# AdsML® Framework for E-Commerce Business Standards for Advertising

## Structured Descriptions of Advertisement Objects 1.0.8
## Part 2
## Specification & Schema

Document Authors: AdsML Technical Working Group

Document ID: AdsMLStructuredDescriptions-1.0.8-SpecP2Schema-AS-9

Document File Name: AdsMLStructuredDescriptions-1.0-SpecP2Schema-AS.pdf

Document Status: Approved Specification

Document Date: 15 April 2010

Draft Number: 9

# Table of Contents

# 1 AdsML Standard Documentation

## 1.1 Document status and copyright

This is the Approved Specification of the *AdsML Structured Descriptions of Advertisement Objects 1.0 Part 2 Specification & Schema*.

## 1.2 Non-Exclusive License Agreement for AdsML Consortium Specifications

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 ID AdsMLStructuredDescriptions-1.0.8-SpecP2Schema-AS-9, is freely available. It will be located at the AdsML website at http://www.adsml.org/.

## 1.5 Purpose of this document

This document specifies the definition of the AdsML Structured Descriptions standard. AdsML Structured Descriptions is an XML-based language used for encoding and exchanging structured descriptions of advertising content.

It exemplifies how an AdsML structured descriptions message is created.

This document and its associated XML Schema together provide a normative definition of the AdsML Structured Descriptions standard.

## 1.6 Audience

The intended audience for this document is primarily user and vendor organizations who seek to implement the AdsML Structured Descriptions standard in their workflows, advertising systems, or software products. Those assessing the conformance of vendor products to the AdsML 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 serves as the reference guide to the AdsML Structured Descriptions schema.  A companion document, *AdsML Structured Descriptions of Advertisement Objects 1.0 Part 1 Usage Rules & Guidelines*, provides additional rules and guidance for using AdsML Structured Descriptions messages to address specific business requirements. 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. AdsML Structured Descriptions 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 (See Section 12 References). 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, '`AdsML`', '`StructuredDescriptions`', and '`Entry`'.

Attribute names begin using lower camel case and begin with lower case (*`lowerCamelCase`*). For example, '*`value`*' or '*`formatMask`*'.

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`'.

### 1.8.3 Typographical conventions

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

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

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 '`11`'.".

## 1.9 Change History

| Draft | Date | Changes | Author |
|---|---|---|---|
| AS 1 1.0.0 | 23 September 2005 | Approved Specification. Earlier change history removed. | JC, CRo, MD |
| AS 2 1.0.1 | 14 December 2005 | Approved Specification. Minor version increment to schema to use current PS versions of the Type Library and Controlled Vocabularies schema. | JC |
| AS 3 1.0.2 | 1 June 2006 | Approved Specification. Minor version increment to schema to use AS versions of the AdsML Type Library and Controlled Vocabularies schema. | JC |
| AS 4 1.0.3 | 1 October 2006 | Maintenance version release; no significant changes. AdsML references updated to reflect registered trademark status. | JC |
| AS 5 1.0.4 | 1 October 2006 | Updated to use Controlled Vocabularies 3.0. No other changes. | UW |
| AS 6 1.0.5 | 10 October 2007 | Updated to use AdsML Type Library 2.0, editorial revisions, updated references. | JC |
| AS 9 1.0.8 | 15 April 2010 | Minor editorial revisions. Errata in minor versioning corrected. | JC |

## 1.10 Acknowledgements

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

- Christian Rohrbach (IWARE SA (Publigroupe)) crohrbach@iware.ch
- Jay Cousins (RivCom.) - jay.cousins@rivcom.com
- Marcel Dumont (Rosetta) - marcel@rosetta.nl

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

# 2 Introduction

This section provides an overview of the key concepts of the Structured Descriptions standard and how it relates to the AdsML Framework for E-commerce Business Standards for the Advertising Industry. It provides an overview of the key concepts of the Structured Descriptions standard that are required to use this specification.

Please see the *AdsML Framework - Overview* and *E-commerce Usage Rules & Guidelines* for a more thorough discussion about the AdsML approach to e-commerce. Also see AdsML *Structured Descriptions of Advertisement Objects 1.0 Part 1 Usage Rules & Guidelines* for explanations of how to use AdsML Structured Descriptions.

