also added as tax to reach the final `TaxInclusiveAmount` of `'6777.20'`. Apparently there were prepayments or rounding adjustments, because the `PayableAmount is` the same as the `TaxInclusiveAmount.`

If taxes are included anywhere in the document, they should be summarized in the `adsml:TaxTotal` structure in the footer. Also taxes that are added at the document level are expressed using this structure as in the following example:

```
<Invoice>
...
<adsml:TaxTotal>
  <adsml:TaxAmount>297.20</adsml:TaxAmount>
  <adsml:TaxSubTotal>
   <adsml:TaxableAmount>7430.00</adsml:TaxableAmount>
   <adsml:TaxAmount>297.20</adsml:TaxAmount>
   <adsml:TaxCategory>
     <adsml:ID>
       <adsml:CodeValue>StandardRate</adsml:CodeValue>
     </adsml:ID>
     <adsml:Percent>4</adsml:Percent>
     <adsml:TaxScheme>
       <adsml:ID>
         <adsml:CodeValue>StateTax</adsml:CodeValue>
       </adsml:ID>
     </adsml:TaxScheme>
   </adsml:TaxCategory>
  </adsml:TaxSubTotal>
 </adsml:TaxTotal>
 <LegalMonetaryTotal>
  <LineExtensionAmount>6280.00</LineExtensionAmount>
  <TaxExclusiveAmount>6280.00</TaxExclusiveAmount>
  <TaxInclusiveAmount>6577.20</TaxInclusiveAmount>
  <PayableAmount>6577.20</PayableAmount >
 </LegalMonetaryTotal>
</Invoice>
```

In the above example, the taxes of `'297.20'` are identified as being State taxes that were calculated as `'4'` percent of `'7,430.00'`. (Presumably the base charges upon which the taxes were applied can be found by examining the line items.)

See also the *AdsML Type Library Specification* for further details including rules and guidelines for populating taxation structures.

It is also common to add a surcharge or discount in the footer that applies to the document as a whole. This is done using the `AdditionalAllowanceCharge` element:

```
<Invoice>
 ...
<AdditionalAllowanceCharge>
  <adsml:TotalPrice>
   <adsml:Amount>-5.90</adsml:Amount>
  </adsml:TotalPrice>
  <adsml:PriceComponent adsml:sequenceNo="1">
   <adsml:PriceComponentName>
     <adsml:CodeValue>Agency Commission</adsml:CodeValue>
   </adsml:PriceComponentName>
   <adsml:Amount>-5.90</adsml:Amount>
  </adsml:PriceComponent>
 </AdditionalAllowanceCharge>
 <LegalMonetaryTotal>
  <LineExtensionAmount>45.90</LineExtensionAmount>
  <AllowanceTotalAmount>-5.90</AllowanceTotalAmount>
  <PayableAmount>40.00</PayableAmount>
 </LegalMonetaryTotal>
</Invoice>
```

5.3.5.3 Usage rules and guidelines

1. The `AdditionalAllowanceCharge` in the footer uses the same structure that is used to convey the line item pricing totals. This enables each additional charge or allowance to be described in as much detail as necessary.
2. Each instance of `adsml:TaxTotal` in a document footer **MUST** include all taxes that are governed by the same `adsml:TaxScheme` as that instance of `adsml:TaxTotal` and were expressed in any of the document's line items, and **MAY** include taxes having that `adsml:TaxScheme` that were not expressed anywhere else in the document (for example, taxes that were allocated as a result of an `AdditionalAllowanceCharge` in the document footer).
3. The `LineExtensionAmount` in a document footer consists of the sum of all of the `LineExtensionAmount` elements in the line items of that document.
4. The `TaxExclusiveAmount` in the footer consists of the sum of all of the `LineExtensionAmounts` in the line items of that document plus any `AdditionalAllowanceCharges` expressed in the footer. The `TaxExclusiveAmount` **MUST NOT** include any taxes.
5. The `TaxInclusiveAmount` in the footer consists of the `TaxExclusiveAmount` plus the sum of any taxes expressed in the document. It does <u>not</u> include any tax-related prepayments that may have been received for this document.
6. The `AllowanceTotalAmount` consists of the sum of all the allowances that are expressed in either a `CalculatedPrice` or `AdditionalAllowanceCharge` structure anywhere in the document, including both the line items and the document footer, where "allowance" is defined as any allowance, discount or price adjustment that benefits the Payer.
7. The `ChargeTotalAmount` consists of the sum of all the charges that are expressed in either a `CalculatedPrice` or `AdditionalAllowanceCharge` structure anywhere in the document, including both the line items and the document footer, where "charge" is defined as any charge, surcharge or price adjustment that benefits the Invoicer.

8. The `LineExtensionAmount`, `TaxExclusiveAmount`, `TaxInclusiveAmount`, `AllowanceTotalAmount` and `ChargeTotalAmount` **MUST NOT** be adjusted to reflect any pre-payments that have been received.
9. `PrepaidAmount` **MUST** contain the sum of all prepayments that have been received for this document, regardless of whether or not the details of those prepayments are recorded in the document. All prepayments that have been received **SHOULD** be recorded as instances of `PrepaidPayment` in the document header. If any `PrepaidPayment` elements are included in the financial document, then they **MUST** reflect all of the prepayments that have been received, and `PrepaidAmount` **MUST** therefore be the sum of those `PrepaidPayments`.
10. The `PayableRoundingAmount` **MUST** equal the result of the calculation: `PayableAmount` *minus* (`TaxInclusiveAmount` *plus* `PrepaidAmount`). If the `PayableRoundingAmount` is not zero it **MUST** be conveyed in the document.
11. The mandatory `PayableAmount` conveys the amount which the recipient of the financial document is expected to pay to or receive from the sender of the financial document as a result of the information contained in that document. It is calculated as the sum of `TaxInclusiveAmount`, `PrepaidAmount` and `PayableRoundingAmount`.

5.4 Payment terms

5.4.1 Overview

AdsMLFinancials provides three complementary mechanisms for expressing the payment terms associated with a financial document: as an unstructured note, as a combination of Due Date and Terms Code, and/or as a set of structures which precisely define the settlement period, a penalty period, and any discounts or surcharges associated with those periods. All of this information is conveyed in the `adsml:PaymentTerms` element in the document header.

A textual description of the terms can go in `PaymentTerms`/`Note`. This recreates the functionality of paper invoices.

The simplest way to convey the payment terms in a machine-processable form is by using the combination of `PaymentDueDate` and `PaymentTermsCode`. As their names suggest, these elements convey the date on which payment is due, and an agreed code or text string (such as "`Net15Days`" or "`PaymentDueUponReceiptOfInvoice`") that identifies the payment terms.

If desired, a structured description of the standard payment period, the first penalty period, and any discounts or penalties associated with these periods can also be conveyed in the elements that follow.

The `TermsReferenceCode` element records the event from which terms are offered for a length of time, identified by a standard code, e.g. "`InvoiceTransmissionDate`" or "`RunDate`" (from the AdsML Payment Terms Reference Event CV).

To specify that a discount will be provided for payments received during a specified period, populate `SettlementPeriod` with an appropriate combination of the `StartDateTime`, `EndDateTime`, and/or `DurationMeasure` of the period in question, and record the percentage that will be deducted from the invoice in `SettlementDiscountPercent`.

To specify that a penalty will be applied to payments received after a particular date or within a date range, populate `PenaltyPeriod` with the start date of the penalty period and either the end date of that period or its duration, and put the penalty percentage in `PenaltySurchargePercent`.

5.4.2 Examples

To indicate that a discount of 10% will be applied to payments received at least 15 days before the invoice’s due date of January 15 2008, you can populate just the discount percentage and the end date of the settlement period, omitting the other elements:

```
<adsml:PaymentTerms>
 <adsml:PaymentDueDate>2008-01-15</adsml:PaymentDueDate>
 <adsml:PaymentTermsCode>
    <adsml:CodeValue>FixedDateEarlyPaymentDiscountApplies</adsml:CodeValue>
 </adsml:PaymentTermsCode>
 <adsml:SettlementDiscountPercent>-10</adsml:SettlementDiscountPercent>
 <adsml:SettlementPeriod>
  <adsml:EndDateTime>2008-01-01</adsml:EndDateTime>
 </adsml:SettlementPeriod>
</adsml:PaymentTerms>
```

To indicate that a penalty of 1.5% per month will be applied to payments received after the due date:

```
<adsml:PaymentTerms>
 <adsml:Note>Late payments will be assessed a penalty of 1.5% per
month</adsml:Note>
 <adsmlPaymentDueDate>2008-01-15</adsmlPaymentDueDate>
 <adsmlPaymentTermsCode>
    <adsml:CodeValue>FixedDateLatePaymentPenaltyApplies</adsml:CodeValue>
 </adsmlPaymentTermsCode>
 <adsmlPenaltySurchargePercent>1.5</adsmlPenaltySurchargePercent>
 <adsmlPenaltyPeriod>
  <adsml:StartDateTime>2008-01-15</adsml:StartDateTime>
  <adsml:EndDateTime>2008-02-15</adsml:EndDateTime>
 </adsmlPenaltyPeriod>
</ dsml:PaymentTerms>
```

In the above example, because AdsMLFinancials only supports the structured description of the first penalty period, the `adsml:Note` element is used to convey the fact that additional penalties will apply after the first month. The Penalty Period is defined using start and end dates, but could equally well have been defined using a starting date and duration.

5.4.3 Usage rules and guidelines

5. It is **RECOMMENDED** that in Invoice documents both a `PaymentDueDate` and a `PaymentTermsCode` **SHOULD** always be provided.
6. Parties wishing to use the Penalty and/or Settlement structures **SHOULD** agree in advance on which sub-elements they will use and in which combinations.
7. It is possible to convey both a Settlement Discount and a Penalty Period, but if both are provided their date ranges **MUST NOT** overlap each other.

8. A `SettlementDiscountPercent`, if provided, **SHOULD** be expressed as a negative number.
9. A `PenaltySurchargePercent`, if provided, **SHOULD** be expressed as a positive number.

5.5 Payment Means and Prepaid Payments

5.5.1 Overview

AdsMLFinancials provides two header structures that can be used either separately or in conjunction with each other. These are `PaymentMeans` and `PrepaidPayment`.

`PrepaidPayment` describes an actual payment that has been received by the Invoicer and credited towards this document. The sum of all the `PrepaidPayments` that have been received is reflected in the document's `LegalMonetaryTotal/PayableAmount`.

Each `PrepaidPayment` can be assigned a business-significant ID, to contain, for example, the check number that was used to make that prepayment.

`PaymentMeans` describes an actual or desired method of payment. When linked to one or more instances of `PrepaidPayment`, it describes the actual method by which those prepayments were made. When used on its own, it describes the mechanism that the sender of the current financial document would like the document's Payer to use when paying the `PayableAmount` of the document. For example, it might provide details of a bank account into which the payments should be transferred.

When `PaymentMeans` information relates to a specific `PrepaidPayment`, the mechanism for indicating their relationship is to populate `PaymentMeans/PaymentID` with the same value as the `PrepaidPayment/ID` of the relevant prepayment.

5.5.2 Examples

This example shows a prepayment of $1,000 that was paid by check '123' and received on January 15:

```
<Invoice>
 ...
 <PaymentMeans>
   <PaymentMeansCode>
      <adsml:CodeValue>PayByCheck</adsml:CodeValue>
   </PaymentMeansCode>
   <PaymentID>123</PaymentID>
 </PaymentMeans>
 <PrepaidPayment>
   <ID>123</ID>
   <PaidAmount>1000.00</PaidAmount>
   <ReceivedDate>2008-01-15</ReceivedDate>
 </PrepaidPayment>
 <InvoiceLine>
 ...
</Invoice>
```

This example instructs the recipient to pay the invoice by wire transfer to a specified account at the First National Bank, for which an international SWIFT code is provided.

```
       <PaymentMeans>
            <PaymentMeansCode>
                <adsml:CodeValue>WireTransfer</adsml:CodeValue>
            </PaymentMeansCode>
          <PayeeFinancialAccount>
              <adsml:Identifier>
                  <adsml:IDLabel>Account Number</adsml:IDLabel>
                  <adsml:IDValue>1234567</adsml:IDValue>
              </adsml:Identifier>

            <FinancialInstitutionBranch>
              <adsml:Identifier>
                    <adsml:IDLabel>SWIFT Code</adsml:IDLabel>
                    <adsml:IDValue>CBUS002</adsml:IDValue>
              </adsml:Identifier>
             <adsml:Name>First National Bank</adsml:Name>
             <Address>
                    <adsml:City>Denver</adsml:City>
                    <adsml:StateProvince>Colorado</adsml:StateProvince>
                    <adsml:CountryName>USA</adsml:CountryName>
             </Address>
           </FinancialInstitutionBranch>
         </PayeeFinancialAccount>
        </PaymentMeans>
```

5.5.3 Usage rules and guidelines

1. Each prepayment that has been received towards this financial document **SHOULD** be recorded as an instance of `PrepaidPayment`. There is no limit to the number of `PrepaidPayments` that can be included in a financial document.
2. The sum of prepayments that have been received towards a financial document **MUST** be taken into account when calculating the document's `PayableAmount`. This is true regardless of whether or not the prepayments are described by instances of `PrepaidPayment`.
3. If the information in a `PaymentMeans` element describes the method by which a specific `PrepaidPayment` was received, then in those two structures `PaymentMeans/PaymentID` **MUST** contain the same value as `PrepaidPayment/ID`.
4. If the information in a PaymentMeans element is meant to describe the method by which the current financial document should be paid by its recipient, then `PaymentMeans/PaymentID` in that instance of `PaymentMeans` **MUST** be empty.
5. A financial document **MAY** contain more than one instance of `PaymentMeans` in which `PaymentID` is empty, in which case the recipient of the document should assume that they represent alternative acceptable payment methods.

5.6 Currencies and exchange rates

5.6.1 Currencies

5.6.1.1 Overview

By default, a single currency applies to all of the values in a financial document. This currency is identified in the document header. However, in order to support situations in which one or more of the line items in the document describe charges that were incurred in a different currency, AdsML provides the ability to identify a different currency on each line item.

The document header contains a mandatory Document Currency (`adsml:DocumentCurrencyCode`). The Document Currency identifies the currency that will be used for the transaction described by the document as a whole (in the case of an invoice, for example, the currency that should be used for payment of that invoice). By default this value also applies to all of the values in the line items.

Each line item contains an optional currency code (`adsml:CurrencyCode`). This value, if provided, applies to the `CalculatedPrice`, `AdditionalAllowanceCharge` and `LineItemCurrencyExtensionAmount` on that line item and overrides the Document Currency with respect to these structures.

The line item currency code does not apply to any tax information contained in `TaxTotal` structures in that line, or to the `LineExtensionAmount` for the line as a whole. These values are always expressed in the Document Currency: the same currency that is used for the document footer.

The allocation of a line item currency to just the prices and allowances, but not the taxes or line item extension total, reflects the fact that the taxes and totals on each line item are routinely summed-up into the document footer and therefore need to be expressed in the same currency as the footer. If a trading partner wishes to convey tax information in a line item using the local currency for that line item, those taxes should be placed in the `AdditionalAllowanceCharges` price stack, and should not be taken into consideration when calculating the tax-related totals in the document footer.

5.6.1.2 *Usage rules and guidelines*

1. A Document Currency **MUST** be provided in each financial document.
2. The Document Currency **MUST** apply to the amounts conveyed in the following structures:
    a. The `LegalMonetaryTotal` in the document footer, including all of its child elements
    b. The `AdditionalAllowanceCharges` and `adsml:TaxTotal` in the document footer, including their child elements.
    c. The `adsml:TaxTotal` structure in each and every line item in the document, including their child elements
    d. The `LineExtensionAmount` on each and every line item in the document.
3. If no currency code is explicitly expressed on a line item (i.e. it does not contain a populated `adsml:Currency` element), then the Document Currency **MUST** apply to the entire line, including any and all amounts conveyed in that line.
4. If a currency code is explicitly provided on a line item, then it **MUST** apply to the amounts conveyed in the following structures on that line (overriding the Document Currency Code with regard to these structures):
    a. The `CalculatedPrice`, including all of its child elements
    b. The `AdditionalAllowanceCharge`, including all of its child elements

   c. The `LineItemCurrencyExtensionAmount`

5. Because only one currency can be used for the prices on a given line item, if a document contains charges that need to be described in varying currencies, they **MUST** be conveyed in multiple line items.
6. If a trading partner wishes to convey tax information in a line item using a currency other than the Document Currency for the document as a whole, those taxes **SHOULD** be conveyed as price components in the `AdditionalAllowanceCharge` structure on that line using `PriceComponentNames` that clearly identify them as taxes, and **MUST NOT** be treated as taxes when calculating the `TaxTotal` and `LegalMonetaryTotal` amounts in the document footer.