## 2.1 Relationship to the AdsML Framework

AdsML provides an XML framework, called the "AdsML Framework for E-commerce Business Standards for the Advertising Industry", for unifying and extending XML advertising standards. Where existing standards such as IFRA adConnexion or CREST focus 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 AdsML Structured Descriptions standard has been developed as the preferred approach for defining and recording structured metadata about advertisement objects contained in ad content for multiple media. During use structured description instance data is designed to be ‘embedded’ as a ‘module’ in an AdsML standard, as is the case of its use in the AdsML Bookings and AdsML Materials specifications of the AdsML Framework.

When trading partners agree on the type(s) of structured description data that they will exchange, then this agreement constitutes a natural part of the TPA between these trading partners. However, it should be noted that a TPA must never overrule the definitions in the AdsML Structured Description specification.

### 2.1.1 Relationship to other advertising standards

AdsML Structured Descriptions embraces functionality previously covered by older standards. In particular by,

- CREST. Developed by the Classified Advertising Standards Task Force of NAA[1], CREST 2.0 is an XML-based media independent format for electronically exchanging and sharing classified advertising data. CREST focuses on the three main areas of classified advertising - real estate, transportation, and employment categories, and provides a generic extension mechanism to record advertising data that falls outside these categories. CREST 2.0 supersedes the earlier '*CREST® NAA Guidelines for Classified Advertising Remote Markup and Transmission, Version 1.0, May 1995*'. (See the entry for CREST in Section 12.2 Other references.)

## 2.2 AdsML Structured Descriptions overview

The AdsML Structured Descriptions standard defines an XML format and approach for describing in a structured, machine-processable way the objects or services that are offered for sale in an advertisement, in order to support the automated classification, syndication and aggregation of advertisements by multiple

[1] NAA is an acronym for 'Newspaper Association of America'.

publishers, and then searching and querying of the contents of those ads by potential consumers. While the format is specifically intended to convey the description of items in classified ads that are primarily textual (e.g. "liner" ads), it will be capable of describing the advertised items in any kind of advertisement in any medium.

The AdsML Structured Descriptions standard defines the structures required to create *advertisement object definitions* and *structured descriptions* of advertisement objects.

The first step in the structured descriptions process is for trading partners to agree what kind of metadata they will use to describe ad objects exchanged between them. The partners identify the types of ad object they will exchange and define the set of properties, together with any constraints on those properties (e.g. cardinality, data type, allowed values), that they will use to describe those objects. This specification of the properties used to describe an ad object is known as a *rule set*. A *rule set* is a generic information model based on a simple property and property group structure that provides a standardized data model for defining the properties of an ad object.

A *rule set* is expressed as a document, and the AdsML Structured Descriptions standard defines both spreadsheet and XML versions of *rule sets*. The spreadsheet version is easier to read and maintain and is used when first creating and recording a *rule set*. The spreadsheet version of the *rule set* can also be expressed as XML. The XML version of a *rule set* is represented as an XML instance conforming to the *advertisement object definitions* schema structure described below. This XML version is used for storing and exchanging a *rule set* and is also used to support the validation of instance ad object data.

The *advertisement object definitions* schema structure provides an XML structure that is capable of representing a *rule set*. It is a generic model with which the properties used to describe a particular type of advertisement object (i.e. a *rule set*) can be defined as an *advertisement object definition* using a standard property and property group structure that enables properties to be recorded and constraints applicable to those properties to be expressed. This creates a generic and standardized data model for an advertisement object definition. This standard property model gives interoperability. Further, the advertisement object definitions schema allows *value tables* to be defined. This allows a list of *values* to be defined to allow for the case where a property has a predefined set of specific *allowed values* associated with it.

The *structured descriptions* schema structure provides a simple XML structure with which an actual (i.e. real) structured description of an object in an advertisement is recorded as a set of properties expressed as simple name:value pairs. Each instance *structured description* is associated by reference with the *advertisement object definition* in which the properties (and their associated constraints) used to describe that type of advertised item have been defined.

Using this method, trading partners are able to agree and create advertisement object definitions for the types of ad object that they exchange in a standardized way. The simplicity of this structure has two purposes,

1. To simplify the exchange process by exchanging instance data as name:value pairs, the structured description referencing the associated advertisement object definition. This ensures that when exchanging actual ad content only variable data is transmitted (i.e. the instance structured

description data) and that static data (i.e. the advertisement object definition) is not resent.

2. To enable a structured description to be embedded in the ad content that it describes as a simple XML fragment from the structured descriptions namespace.