5.6.2 Exchange rates

5.6.2.1 Overview

In some cases it is useful or necessary to identify the exchange rate that was used to arrive at a particular set of currency values in a financial document. AdsMLFinancials provides an optional exchange rate structure at each location where a currency code can be specified:

- A Document Currency Exchange Rate in the document header (`ExchangeRate`)
- A Line Item exchange rate in each line item (`InvoiceLine/ExchangeRate` or `CreditLine/ExchangeRate`).

Each exchange rate structure expresses the relationship between one of the currencies in use in the financial document and another currency from which its amount was derived or into which it can be converted. Each structure provides the following types of information, most of which are optional:

- Source currency code and base rate
- Target currency code and base rate
- Conversion date
- Calculation rate and operator (multiply or divide)
- Exchange market
- Contract reference

In each exchange rate structure either the source or target currency must correspond to the currency being used for the relevant type of information in the financial document. In the `ExchangeRate` structure in the document header, either the Source or Target currency must be the same as the document's `DocumentCurrencyCode`, while in a Line Item exchange rate structure, either the Source or Target currency must be the same as the currency code for that line item and the other currency must be the same as the currency that applies to the document as a whole.

The exchange rate structures should only be used to support currencies that are explicitly specified in the document:

- If only a Document Currency is specified, then only a document header Exchange Rate may be given in the document. This would show the relationship between the Document Currency and one other currency.

- If a Line Item currency is specified then an `ExchangeRate` may be given for that line item in order to show the relationship between the line item's currency and the document's Transaction Currency.

The document level exchange rate is particularly useful in cases where all the line items in the document are in a single currency that is different from the document's currency, and they were all calculated using the same exchange rate. In this case the document exchange rate can be used to show the relationship between the document currency and the currency used in all the line items.

The exchange rate on a line item, if provided, applies only to that line item. This is meant to support situations in which the line items in a document use varying currencies or were calculated on different dates. In this case each line item can contain the conversion information which applies to that line item.

5.6.2.2 Usage rules and guidelines

1. Exchange rates may only be defined for types of currencies that are explicitly identified in the document. Therefore:
    a. A Document Exchange Rate (`ExchangeRate`) **MAY** always be provided.
    b. If a Line Item currency code (`adsml:CurrencyCode`) is specified on a given line item, then a line item Exchange Rate (`ExchangeRate`) **MAY** be provided on that line item. However, if a line item does not contain a currency code, then that line **MUST NOT** contain an Exchange Rate structure.
2. In the document's `ExchangeRate` structure either the Source or Target currency **MUST** be the same as the document's `adsml:DocumentCurrencyCode`.
3. In a Line Item exchange rate structure, either the Source or Target currency **MUST** be the same as the currency used for that line item, and the other currency **MUST** be the same as the document's `adsml:DocumentCurrencyCode`.
4. If a Line Item exchange rate is provided, it applies only to the financial information on that line item.
5. Within an exchange rate structure at least one of the following **MUST** be provided: a `CalculationRate` or a currency conversion `Contract`.
6. When a `CalculationRate` is provided, then the formula being expressed is: "*SourceCurrencyBaseRate* units of the *SourceCurrency =* (*TargetCurrencyBaseRate* [multiplied or divided by] *CalculationRate)* units of the *TargetCurrency"*.
    a. The sequence of operation **MUST** always be *from* the source currency *to* the target currency.
    b. If no operation is specified, the operation **MUST** be assumed to be "Multiply".
    c. If no `SourceCurrencyBaseRate` is provided, its value **MUST** be assumed to be 1.
    d. If no `TargetCurrencyBaseRate` is provided, its value **MUST** be assumed to be 1.

    For example:

a. If *SourceCurrency* = "EUR", *TargetCurrency* = "USD", *CalculationRate* = "1.25" and the other values are omitted, then the exchange being expressed is "**1 Euro = (1 * 1.25) US Dollars**".

b. If *SourceCurrency* = "TRL", *TargetCurrency* = "USD", and *CalculationRate* = "0.000000716", then the exchange being expressed is "**1 Turkish Lira = 0.000000716 US Dollars**". Because many systems cannot handle such minute fractions, it is common to express conversions like this using a different base rate. See the next example.

c. If *SourceCurrency* = "TRL", *TargetCurrency* = "USD", SourceCurrencyBaseRate = "1000000" and *CalculationRate* = "0.716", then the exchange being expressed is "**1,000,000 Turkish Lira = 0.716 US Dollars**". This is effectively identical to the previous example.

5.6.2.3 Examples

The simplest format for expressing an exchange rate is to provide the mandatory source and target currency codes and the calculation rate. In the example below, one Euro is shown to be valued at 1.25 US Dollars:

```
<ExchangeRate>
 <SourceCurrencyCode>EUR</SourceCurrencyCode>
 <TargetCurrencyCode>USD</TargetCurrencyCode>
 <CalculationRate>1.25</CalculationRate>
</ExchangeRate>
```

Here is the same information, but this time all of the details are explicitly provided:

```
<ExchangeRate>
 <SourceCurrencyCode>EUR</SourceCurrencyCode>
 <SourceCurrencyBaseRate>1</SourceCurrencyBaseRate>
 <TargetCurrencyCode>USD</TargetCurrencyCode>
 <TargetCurrencyBaseRate>1</TargetCurrencyBaseRate>
 <CalculationRate>1.25</CalculationRate>
 <OperatorCode>Multiply</OperatorCode>
</ExchangeRate>
```

It is usually desirable to provide a date on which the exchange rate was effective, and when specifying a currency for a line item, to show the subtotal for that line in both the source and target currency. For example, if the document's transaction currency is USD and a line item's pre-tax total is 1,500 Euros, then the currency-related information in the line might look like this:

```
<InvoiceLine>
 <ID>1</ID>
 <adsml:CurrencyCode>EUR</adsml:CurrencyCode>
 <ExchangeRate>
  <SourceCurrencyCode>EUR</SourceCurrencyCode>
  <TargetCurrencyCode>USD</TargetCurrencyCode>
  <CalculationRate>1.25</CalculationRate>
  <adsml:Date>2007-01-01</adsml:Date>
 </ExchangeRate>
 <CalculatedPrice>
  <adsml:TotalPrice>
    <adsml:Amount>1500.00</adsml:Amount>
  </adsml:TotalPrice>
 </CalculatedPrice>
 <LineItemCurrencyExtensionAmount>1500.00</LineItemCurrencyExtensionAmount>
 <LineExtensionAmount>1875.00</LineExtensionAmount>
 <adsml:TaxTotal>
  <adsml:TaxAmount>150.00</adsml:TaxAmount>
 </adsml:TaxTotal>
</InvoiceLine>
```

In this example the `LineItemCurrencyExtensionAmount` contains the pre-tax subtotal expressed in the currency assigned to the line (in this case, Euros), while the `LineExtensionAmount` contains the same value converted to the currency that is used for the document as a whole (in this case, USD) using the exchange rate that applies to the line (in this case, 1.25).

5.7 The use of Response and Status messages

As noted above, AdsMLFinancials supports a simple delivery model for both invoices and credits: the invoicing party sends an invoice or credit, and if request-response choreography is used, the recipient replies with either an Invoice Response (FD-NVR) or Credit Response (FD-CRR) message.

Once an invoice has received further processing its status may change. In this case the document recipient may optionally send an Invoice Status (FD-NVS) message to convey the new status. At any time during the processing of an invoice, the original sender may follow up with a status enquiry message (FD-NVSE) to request information about its current status.

(If datagram choreography is used, the recipient sends only an administrative response and does not send Response or Status messages.)

The information that can be contained in a Response message (FD-NVR or FD-CRR) is extremely limited. Basically, the response message can either indicate that the entire financial document was “denied”, indicating that it could not be successfully loaded into the recipient’s system, or it can provide a single status code that applies to the entire financial document. This status code can be used for any purpose that is agreed between the trading partners. However, an Invoice or Credit Response message **SHOULD** be used only to convey the recipient’s “first glance” response to the invoice or credit note (for example, “Invoice received and being processed”), and **SHOULD** be sent within a short time following receipt of the Invoice or Credit.

The information that can be contained in an Invoice Status message (FD-NVS) is essentially identical to the information in a Response message. The primary difference is one of timing. While a Response message conveys the recipient’s <u>initial</u> response to the invoice, the Status message is normally used to transmit <u>new or updated</u> information about the status of that document. There are no restrictions on the timing of when a Status message should be sent, nor are there limits on how many Status messages may be sent regarding the same initial document.

A secondary difference between Status and Response messages is that a Response message can be sent only to the party which submitted the original financial document, while a Status message can go to as many other parties as the sender wishes to notify.

Status information in either a Response or Status message is conveyed in the optional `adsml:Status` element, which contains a code and an optional, repeatable `adsml:StatusQualifier` code that can provide more information about the status. Please see the *E-Commerce Usage Rules & Guidelines* for a general discussion about use of status values.

The status information about a financial document applies to the document as a whole. In particular, business statuses which would reference specific line items, such as "We agree to pay everything except for line item number 5" or "Item 2 has the wrong price" **SHOULD NOT** be conveyed in an Invoice Response or Status or Credit Response message. The *Advertising Components Interactions Analysis* defines another set of financial messages, "Claims" and "Responses to Claims", that will be able to convey such information once they have been developed. However, these messages are not supported in the current release of AdsMLFinancials.

5.8 Usage and Definitions of Controlled Vocabularies

AdsMLFinancials enables trading partners to use controlled vocabularies (CVs), i.e. defined lists of values, for many element values. In most cases, CVs recommended by the AdsML Consortium are available in the AdsML Controlled Vocabularies schema, imported into the AdsMLFinancials schema. In any case, trading partners may use any agreed value, either directly without schema-based validation, or as schema-defined CVs located in a user extension schema. Please see the *E-Commerce Usage Rules & Guidelines* for a general discussion about use of CVs.

6 Use Cases and Recommended Solutions

This section attempts to address, in a lightweight way, how to use the AdsML Financials standard in many common situations. It assumes a basic familiarity with the financials format, and with the more detailed choreography and usage information provided above.

6.1 Invoices

6.1.1 Invoice for publication of one advertisement with full structured details

Scenario: Invoicer sends invoice for publication of a newspaper advertisement to a party that it expects to pay for the publication. The paying party is not necessarily the buying party that initiated the booking. The invoice identifies the booking and provides details about the actual publication of the advertisement; this information helps the payer to match the invoice with the order that initiated it, and to determine whether to approve payment of the invoice.

Optionally, the invoice may also reference a suitable proof of publication which has been provided by the invoicer or by the publisher.

AdsML handling: The sender creates an invoice with a single line item, using `Item.AdvertisementPublication` to describe the details about the line item (because it relates to a published advertisement) and `AdvertisementPublicationInstance` to describe this specific instance of publication of the ad.

The anticipated payer party is identified in the `PayerParty` element in the document header, while the party that originally booked the advertisement is identified in `BookingParty` inside `Item.AdvertisementPublication.`

The booking is referenced by populating `adsml-bo:BookingReference` (if an AdsML QID is available) and `AuxiliaryBookingReferences`. If the buyer provided a purchase order number, it is carried in `adsml:PurchaseOrderReference` at the line item level, and the applicable contract can be identified in `adsml:Contract`, also at the line item level.

The specific placement is identified using `PlacementReference` and `AuxiliaryPlacementReferences` in `BookingInformation.NewspaperMagazine` in the `AdvertisementPublicationInstance` structure. If the Buyer's Placement IDs are known, they should be placed in `AuxiliaryPlacementReferences`. Additional details from the booking can also be conveyed here if desired.

Structured details about the ad as it *actually* appeared (e.g. publication name, dates, regions, editions, size, color, etc.) are placed in `AppearanceInformation.NewspaperMagazine` inside `AdvertisementPublicationInstance`. This information is conveyed using the same or similar elements to those found in AdsMLBookings, so they can be populated in the same way.

If a tearsheet for this ad has been sent to the payer it can be identified using `ProofOfPublicationReference` and

`AuxiliaryProofOfPublicationReferences` in `ProofOfPublicationInformation`.

The price for publishing the ad can be broken out into separate components (e.g. rate card price, color charge, production charge, etc.) by creating a stack of `CalculatedPrice/adsml:PriceComponents`. Alternatively, the pricing breakdown can be described as a text string in `CalculatedPrice/adsml:TotalPrice/adsml:DescriptionLine`. In either case the total price (before additional allowances and taxes) goes in `CalculatedPrice/adsml:TotalPrice/adsml:Amount`.

Any additional allowances or charges (but not taxes) should be conveyed in `AdditionalAllowanceCharge`. The sum of `CalculatedPrice` and `AdditionalAllowanceCharge` is then copied to `LineExtensionAmount`, which represents the total price for the line item, excluding taxes.

Any taxes that apply to this line item should be conveyed in the `TaxTotal` structure on the line item.

Because there is only one line item in this invoice, the `LineExtensionAmount` in the line item is the same as the optional `LegalMonetaryTotal/LineExtensionAmount` in the document footer. If any taxes apply, the mandatory `LegalMonetaryTotal/PayableAmount` in the document footer should be adjusted to include them.

Notes:

- It is possible to break out taxes separately in the invoice footer, and to add additional allowances to the line item total, by populating the other pricing-related fields there.
- `adsml:PayerParty/adsml:Name` should always contain the name of the legal entity that is responsible for making the payment. In cases where the invoice is sent to a third party other than the Payer Party, it is possible to format an address that contains both the recipient party and the Payer Party by placing the recipient party's name in the Department or Street 1 element as appropriate. When formatted for human display this yields an address such as:

  Payer Party Name

  Invoice Recipient Name

  Recipient's street

  Recipient's city, etc.

6.1.2 Invoice for publication of one advertisement with textual details only

Scenario: Invoicer sends invoice for publication of an advertisement to a party that it expects to pay for the publication. All of the information about the advertisement is conveyed in the simplest possible manner, using primarily text strings.

AdsML handling: The sender creates an invoice with a single line item, using `Item.AdvertisementPublication` to contain the details about the line item because it relates to the publication of an advertisement.

The full description of the published advertisement is conveyed as a text string in `InvoiceLine/Item.AdvertismentPublication/adsml:Description`. Information such as the description of the advertisement and references to its booking and placement numbers is contained in-line in the description. The description may contain embedded carriage returns so that it matches the layout of the invoicer's printed invoices.

In a truly minimal implementation, the price for the ad is carried in `LineExtensionAmount` and no other line item pricing elements are used. If a slightly more detailed pricing breakdown is required, the sender can populate `CalculatedPrice`, `AdditionalAllowanceCharge` and/or `adsml:TaxTotal`.

6.1.3 Invoice for publication of multiple placements with separate prices

Scenario: Invoicer sends invoice for publication of multiple advertisements, which may have been booked either as multiple placements in a single booking, or on separate bookings. The invoice identifies a separate price for each specific instance of publication.

AdsML handling: Sender creates an invoice containing multiple line items, one for each instance of the publication of an advertisement that has its own price. (Generally, this will equate to one for each placement, but sometimes it will equate to a lower level of granularity such as one line item for each schedule entry structure or appearance date.) Each line item can then contain structured references to the placements, tearsheets and booking and appearance details for that instance of the published advertisement.

Note: It is also possible to invoice for multiple placements in a single line item. However, only one set of pricing information will be provided per line item.

6.1.4 Invoice for publication of multiple ad instances with a single price

Scenario: Invoicer invoices for publication of multiple advertisements (or multiple insertions/appearances of the same advertisement) but needs to group them together with a single "package" price. This can occur either when a single placement specifies multiple dates, or when a group of placements are purchased together at a package price. Despite providing only one price, Invoicer wishes to provide structured details about each individual appearance, in order to facilitate order reconciliation and approval.

AdsML handling: Sender creates an invoice containing a single line item representing the total buy. All of the pricing information is expressed in this line item, and the description is used to describe the overall package.

Metadata about each instance of publication of an ad is provided in an `AdvertisementPublicationInstance` structure inside the line item. The definition of "instance" will vary according to the medium in question. For example, if the line item price relates to five ad appearances, there will be five `AdvertisementPublicationInstances` inside the line item. But if the line item relates to five months of an interactive campaign that was booked using five `Scheduling` structures, there could be five `AdvertisementPublicationInstances`, one for each of those months.