The diversity of handling multiple structured descriptions with different properties is removed from the exchange process and is replaced by a common data model. Further, validation of instance data is simplified and control over the level of validation is given to the user. Firstly, validation is removed from the exchange, and the exchange process is seen as a pure serialization-deserialisation process during which property names and values are not validated beyond the structural and data type constraints of the structured description schema. Secondly, the extent and method by which the user validates a structured description instance for conformance to an advertisement object definition is up to the user. A user may validate by writing their own custom code, or they may do so by using standard XML tools. How users choose to perform this validation is for a user to decide, but a method for performing this validation using XML tools is given in this specification. Using XSLT, (a) the advertisement object definition can be used as a meta schema from which an XML Schema representation of the advertisement object definition is generated, (b) an instance of that specific schema can be populated using data from a structured descriptions instance, (c) the data can then be XML Schema validated for conformance to the advertisement object definition constraints, and (d) a report of conformance to non-schema enforceable constraints can be generated.

Each of these areas is covered in the following parts of this specification,

- Section 4 Advertisement Object Definitions schema vocabulary defines the elements, attributes, and types used by the *advertisement object definitions schema*.
- Section 5 Structured Descriptions schema vocabulary defines the elements, attributes, and types used by the *structured descriptions schema*.
- Section 6 Common component listing defines the common elements, attributes, and types used by the *advertisement object definitions schema* and the *structured descriptions schema*.
- Section 7 Creating & maintaining user-defined rule sets for advertisement object definitions explains how to create a user-defined advertisement object definition
- Section 10 Validation of instance data explains how to apply additional validation to the data contained in a structured description instance by creating a specific schema from an advertisement object definition using W3C XML Schema, and populating an instance conformant to that specific schema with instance structured description data in order to,
    - Validate that instance data conforms to the advertisement object definition constraints using XML Schema validation
    - Report that instance data conforms to the advertisement object definition constraints that cannot be enforced using XML Schema validation

# 3 AdsMLStructuredDescriptions XML Schema – Overview

This section describes the use of XML Schema in the definition of AdsMLStructuredDescriptions.

## 3.1 Schema Architecture

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

- The **Main Schema** – This schema defines the root element AdsMLStructuredDescriptions and all other components used in the standard, either by local definitions or by importing and/or including other schema files.
- The **Public Type Library** – This schema includes all components from AdsMLStructuredDescriptions that may be imported into other standards and reused.
- The **AdsML Type Library** – This schema defines reusable components from the AdsML Framework.
- The **AdsML Controlled Vocabularies** – This schema defines all controlled vocabularies recommended by the AdsML Consortium.
- **The "xml" Namespace schema** - This schema defines the W3C 'xml:' attributes used by AdsMLStructuredDescriptions.

All structures specific to AdsMLStructuredDescriptions are defined in the Main Schema or the Public Type Library that is included into the Main schema. These structures are all defined in the AdsMLStructuredDescriptions namespace.

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

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

### 3.1.1 Schema Files

The schema files from a particular standard are named as follows:

```
AdsMLStructuredDescriptions-1.0-Main-AS.xsd
```

It starts with the name of the standard, "AdsMLStructuredDescriptions" followed by current version number and name of the schema within the standard. The last two characters provide the status of the standard as either 'PS' (Proposed Standard) or 'AS' (Approved Standard) for public releases (internal working document have status code 'WD' for Working Draft).
The complete set of schema files for the AdsMLStructuredDescriptions version 1.0, Approved Standard is thus:

```
AdsMLStructuredDescriptions-1.0-Main-AS.xsd
```

```
AdsMLStructuredDescriptions-1.0-PublicTypeLibrary-AS.xsd
AdsMLTypeLibrary-2.0-AS.xsd
AdsMLControlledVocabularies-3.0-AS.xsd
xml.xsd
```

Note that successive versions of the standard will import and use later versions of the AdsML Framework Type Library and Controlled Vocabularies schema as the Framework evolves.

## 3.2 AdsMLStructuredDescriptions Namespace

AdsMLStructuredDescriptions defines a namespace:

`'http://www.adsml.org/adsmlstructureddescriptions/1.0'`

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

```
<xs:schema
targetNamespace="http://www.adsml.org/adsmlstructureddescriptions/1.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.adsml.org/adsmlstructureddescriptions/1.0"
xmlns:adsml-sd="http://www.adsml.org/adsmlstructureddescriptions/1.0" ... >
```

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

```
adsml='http://www.adsml.org/adsmltypelibrary/2.0'

adsml-cv='http://www.adsml.org/adsmlcontrolledvocabularies/3.0'

xml='http://www.w3.org/XML/1998/namespace'
```

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

## 3.3 Validation and Schema Location

Although possible to use AdsML Structured Descriptions standalone, its expected usage is as a module 'plug in' used by other AdsML standards, in which case the AdsML Structured Descriptions content is validated as part of the containing message.