Note: This approach assumes that all of the ad appearances referenced by the line item were ordered in a single booking. If they were ordered in different bookings, then either they must be invoiced on multiple line items (one for each booking), or they must be invoiced using just the "price plus textual description" approach which omits any structured Booking and Appearance metadata about them.

6.1.5 Invoice for a campaign buy (no reference to a booking)

Scenario: Invoicer sends invoice for a campaign buy to the party that agreed to pay for the campaign. The invoice does not reference the booking or publication of any particular advertisement.

AdsML handling: The sender creates an invoice containing a line item for the campaign buy. Because it does not refer to a specific advertisement, the sender populates `Item.Generic` rather than `Item.AdvertisementPublication`, and uses `adsml:Name`, `adsml:Description` and `adsml:Specifications` to describe what is being invoiced.

Note: The sender can also use `adsml:PurchaseOrderReference` and `adsml:Contract` to provide additional information.

6.1.6 Invoice for a month of published advertisements

Scenario: Invoicer sends invoice for publication of multiple advertisement placements to a party that it expects to pay for their publication. The advertisements listed on the invoice may have been ordered in multiple bookings, and the paying party is not necessarily the buying party that initiated those bookings.

Optionally, for each entry on the invoice, the invoice may reference a suitable proof of publication which has been provided by the invoicer or by the publisher.

AdsML handling: The sender creates an invoice containing multiple line items, one for each instance of the publication of an advertisement during the month in question.

The first and last days of the month are specified in `InvoicePeriod/adsml:StartDateTime` and `InvoicePeriod/adsml:EndDateTime`. Line items are described using `Item.AdvertisementPublication`. A single `AdvertisementPublicationInstance` is provided for each line item, containing structured references to the placements, tearsheets and booking and appearance details for that instance of a published advertisement.

All proof of publication information is conveyed in `ProofOfPublicationInformation`. The tearsheet ID goes in `ProofOfPublicationInformation/adsml-pp:ProofOfPublicationReference`.

Notes:

- It is possible to combine multiple advertisement publication instances into a single line item by including multiple `AdvertisementPublicationInstance` elements. However, this practice is only recommended when multiple advertisement insertions/appearances need to be associated with a single price.
- Additional information about the tearsheet can be conveyed in `adsml-pp:TearSheet`.

6.1.7 Invoice for a month of activity including claim resolutions

Scenario: Invoicer sends invoice to a paying party for a month of activity, including both publication of multiple advertisements during the period and resolution of claims or queries relating to previous invoices.

AdsML handling: The sender creates an invoice containing multiple line items. Each newly published advertisement is represented by an instance of `InvoiceLine`, as usual.

Each claim resolution that resulted in the issuance of a credit is represented by an instance of `CreditLine`. The reason for the credit is conveyed in `CreditLine/CreditReason`, and the specific invoice that was queried is identified in `RelatedInvoice/InvoiceReference`. If the claim was against one or more specific line items on that invoice, they can be identified in `RelatedInvoice/LineItemReference`.

If the resolution of a claim generated not a credit but rather the issuance of a revised or "adjusted" invoice, this should be handled by generating a replacement invoice or, in the current scenario, one or more line items on the monthly invoice which represent the replacement charge(s). Each such line item should be prepared exactly like a line on the initial invoice, with the exception that `RelatedInvoice` is populated with information about the invoice (or invoice plus line item) that it replaces. Use `InvoiceReference` and `LineItemReference` to identify the original invoice, and either `adsml:RelationshipName` or `adsml:Description` to indicate that this is an "adjusted replacement" (or whatever term is more appropriate) for the prior invoice. Use `RelationshipName` if you and your trading partners have agreed to use predefined terms from a controlled vocabulary to describe the relationship between two invoices, or use the `Description` if you wish to describe the nature of this replacement using free text.

Notes:

- The invoice message is not a substitute for a message which would fully describe how the claim was resolved. For example, there is no way in an invoice message to indicate that a query or claim was resolved except by issuance of a credit or replacement invoice. The AdsML Technical Working Group expects to define a set of Query/Claim messages in the future, but as of this writing has not yet done so.
- Credit amounts should be conveyed as negative numbers.

6.1.8 Send a statement

Scenario: Invoicer wishes to send a statement that summarizes all recent activity on a Payer's account and provides an aged list of receivables due from that payer.

AdsML handling:

AdsMLFinancials does not support sending machine-processable financial statements. However, it is possible to use multiple instances of the `InformationalLine` structure in an Invoice to transmit all of the information that would appear on a statement in a human-readable format.

The ability to use `InformationalLines` to transmit statement-like information is provided in order to support current workflows in which an invoice document contains one or more invoice line items and <u>also</u> some statement information.

Sending an Invoice or Credit message that only contains statement information would be a non-standard use of the AdsML message that is strongly discouraged.

Notes: Although statements are common in paper workflows, it is the consensus of the financial experts consulted by the AdsML Technical Working Group that they are largely irrelevant in a system-to-system e-commerce workflow. The secure, validated exchange of invoice and credit line items will ensure that all of the relevant information is in both parties' systems. Either party can then run a report from their own system showing activity during the month and aged balances.

6.1.9 Include aging activity in an invoice

Scenario: Invoicer wishes to include in each invoice an analysis of the aged debt owed by the payer to the invoicer. (This is common practice in current American newspaper invoices.)

AdsML handling: There are two ways to transmit aging information (or any other information which is not part of the current transaction) in an AdsML Invoice document.

The first approach is to use the `InformationalLine` structure. Senders can populate multiple instances of `InformationalLine` with the values from the aging activity summary: one informational line for the 30-day total; one line for the 60 day total; etc. Use the `adsml:Type` code on each informational line to identify the type of information in that line. Because the components of the aging information are in structured, identifiable locations, it is possible to apply a stylesheet to the document to format this information as desired.

The second approach is to use the `adsml:Note` element in the financial document header. This element can contain a string of any length, including embedded carriage returns, so it is possible to populate this element with an aging debit analysis if so desired. The information, however, can only be formatted to the extent that it is possible to format the raw text in a note field.

Notes: Although statements are common in paper workflows, it is the consensus of the financial experts consulted by the AdsML Technical Working Group that they are largely irrelevant in a system-to-system e-commerce workflow. The secure, validated exchange of invoice and credit line items will ensure that all of the relevant information is in both parties' systems. Either party can then run a report from their own system showing activity during the month and aged balances.

6.1.10 List recently received payments on an invoice or credit

Scenario: Invoicer wishes to include at the start of an invoice one or more line items listing payments that have recently been received from that payer party. (This is common practice in some American newspaper invoices.)

AdsML handling: Information that is not part of the current transaction, such as payments received in the context of an invoice, should be conveyed using the `InformationalLine` structure. This can easily be displayed to a human being but will be ignored when the data contents of the invoice are imported into the payer party's accounting system. In order to force the Informational Line to sort ahead of the Invoice line items, assign it a lower `ID` value and also position it ahead of the line items in the XML message.

Notes: In an Invoice message, information about a recently received payment is merely an annotation to the transaction and should always be sent as an `InformationalLine`. In a Credit message, however, it is up to the sender to decide whether the record of having received the payment is a material part of the transaction (in which case it should be transmitted as a `CreditLine`) or merely an annotation (in which case it should be transmitted in an `InformationalLine`).

6.1.11 Identify payments received or balance carried forward

Scenario: Invoicer wishes to include at the start of the invoice the current status of the payer's account, by providing the balance carried forward from previous invoices and/or the most recent payment received. (This is common practice in many American newspaper invoices.)

AdsML handling: Use the `InformationalLine` structure to transmit any information that should be printed as line items on the invoice but is not part of the current e-commerce transaction. The line item Identifier can be used as a sequence number, allowing each Informational line to be positioned appropriately within the sequence of Invoice and/or Credit lines in the document.

Note: An Informational line can also be used for subtotals or section breaks within the body of the invoice.

6.1.12 Invoice multiple payers for a single advertisement

Scenario: Invoicer sends invoices to multiple parties that relate to a single advertisement, for example, in a co-op advertising scenario.

AdsML handling: An AdsML Invoice can be sent to one and only one paying party. In co-op advertising scenarios, a separate invoice must be sent to each of the expected payers, and each invoice should only include the pricing information that is relevant to that payer. However, the invoices can all share the same `Item.AdvertisementPublication` information to describe the published advertisement.

6.1.13 Send a replacement invoice

Scenario: After receiving an invoice, Payer informs the Invoicer that they have lost the invoice and would like the Invoicer to send another copy.

AdsML handling: In order to preserve the integrity of the e-commerce message exchange, AdsML does not support re-sending an invoice message after the initial Administrative Response from the Payer has indicated that it was successfully received. However, AdsML does support sending a replacement invoice which references the first one. This is done by generating a new invoice (with a new invoice identifier) that contains all of the same business information as the first one, and uses the `RelatedInvoice` structure to indicate that the second invoice is an identical replacement for the first one. In this case, a `RelationshipName` such as "IdenticalReplacement" should be specified.

Notes: Many accounting systems will be unable to generate a replacement invoice as described here. Trading partners may find alternative ways of providing a replacement invoice, such as by fax or email.

6.1.14 Send duplicate copies of an invoice

Scenario: Invoicer wishes to send cc: copies of an invoice to one or more parties in addition to the Payer.

AdsML handling: As of this writing, AdsML does not support sending duplicate copies of an invoice via e-commerce messaging. Only the primary, actionable copy of the invoice may be transmitted in an AdsML message.

Note: Trading partners may find alternative ways of transmitting the duplicate invoices, such as by fax or email.

6.1.15 Send an adjusted invoice

Scenario: Invoicer wishes to send an adjusted or revised copy of an invoice.

AdsML handling: AdsML does not support sending an adjusted or revised copy of an invoice. The Invoicer should generate a second invoice (with a new invoice identifier) for the correct amount, and use the `RelatedInvoice` structure to explicitly indicate that the second invoice replaces the first one.

Note: It may be preferable to leave the original invoice in effect and make the adjustment via either a Credit or an additional invoice. A reduction in price can be accomplished by sending a Credit. A price increase or additional charge can be conveyed in an additional invoice. In both cases, the appropriate line item on the credit or additional invoice can include a `RelatedInvoice` structure that references the original invoice.

6.1.16 Provide information about a print ad that ran on multiple dates

Scenario: Invoicer wishes to bill for the publication of a newspaper ad that appeared on two different dates but was booked in a single Placement and therefore has a single price.

AdsML handling: Because the two appearances will be billed for a single price, the ad should be invoiced using one line item containing two `AdvertisementPublicationInstance` elements, one for each appearance of the ad. In each `AdvertisementPublicationInstance`, populate the `BookingInformation` as fully as possible.

Assuming that the ads were published as booked, then most of the appearance information can be inferred from the detailed booking information. The only significant information that needs to be provided in `adsml-pp:AppearanceInformation.NewspaperMagazine` is that which differs from the Booking information: when (which date) and where (which page) each ad actually ran. So populate the `adsml-pp:Appearance` and `adsml-pp:ProductionDetail.NewspaperMagazine/Positioning` elements with these values, and omit the rest of the `AppearanceInformation`.

Note:

Because the ad was booked in a single placement, the instances of `BookingInformation` in the two `AdvertisementPublicationInstance` elements will be identical. This is a deliberate redundancy that is intended to make each `AdvertisementPublicationInstance` substantially complete and self-contained.

6.1.17 Include information about the performance of an ad

Scenario: Invoicer is invoicing for the publication of one or more interactive or broadcast ads and wishes to include information about how many impressions of the ad were delivered or how many people saw it.

AdsML handling: Invoicer generates an invoice line item for the published ad, using `Item.AdvertisementPublication`. Information about when, where, how and to whom the ad was published goes in `adsml-pp:AppearanceInformation`. In particular, `PlacementResult` is used to convey the overall achieved results (e.g. the total number of impressions or appearances), and if more specific information is required about the distribution of the ad (for example the count of people who actually viewed it), it can be conveyed in `DistributionResult`. Information about the provenance of all of this performance-related information can be conveyed in the `Provenance` structure; if a third party service provided the information, that party can be identified in `Provenance/adsml:ProvenanceParty`.

6.1.18 Include a tearsheet in-line

Scenario: Invoicer is invoicing for the publication of one or more classified ads and wishes to include a digital copy of each published ad (e-tearsheet) inside the invoice message.

AdsML handling: Invoicer generates a separate invoice line item for each published ad, using `Item.AdvertisementPublication`. An identifier for the tearsheet is conveyed in `ProofOfPublicationInformation/adsml-pp:ProofOfPublicationReference`, while supporting information about when, where and how the ad was published can go in `adsml-pp:AppearanceInformation`. The tearsheet and its metadata are conveyed in `ProofOfPublicationInformation/adsml-pp:TearSheet`. The tearsheet itself is transmitted as encoded binary data in `adsml-pp:TearSheet/Rendering.Tearsheet/adsml:ContentData`. A description of the tearsheet, including its format and the encoding method that was used, should be placed in `Rendering.Tearsheet/adsml:ContentProperties`. Because the tearsheet is being delivered in-line, the `Delivery.TearSheet` element is not used.

Notes:

- The ability to include an e-tearsheet inside an invoice message is intended to support workflows in which the tearsheet itself will be relatively lightweight, for example, a textual representation of a published lineage ad. Large binary e-tearsheets should usually be transmitted separately and referenced from the invoice rather than contained within it.
- Trading partners must agree in advance on whether they will support in-line transmission of e-tearsheets, and if so, what formats or encoding methods may be used.
- See the AdsMLProofOfPublication specification for more information about elements that are defined in the `adsml-pp` namespace, and the AdsMLMaterials specification for information about elements in the `adsml-ma` namespace.

6.1.19 Convey detailed information about a tearsheet sent separately

Scenario: Invoicer is invoicing for the publication of one or more advertisements for which a digital or hard-copy tearsheet has been or will be sent to the Payer. For each published advertisement, Invoicer wishes to identify the matching tearsheet and provide delivery information about it.

AdsML handling: Invoicer generates a separate invoice line item for each published ad, using `Item.AdvertisementPublication`. An identifier for the tearsheet is conveyed in `adsml-pp:ProofOfPublicationReference`, while supporting information about when, where and how the ad was published can go in `adsml-pp:AppearanceInformation`. Metadata about the tearsheet (such as its QID and format) is conveyed in `ProofOfPublicationInformation/adsml-pp:TearSheet`. Information about how the tearsheet is being delivered, including retrieval instructions if appropriate, is conveyed in `Delivery.TearSheet`.

Note: See the AdsMLProofOfPublication specification for more information about elements that are defined in the `adsml-pp` namespace, and the AdsMLMaterials specification for information about elements in the `adsml-ma` namespace.

6.1.20 Identify the tearsheet for an invoice line item

Scenario: Invoicer is invoicing for the publication of one or more advertisements for which an e-tearsheet will be sent to the Payer. For each published advertisement, Invoicer wishes to provide an identifier for the matching tearsheet.

AdsML handling: Invoicer generates a separate invoice line item for each published ad, using `Item.AdvertisementPublication`. The business-significant identifier for the matching tearsheet is conveyed in `adsml-pp:ProofOfPublicationReference`, while supporting information about when, where and how the ad was published can go in `adsml-pp:AppearanceInformation`.

Note: See the AdsMLProofOfPublication specification for more information about elements that are defined in the `adsml-pp` namespace.

6.1.21 Send a paid invoice

Scenario: Invoicer sends an invoice for which payment has already been received, for example, for a classified ad that was pre-paid by credit card.

AdsML handling: Details of any payments that have already been applied to this invoice are recorded in one or more instances of the `PrepaidPayment` structure in the invoice header, one for each prepayment that has been received. At a minimum the `PrepaidPayment/PaidAmount` element must be populated with the amount of the received payment, and optionally the `ReceivedDate` and `ID` sub-elements can be used to provide the date on which the payment was made and a check number or similar identification. Information about the method of payment can be conveyed by populating a `PaymentMeans` structure in the document header in which `PaymentMeans/PaymentID` contains the same value as the `PrepaidPayment/ID` for the prepayment in question. Use `PaymentMeans/PaymentMeansCode` to characterize the payment mechanism for the payment in question, and populate the `PaymentMeans/CardAccount` structure if the payment was made by credit card and you wish to echo some or all of the credit card details back to the Payer.