A trading partner **MUST NOT** send any invalid AdsML Structured Descriptions content, as a standalone message or embedded within another AdsML message. However, use of XML Schema based validation of production messages in runtime is **OPTIONAL**. Systems are allowed to use any available approach to ensure that their output is valid.

When structured descriptions content is used as content embedded within another message, then the rules for validation and schema location for that message naturally apply to the structured descriptions content of that message.

In the case of AdsML Structured Descriptions message being used standalone, then a schema location **SHOULD NOT** be given in document instances using the *`xsi:schemaLocation`* attribute. Systems are **REQUIRED** to be able to identify which schema a particular document instance belongs to by reading the mandatory *`adsml:schemaVersion`* attribute.

## 3.4 Empty values for elements and attributes

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

## 3.5 Fixed and Default values

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

## 3.6 AdsML defined advertisement object definitions

AdsML Structured Descriptions has currently defined AdsML model vocabularies for describing the following categories of metadata: Autos, Homes, Miscellaneous Goods, Miscellaneous Services, Recruitment, and ReplyTo.[2] Users are able to use these vocabularies or define their own as required by their business circumstance.

See Section 7 Creating & maintaining user-defined rule sets for advertisement object definitions for how to create a user-defined advertisement object definition.

# 4 Advertisement Object Definitions schema vocabulary

The elements and attributes of the advertisement object definitions schema are defined below.

## 4.1 AdsMLAdObjectDefinitions element

The `AdsMLAdObjectDefinitions` element is the root element of an advertisement object definitions instance document and provides a container for individual advertisement object definition(s) and value table(s). Uniqueness constraints are specified in the `AdsMLAdObjectDefinitions` element to ensure that,

- The value of the *`objectDefinitionURI`* attribute of the `ObjectDefinition` element is unique amongst all instances of `ObjectDefinition` elements within the `AdsMLAdObjectDefinitions` file

[2] Note: these model vocabularies are available as rule sets recorded in the spreadsheet format.

- The value of the *`propertyID`* attribute of the `ObjectProperty`, `ObjectPropertyGroup`, or `ObjectPropertyModifier` elements is unique amongst all instances of those elements within the `AdsMLAdObjectDefinitions` file
- The value of the *`valueTableURI`* attribute of the `ValueTable` element is unique amongst all instances of those elements within the `AdsMLAdObjectDefinitions` file.

```
<xs:element name="AdsMLAdObjectDefinitions" type="AdObjectDefinitionsType">
 <xs:unique name="ObjectDefinitionURIUniqueConstraint">
  <xs:selector xpath="adsml-sd:ObjectDefinition"/>
  <xs:field xpath="@objectDefinitionURI"/>
 </xs:unique>
 <xs:unique name="ObjectPropertyIDUniqueConstraint">
  <xs:selector xpath=".//adsml-sd:ObjectProperty|.//adsml-
sd:ObjectPropertyGroup|.//adsml-sd:ObjectPropertyModifier"/>
  <xs:field xpath="@propertyID"/>
 </xs:unique>
 <xs:unique name="ValueTableURIUniquenessConstraint">
  <xs:selector xpath=".//adsml-sd:ValueTable"/>
  <xs:field xpath="@valueTableURI"/>
 </xs:unique>
</xs:element>
```

The `AdsMLAdObjectDefinitions` element contains a sequence of required and repeatable `ObjectDefinition` element(s) followed by an optional `ValueTables` element. The `ObjectDefinition` element records the set of properties used to define the characteristics of a category of ad object. If present, the `ValueTables` element contains a list of controlled values that have been designated as the set of allowed values for a property or property modifier in an object definition.

A required *`adObjectDefinitionsURI`* attribute records an identifier for the ad object definitions file in the form of a URI using the `URIType` data type defined in the AdsML Type Library.

An optional *`xml:base`* attribute allows a base URI to be set for interpreting any relative URIs that appear in the element scope of the `AdsMLAdObjectDefinitions` element. Note that if a base URI set at the `AdsMLAdObjectDefinitions` element level can be overridden further down the element tree using optional *`xml:base`* attributes on the `ObjectDefinition` and `ValueTable` elements. This enables URIs to be used flexibly inside an ad object definitions file.

```
<xs:complexType name="AdObjectDefinitionsType">
 <xs:sequence>
  <xs:element ref="ObjectDefinition" maxOccurs="unbounded"/>
  <xs:element ref="ValueTables" minOccurs="0"/>
 </xs:sequence>
 <xs:attribute name="adObjectDefinitionsURI" type="adsml:URIType"
use="required"/>
 <xs:attribute ref="xml:base" use="optional"/>
</xs:complexType>
```

## 4.2 ObjectDefinition element

An `ObjectDefinition` element names, describes, and classifies a category or type of ad object.

The required `ObjectDefinitionHeader` provides header-level metadata that names and describes an object definition.

An optional `IndustryCode` element records an industry classification of the advertisement content that can be described using the object definition. See definition of `IndustryCode` element for more information about industry code.

The collection of properties that describe the characteristics of the advertisement object defined by an object definition are specified by an optional and repeatable choice between `ObjectProperty` and `ObjectPropertyGroup` element(s). Note that properties are optional to allow for the use case where an 'empty' top level object definition may be specified for the purpose of grouping individual object definitions in a containing object hierarchy.

Unique identification and hierarchical class relationships for an object definition are expressed using the `ObjectDefinition`'s identification attributes. A required *`objectDefinitionURI`* attribute records an identifier for the object definition in the form of a URI using the `URIType` data type defined in the AdsML Type Library. This attribute provides a machine-readable and externally addressable unique id for the object definition, enabling the object definition to be referenced in a structured description instance. An optional *`parentObjectDefinitionURIRef`* attribute enables hierarchical relationships between object definitions to be expressed, identifying the parent object definition by referencing its *`objectDefinitionURI`* attribute.

If present, an optional *`xml:base`* attribute allows a base URI to be set for the scope of the `ObjectDefinition` element if needed. For example, a base URI would be specified in the event that relative URIs are used in the scope of the `ObjectDefinition` element and the base for those URIs differs to a base URI set further up the element hierarchy in the `AdObjectDefinitions` element.

An optional *`xml:lang`* attribute allows the human language used in the object definition to be identified.

```
<xs:element name="ObjectDefinition" type="ObjectDefinitionType"/>

<xs:complexType name="ObjectDefinitionType">
 <xs:sequence>
  <xs:element ref="ObjectDefinitionHeader"/>
  <xs:element ref="IndustryCode" minOccurs="0"/>
  <xs:choice minOccurs="0" maxOccurs="unbounded">
   <xs:element ref="ObjectProperty"/>
   <xs:element ref="ObjectPropertyGroup"/>
  </xs:choice>
 </xs:sequence>
 <xs:attribute name="objectDefinitionURI" type="adsml:URIType"
use="required"/>
 <xs:attribute name="parentObjectDefinitionURIRef" type="adsml:URIType"
use="optional"/>
 <xs:attribute ref="xml:base" use="optional"/>
 <xs:attribute ref="xml:lang" use="optional"/>
</xs:complexType>
```

### 4.2.1 ObjectDefinitionHeader element

The `ObjectDefinitionHeader` element provides header-level metadata for an object definition. It extends the content model of the `HeaderType` (required `InternalName` element, optional `DisplayName` and `Description` elements, and optional *`issuedBy`* attribute) to add optional attributes for recording metadata about the 'rule set' from which the object definition was created.

A *rulesetVersion* is used to record a version number for the object definition's rule set. Note that for the first version of a ruleset the value of this attribute will be '1.0' by default and that this value will augment with successive versions of the ruleset.

The value of the *rulesetVersion* attribute should be augmented if successive versions of an object definition are published. The value is recorded as a ShortStringType data type defined in the AdsML Type Library.

A *rulesetUniqueID* attribute can be used to record a unique id for the object definition's rule set. The value is recorded as a ShortStringType data type defined in the AdsML Type Library.

A *rulesetIssuedDateTime* attribute can be used to record the date and time at which the object definition's rule set was issued. The value is recorded as a DateTimeType data type defined in the AdsML Type Library.The *issuedBy* attribute inherited from the HeaderType can be used to identify the party who issued the object definition's rule set.

```
<xs:element name="ObjectDefinitionHeader"
type="ObjectDefinitionHeaderType"/>

<xs:complexType name="ObjectDefinitionHeaderType">
 <xs:complexContent>
  <xs:extension base="HeaderType">
   <xs:attribute name="rulesetVersion" type="adsml:ShortStringType"/>
   <xs:attribute name="rulesetUniqueID" type="adsml:ShortStringType"/>
   <xs:attribute name="rulesetIssuedDateTime" type="adsml:DateTimeType"/>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
```

### 4.2.2 ObjectProperty element

The ObjectProperty element defines a structured description of a characteristic of an object (i.e. it records an 'attribute' of an 'entity' in the real world using a combination of attribute and element data content.