Notes:

- The amount of the payment previously received should be expressed as a negative number.
- Payments received do not affect the `TaxInclusiveAmount` in the document footer, but they do affect its `PayableAmount`. For example, a pre-paid invoice for 1,000 Euros would have a `TaxInclusiveAmount` of `1,000` and a `PrepaidPayment/PaidAmount` of `-1,000` to indicate receipt of full payment, resulting in a `LegalMonetaryTotal/PayableAmount` of `0`.
- The `CardAccount` structure is designed to support many different usage scenarios, and therefore contains more elements than are needed to confirm a pre-payment.

6.1.22 Specify payment terms with penalty charges

Scenario: Invoicer wishes to specify that payment is due in 30 days, after which a 1.5% penalty will be applied.

AdsML handling: Payment terms, including penalties, are conveyed in the `adsML:PaymentTerms` element in the document header. A textual description of the payment terms, such as "`Late payments will be assessed a penalty of 1.5% per month`" can go in `adsML:Note`, along with an appropriate `PaymentTermsCode` value such as "`FixedDateLatePaymentPenaltyApplies`".

To specify the penalty in a machine-processable form, indicate a `PenaltySurchargePercent` of "`1.5`" and a `PenaltyPeriod` that starts on the invoice due date and extends for 30 days. (Or alternatively, populate `EndDateTime` rather than `DurationMeasure`.)

Notes:

- Only the first penalty period applicable to an invoice can be described using the `PenaltyPeriod` and `PenaltySurchargePercent` elements. Additional penalty periods, if any, should be referenced in the `adsml:Note` element as described above.
- Trading partners wishing to use the Penalty structures described above should agree in advance on exactly which elements they will exchange and in what combinations.

6.1.23 Send an invoice with line items in a different currency

Scenario: Invoicer wishes to submit an invoice (and receive payment) in US Dollars, but all of the line items on the invoice describe charges that were incurred in Euros. Invoicer also needs to provide the exchange rate that was used when converting the Euro charges into USD.

AdsML handling: Populate `adsml:DocumentCurrencyCode` in the document header with "`USD`" to indicate that the primary currency for the invoice as a whole is dollars. On each line item, populate `adsml:CurrencyCode` with "`EUR`" to indicate that the pricing for that line item is in Euros.

If all of the line items share the same currency and were calculated using the same exchange rate, the details of the currency conversion between Dollars and Euros can be specified in `ExchangeRate` in the document header.

If more than one exchange rate was used during the preparation of the invoice, then instead of using `ExchangeRate` in the document header, the conversion details should be recorded in `InvoiceLine/ExchangeRate`. This allows for different currency conversion details to be specified on each line item.

For informational purposes, in each line populate `LineItemCurrencyExtensionAmount` with the pre-tax subtotal for that line expressed in Euros.

Notes: The currency specified on a line applies only to the `CalculatedPrice`, `AdditionalAllowanceCharge` and `LineItemCurrencyExtensionAmount` structures on that line.

6.1.24 Send an invoice with line items in varying currencies

Scenario: Invoicer wishes to submit an invoice (and receive payment) in US Dollars, but the line items on the invoice describe charges that were incurred in Euros, British Pounds and Swiss Francs. Invoicer also needs to provide the exchange rates that were used when converting these charges into USD.

AdsML handling: Populate `adsml:DocumentCurrencyCode` in the document header with "`USD`" to indicate the primary currency for the invoice as a whole. On each line item, populate `adsml:CurrencyCode` with the currency code applicable to that line, and record the details of the currency exchange applied to that line in `InvoiceLine/ExchangeRate`.

For informational purposes, in each line populate `LineItemCurrencyExtensionAmount` with the pre-tax subtotal for that line expressed in the line item currency.

Note: The currency specified on a line applies only to the `CalculatedPrice`, `AdditionalAllowanceCharge` and `LineItemCurrencyExtensionAmount` on that line.

6.1.25 Send a PDF of the invoice along with the XML message

Scenario: Invoicer is required by local regulations to submit a copy of the printed invoice in order to be paid. Therefore, Invoicer needs to transmit a PDF of the invoice along with the invoice message.

AdsML handling: The `adsml:DocumentRendering` structure in the financial document footer allows the sender to convey a digital rendering of the document either by containership (e.g. a PDF is embedded in the message) or reference (a URL or equivalent is provided so that the recipient can automatically retrieve the rendering).

Populate `adsml:ContentProperties` with the filename and other descriptive metadata about the rendering. To convey the rendering inside the XML message, encode it in an XML-friendly fashion and place it in the `adsml:ContentData` element. (Be sure to identify the type of encoding in `adsml:ContentProperties`.) To convey it by reference, specify the location from which it can be retrieved using one of the `adsml:CommunicationChannel` elements.

Notes:

- The elements in `adsml:DocumentRendering` are also found in the AdsMLMaterials and AdsMLProofOfPublication standards. See the documentation of those standards for more detailed instructions on how to convey an `adsml:Rendering`.
- The ability to convey digital renderings is not limited to PDF format, and can be used for any kind of financial document, not just invoices. However, the `DocumentRendering` structure cannot be used to reference the delivery of a non-digital (e.g. paper) document
- Trading partners wishing to exchange embedded digital objects such as PDFs in their messages must agree in advance to support this capability.

6.1.26 Acknowledge receipt of an invoice

Scenario: Payer acknowledges receipt of an invoice and indicates whether it contains line items that cannot immediately be approved for payment.

AdsML handling: Payer sends an FD-NVR (Invoice Response) message to convey its initial "business significant" response to the invoice. The QID of the invoice to which this is a response is placed in `DocumentIdentifier`, and the date on which the information in this response was generated goes in `adsml:BusinessMessageDate`.

The status of the invoice in the payer's system should be conveyed in `adsml:Status`. This should take the form of a code value from the AdsMLStatusCodeCV plus, if appropriate, a supplementary text message and/or additional `adsml:StatusQualifier` code to provide further information.

Notes:

- This scenario assumes that the payer has previously received the invoice, sent an immediate Administrative Response, loaded the invoice into its financial system, and performed an initial examination of the line items to see whether they match existing orders.
- The status information in the response message applies to all of the line items in the invoice. It is not possible to provide separate responses for particular line items.
- Trading partners should agree in advance on the `Status` code values that they are going to use in Response and Status messages. It is recommended that the AdsMLStatusCodeCV be used, which will have the desirable side-effect of keeping them simple and high-level. In this example, an initial status of "BeingProcessed" would indicate that the invoice has been loaded into the payer's system and no issues have been identified.
- If the payer wishes to reject an invoice without even loading it into their financial system, for example because it has accidentally been sent to the wrong party, the payer should populate `adsml:RequestDenied` rather than `adsml:NatureOfResponse`.

6.1.27 Indicate that an invoice contains issues requiring resolution

Scenario: Payer has previously acknowledged receipt of an invoice, but following further processing has discovered that it contains apparent errors or discrepancies that will need to be discussed and resolved before the invoice can be approved.

AdsML handling: Payer sends an FD-NVS (Invoice Status) message to convey the revised status of the invoice. The QID of the invoice is placed in `DocumentIdentifier`, and the date on which the status information was generated goes in `adsml:BusinessMessageDate`.

The status of the invoice is conveyed in `adsml:Status`. This should take the form of a code value from the AdsMLStatusCodeCV plus, if appropriate, a supplementary text message and/or additional `adsml:StatusQualifier` code to provide further information. In this case a status value of "OnHold" should be used, to indicate that the invoice is on hold pending issue resolution.

Notes: This status can also be conveyed in the initial Invoice Response message, if the problem is already known at that time.

6.2 Credit notes

6.2.1 Send a credit note to resolve a claim

Scenario: Invoicer sends a credit note to a payer based on the resolution of a claim or query about a previously sent invoice. The credit note retroactively adjusts the total amount due for the invoice in question, and includes information explaining the reason for the adjustment.

AdsML handling: The sender creates a credit note with a single line item. The reason for the credit is conveyed as free text and/or machine-processable codes in `CreditLine/CreditReason`, and the invoice to which this credit applies is referenced in `CreditLine`/`RelatedInvoice`. The total amount of the credit before taxes is placed in `LineExtensionAmount`.

Notes:

- The credit note could contain other line items reflecting various credits being issued to the same payer.
- Credit amounts should be conveyed as negative numbers, reflecting the fact that they reduce the balance that is owed by the payer to the invoicer.
- A breakdown of the credit and any related taxes can be conveyed using the `CalculatedPrice`, `AdditionalAllowanceCharge` and `TaxTotal` structures as necessary. These are populated exactly as they would be on an invoice.
- If the invoicer wishes to identify the specific booking or placement to which this credit applies, he populates `Item.AdvertisementPublication` with the relevant identifiers.