The ObjectProperty is essentially defined using attributes defined in the *ObjectPropertyAttributes* attribute group, the attributes specifying the data and any constraints that apply to that data. Descriptive information about an ObjectProperty can be recorded using an optional Description element. An ObjectProperty can be 'modified' or 'qualified' by using ObjectPropertyModifier element(s). An ObjectPropertyModifier is in essence equivalent to an unmodified property and only differs from an ObjectProperty in that it cannot itself be modified.

See definition of *ObjectPropertyAttributes* attribute group and the Description and ObjectPropertyModifier elements.

```
<xs:element name="ObjectProperty" type="ObjectPropertyType"/>

<xs:complexType name="ObjectPropertyType">
 <xs:sequence>
  <xs:element ref="adsml:Description" minOccurs="0"/>
  <xs:element ref="ObjectPropertyModifier" minOccurs="0"
maxOccurs="unbounded"/>
 </xs:sequence>
 <xs:attributeGroup ref="ObjectPropertyAttributes"/>
```

```
</xs:complexType>
```

### 4.2.3 ObjectPropertyModifier element

The `ObjectPropertyModifier` element is used to 'modify' or 'qualify' an `ObjectProperty`. The `ObjectPropertyModifier` is defined using attributes defined in the *`ObjectPropertyAttributes`* attribute group, the attributes specifying the data and any constraints that apply to that data. Descriptive information about the `ObjectPropertyModifier` can be recorded using an optional `Description` element.

See definition of *`ObjectPropertyAttributes`* attribute group and the `Description` element.

```
<xs:element name="ObjectPropertyModifier"
type="ObjectPropertyModifierType"/>

<xs:complexType name="ObjectPropertyModifierType">
 <xs:sequence>
  <xs:element ref="adsml:Description" minOccurs="0"/>
 </xs:sequence>
 <xs:attributeGroup ref="ObjectPropertyAttributes"/>
</xs:complexType>
```

### 4.2.4 ObjectPropertyGroup element

The `ObjectPropertyGroup` element defines a structured description of a characteristic of an object that is complex and so requires more than one property to describe it. An `ObjectPropertyGroup` can itself contain an `ObjectPropertyGroup` to allow nested complex properties to be recorded.

The `ObjectPropertyGroup` records property data using a sequence of an optional `Description` element followed by a required and repeatable choice between `ObjectProperty` and `ObjectPropertyGroup` elements. Descriptive information about the `ObjectPropertyGroup` can be recorded using an optional `Description` element. The *`CommonObjectPropertyAttributes`* attribute group records information specific to the `ObjectPropertyGroup` itself. See definitions of *`CommonObjectPropertyAttributes`* attribute group, and the `Description` and `ObjectProperty` elements.

```
<xs:element name="ObjectPropertyGroup" type="ObjectPropertyGroupType"/>

<xs:complexType name="ObjectPropertyGroupType">
 <xs:sequence>
  <xs:element ref="adsml:Description" minOccurs="0"/>
  <xs:choice maxOccurs="unbounded">
   <xs:element ref="ObjectProperty"/>
   <xs:element ref="ObjectPropertyGroup"/>
  </xs:choice>
 </xs:sequence>
 <xs:attributeGroup ref="CommonObjectPropertyAttributes"/>
</xs:complexType>
```

### 4.2.5 CommonObjectPropertyAttributes attribute group

The *CommonObjectPropertyAttributes* attribute group provides a set of attributes for describing characteristics common to the `ObjectProperty`, `ObjectPropertyModifier` and `ObjectPropertyGroup` elements.

The following attributes are defined:

- A required *propertyID* attribute. Records a unique identifier using the `ShortTokenType` data type defined in the AdsML Type Library. The *propertyID* is constrained by the uniqueness constraint defined in Section 4.1 AdsMLAdObjectDefinitions element.
- A required *internalName* attribute. Records an 'internal' name for the property used for internal processing. The internal name is intended to be 'machine-readable' and so does not have to be 'human-readable'. The internal name is recorded as a string using the `ShortStringType` data type defined in the AdsML Type Library.
- An optional *displayName* attribute. If present, records a user-oriented and 'human-readable' name for the property. The *displayName* **SHOULD** take the value of the *internalName* by default unless another value is specified. The display name is recorded using `ShortStringType` data type defined in the AdsML Type Library.
- An optional *usage* attribute. If present, records the specified occurrence of a property in instance structured description data using values defined by the UsageType data type.
- An optional *repeatable* attribute. If present, indicates if the property can appear multiple times in a single structured description instance. The repeatable status is recorded as a Boolean value using the `BooleanType` data type defined in the AdsML Type Library. A value of '`true`' identifies the property as repeatable.

```
<xs:attributeGroup name="CommonObjectPropertyAttributes">
 <xs:attribute name="propertyID" type="adsml:ShortTokenType"
use="required"/>
 <xs:attribute name="internalName" type="adsml:ShortStringType"
use="required"/>
 <xs:attribute name="displayName" type="adsml:ShortStringType"/>
 <xs:attribute name="usage" type="UsageType"/>
 <xs:attribute name="repeatable" type="adsml:BooleanType"/>
</xs:attributeGroup>
```