6.2.2 Send a credit note that does not reference an invoice or query

Scenario: Invoicer sends a credit notice that is based on the buyer's fulfillment of a contractual agreement, and does not reference a particular invoice, booking or query.

AdsML handling: The sender creates a credit note containing a line item reflecting the credit in question. The reason for the credit is conveyed as free text and/or machine-processable codes in `CreditLine/CreditReason`. The total amount of the credit before taxes is placed in `LineExtensionAmount`. Information about the contract which triggered this credit should be provided in `adsml:Contract`.

Note:

- A credit note can also directly reference a booking, as described above.
- If the credit is based on a prior document other than a contract, that document should be identified using either `adsml:PurchaseOrderReference` or `adsml:OtherReference`.

6.2.3 Acknowledge receipt of a credit

Scenario: Payer acknowledges receipt of a credit message and indicates whether it contains line items that cannot be accepted as stated.

AdsML handling: Payer sends an FD-CRR (Credit Note Response) message to convey its initial "business significant" response to the credit. The QID of the credit to which this is a response is placed in `CreditNoteResponse/DocumentIdentifier`, and the date on which the information in this response was generated goes in `adsml:BusinessMessageDate`.

The status of the credit in the payer's system should be conveyed in `adsml:Status`. This should take the form of a code value from the AdsMLStatusCodeCV plus, if appropriate, a supplementary text message and/or additional `StatusQualifier` code to provide further information.

Notes:

- This scenario assumes that the payer has previously received the credit, sent an immediate Administrative Response, loaded the credit into its financial system, and performed an initial examination of the credit line items.
- The status information in the response message applies to all of the line items in the credit. It is not possible to provide separate responses for particular line items.
- Trading partners should agree in advance on the `Status` code values that they are going to use in Response and Status messages. It is recommended that the AdsMLStatusCodeCV be used, which will have the desirable side-effect of keeping them simple and high-level. In this example, an initial status of "`BeingProcessed`" would indicate that the credit has been loaded into the payer's system and no issues have been identified.
- If the payer wishes to reject a credit without even loading it into their financial system, for example because it has accidentally been sent to the wrong party, the payer should populate `adsml:RequestDenied` rather than `adsml:NatureOfResponse`.

6.3 Financial documents (both invoices and credits)

6.3.1 Integrate AdsMLFinancials with another financial standard

Scenario: A group of trading partners who are currently transmitting invoice and credit messages that conform to a non-AdsML standard, such as UBL or *Finvoice,* wish to incorporate AdsML structures into their messages in order to add advertising-specific information to the generic line-items that the other standard provides.

AdsML handling:

Depending on the extension capabilities of the hosting non-AdsML e-invoicing standard, AdsMLFinancials' advertising-specific structures may be reused at the line item level, or at the document level with optional pointers to specific line items within the invoice. It is, however, required that the hosting standard have extension points defined at either level that allow content from the AdsMLFinancial's namespace to be included.

The main advertising specific structure in AdsMLFinancials is `Item.AdvertismentPublication`. This structure includes a line item description with detailed machine processable data about how the advertisement was originally booked, and how it eventually appeared, both in terms of metadata describing the appearance and as referenced or included electronic tearsheets.

In case the full `Item.AdvertisementPublication` is not wanted, it is also possible to reuse AdsMLFinancials' structures at a lower level. Child elements such as `adsml-bo:BookingDate`, `BookingInformation.Insert` or `adsml-pp:TearSheet` are all defined as global elements in the AdsML schema set and available for reuse. It is, however, **RECOMMENDED** to reuse AdsML structures in other financial standards starting with the parent element wrapper, i.e. using the `Item.AdvertismentPublication`.

`Item.AdvertismentPublication` is particularly designed for reuse at the line item level. In case the hosting standard does not support extension at that level, an alternative version of `Item.AdvertisementPublication`, `AdsMLItem.AdvertisementPublication`, is available for reuse at the document level. `AdsMLItem.AdvertisementPublication` is an extension of `Item.AdvertismentPublication` that includes a *`lineItemRef`* attribute that can hold a pointer to a particular line item in the invoice. Multiple instances of `AdsMLItem.AdvertisementPublication` can be conveyed in a non-AdsML financial message, each of them providing advertising-specific extensions to a different line item in that message.

Notes:

This integration mechanism is one-way only: it enables AdsMLFinancials structures to be imported into other, more generic e-commerce formats. It is not possible to do the reverse and import schema structures specified in a non-AdsML standard into an AdsMLFinancials document.

6.3.2 Provide a stylesheet that can be used to format the document

Scenario: Invoicer wishes to provide a stylesheet that can be used by the document recipient to view the data in a user-friendly format.

AdsML handling: Invoicer makes the stylesheet available at a publicly accessible location, and provides the URL, URI or other identifier in the *presentationTransformation* attribute in the document header. The document recipient retrieves the appropriate stylesheet and applies it to this business message.

Notes:

- The stylesheet can also be transmitted to the recipient in advance, in which case the *presentationTransformation* attribute may contain a filename or any other string that appropriately identifies it.
- This *presentationTransformation* attribute is available in all AdsML messages.

7 Configuration checklist

In order to facilitate implementation and interoperability, pre-defined packages of features and functionality are a valuable tool. Please see the *E-Commerce Usage Rules and Guidelines* document for a general discussion on this subject.

The following packages of features have been defined to date. Each package consists of either:

- a set of hierarchical levels from which one must be selected (represented by a numbered list), or
- a set of non-exclusive options from which any combination can be selected (represented by a bullet list), or
- a list of mutually-exclusive choices from which one must be selected (represented by a textual description).

7.1 Financial document types

There are four available types of financial documents:

1. Invoice
2. Invoice Status
3. Invoice Status Enquiry
4. Credit Note

Trading partners must agree on which types of messages they will exchange.

7.2 Message exchange mode

There are two defined message exchange modes:

1. Full Request-Response
2. Datagram model from invoicer to payer

Trading partners must select one of these exchange modes.

Note that the ability to send and receive Administrative Responses is a fundamental feature of AdsML messaging and is required in all modes.

7.3 Use of advertising specific structures in line items

The AdsMLFinancials line item descriptions can include a detailed set of machine processable advertising specific structures wrapped in the `AdvertisementPublicationInstance` element. There are three available structures for this purpose:

- `BookingInformation`
- `AppearanceInformation`
- `ProofOfPublicationInformation`

Trading partners should agree on which of these structures, if any, should be supported.

Note that within each structure the partners, as always, get to decide which specific data to implement.

7.4 Credit lines within an invoice

Trading partners must agree on whether they will allow an invoice to include credit note line items.

7.5 Multiple Currencies

AdsMLFinancials allows for different currencies per line item. Trading partners should agree on whether or not to use this feature.

7.6 Inclusion of binary objects

A financial document may include binary objects. Trading partners should agree whether this feature is supported at all, and if so, which types of objects are supported. The options are:

- A document rendering, for instance a PDF of a human readable invoice.
- One or more digital tearsheets.

7.7 Use of multilingual metadata

Trading partners must agree on whether or not they support the provision of alternative versions of human-readable textual metadata in more than one language. (For example, alternative versions of a description or note can be provided, each in a different language.)

If multilingual metadata is supported, trading partners need to agree on:

- Which languages will they use in their messages?
- Which language, if any, takes priority as the 'default language' of the message?
- Any rules for processing and presenting multilingual content to users.

7.8 User defined Properties

Use of user defined properties, i.e. the `adsml:Properties` element, need to be agreed in advance by trading partners. A receiver of a message **SHOULD** ignore any unknown user defined properties.

8 Appendix A: Acknowledgment for contributions to this document

This document is a product by the AdsML technical working group.

Acknowledgment and thanks for contributions to this specification are also due to,

- Mark Bradford (Associated Newspapers Ltd)
- Michael Brier (Neasi-Weber International)
- Richard Cichelli (Software Consulting Services, LLC)
- Jed Cope (Accord Holdings)
- Merv Griffin (Atex)
- Mohammad Samarah (Mediaspan Media Software)
- Naomi Smigel (Time Inc)
- Mark Stepuszek (Tribune Company)
- Claude Studer (Publicitas)
- Don Woodall (Washington Post).