### 4.2.6 ObjectPropertyAttributes attribute group

The *ObjectPropertyAttributes* attribute group provides a set of attributes for describing characteristics common to the `ObjectProperty` and `ObjectPropertyModifier` elements.

The attribute group references the following attributes:

- The *CommonObjectPropertyAttributes* attribute group.

The attribute group defines the following attributes:

- An optional *dataType* attribute. Identifies the data type of the property using values defined by the DataType data type.
- An optional *value* attribute. If present, records the value of the property using the `StringType` data type defined in the AdsML Type Library. The *value* attribute is only used in the case where an object property will

always take a specific value and so this value must therefore be recorded in the object property definition.

- An optional *defaultValue* attribute. If present, records a default value for the property using the `StringType` data type defined in the AdsML Type Library.
- An optional *allowedValuesURIRef* attribute. If present, identifies a controlled vocabulary or list of allowed values specified for use in that context. The reference is recorded as a URI using the `URIType` data type defined in the AdsML Type Library. The format of the referenced set of allowed values is not constrained and they may take any form – for example, the reference could point to a list of values held in a spreadsheet, a flat file, an AdsML ValueTable.
- An optional *minValue* attribute. If present, identifies the minimum value of a property. Only applies to properties taking a numeric value and is used to specify the minimum inclusive numeric value allowed. The minimum value is recorded as a double using the `DoubleType` data type defined in the AdsML Type Library.
- An optional *maxValue* attribute. If present, identifies the maximum value of a property. Only applies to properties taking a numeric value and is used to specify the maximum inclusive numeric value allowed. The maximum value is recorded as a using the `DoubleType` data type defined in the AdsML Type Library.
- An optional *minLength* attribute. If present, identifies the minimum character length of the value of a property. The minimum length is recorded as an integer using the `IntegerType` data type defined in the AdsML Type Library.
- An optional *maxLength* attribute. If present, identifies the maximum character length of the value of a property. The maximum length is recorded as an integer using the `IntegerType` data type defined in the AdsML Type Library.
- An optional *formatMask* attribute. If present, specifies the allowed 'format' or 'mask' of the value of a property. This may be specified as a regular expression pattern using the form defined by the W3C[3] in the XML Schema Part 2: Datatypes specification, or can specify the format mask as an unconstrained string. The format mask is recorded using using the `StringType` data type defined in the AdsML Type Library.
- An optional *searchable* attribute. If present, indicates if the property is identified as a 'searchable' property that is suitable for search and indexing purposes. The searchable status is recorded as a Boolean value using the `BooleanType` data type defined in the AdsML Type Library. A value of '`true`' identifies the property as searchable.
- An optional *publishable* attribute. If present, indicates if the property can be publicly published or is for internal use only. The publishable status is recorded as a Boolean value using the `BooleanType` data type defined in the AdsML Type Library. A value of '`true`' identifies the property as publishable. A value of '`false`' means that the attribute is for internal use only.

```
<xs:attributeGroup name="ObjectPropertyAttributes">
 <xs:attributeGroup ref="CommonObjectPropertyAttributes"/>
 <xs:attribute name="dataType" type="DataType"/>
 <xs:attribute name="value" type="adsml:StringType"/>
 <xs:attribute name="defaultValue" type="adsml:StringType"/>
```

[3] See the W3C specification *XML Schema Part 2: Data types*, Appendix F Regular Expressions (http://www.w3.org/TR/xmlschema-2/#regexs).

```
 <xs:attribute name="allowedValuesURIRef" type="adsml:URIType"/>
 <xs:attribute name="minValue" type="adsml:DoubleType"/>
 <xs:attribute name="maxValue" type="adsml:DoubleType"/>
 <xs:attribute name="minLength" type="adsml:IntegerType"/>
 <xs:attribute name="maxLength" type="adsml:IntegerType"/>
 <xs:attribute name="formatMask" type="adsml:StringType"/>
 <xs:attribute name="searchable" type="adsml:BooleanType" />
 <xs:attribute name="publishable" type="adsml:BooleanType" />
</xs:attributeGroup>
```

## 4.3 ValueTables element

The `ValueTables` element is a container for grouping one or more `ValueTable`(s) that contain allowed values specified for use in an ad object definitions file.

```
<xs:element name="ValueTables" type="ValueTablesType"/>

<xs:complexType name="ValueTablesType">
 <xs:sequence>
  <xs:element ref="ValueTable" maxOccurs="unbounded"/>
 </xs:sequence>
</xs:complexType>
```

### 4.3.1 ValueTable element

The `ValueTable` element provides a structure for recording lists of allowed values in a structured form. It enables those values to be referenced from the object property definitions where they have been specified for use and from the structured description instances where they are being used.

A required `ValueTableHeader` element provides header-level metadata for the value table.

The individual values and their associated definitions that constitute the set of values in a value table are recorded as entries in the value table using individual `Entry` children element(s).

A required *`valueTableURI`* attribute records an identifier for the value table in the form of a URI using the `URIType` data type defined in the AdsML Type Library. The *`valueTableURI`* provides a machine-readable and externally-addressable unique id for the value table, enabling the value table to be referenced by the *`allowedValuesURIRef`* attribute in an object definition or a structured description instance.

An optional *`xml:base`* attribute allows a base URI to be set for the `ValueTable` element if needed.

An optional *`xml:lang`* attribute allows the human language used in the value table to be identified.

```
<xs:element name="ValueTable" type="ValueTableType"/>

<xs:complexType name="ValueTableType">
 <xs:sequence>
  <xs:element ref="ValueTableHeader"/>
  <xs:element ref="Entry" maxOccurs="unbounded"/>
 </xs:sequence>
 <xs:attribute name="valueTableURI" type="adsml:URIType" use="required"/>
 <xs:attribute ref="xml:base" use="optional"/>
 <xs:attribute ref="xml:lang" use="optional"/>
</xs:complexType>
```

#### 4.3.1.1 ValueTableHeader element

The `ValueTableHeader` element provides header-level metadata for a value table. It extends the content model of the `HeaderType` (required `InternalName` element, optional `DisplayName` and `Description` elements, and optional *`issuedBy`* attribute) to add an optional *`valueTableVersion`* attribute.

The *`valueTableVersion`* attribute is used to record a version number for the value table to facilitate versioning of value tables. The *`valueTableVersion`* attribute is recorded as the `ShortStringType` data type defined in the AdsML Type Library. Note that for the first version of a ruleset the value of this attribute will be '`1.0`' by default and that this value will augment with successive versions of a value table.

```
<xs:element name="ValueTableHeader" type="ValueTableHeaderType"/>

<xs:complexType name="ValueTableHeaderType">
 <xs:complexContent>
  <xs:extension base="HeaderType">
   <xs:attribute name="valueTableVersion" type="adsml:ShortStringType" />
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
```

#### 4.3.1.2 Entry element

The `Entry` element describes and records a specific value. A required `Description` element describes the meaning of the value. The value itself is recorded as a string using a `Value` element. The `Value` element is required and repeatable to allow different lexical representations of a single value to be recorded.

An optional *`id`* attribute records can be used to record an identifier for the `Entry` element as an `xs:ID` data type using the `IDType` data type defined in the AdsML Type Library.

```
<xs:element name="Entry" type="EntryType"/>

<xs:complexType name="EntryType">
 <xs:sequence>
  <xs:element ref="Description"/>
  <xs:element ref="Value" maxOccurs="unbounded"/>
 </xs:sequence>
 <xs:attribute name="id" type="adsml:IDType"/>
</xs:complexType>
```

##### 4.3.1.2.1 Value element

The `Value` element records a value as a string using the `ValueTextType`.

The `ValueTextType` type extends the `LongStringType` data type defined in the AdsML Type Library to add optional *`id`*, *`preferredName`*, *`codeAuthority`*, *`codeSchemeName`*, *`codeSchemeVersion`*, and *`codeDisplayName`* attributes.

The *`id`* attribute records an identifier for the `Value` element as an `xs:ID` data type using the `IDType` data type defined in the AdsML Type Library. The *`id`* attribute provides a unique id for the `Value` element that can be used as a fragment identifier, enabling the value to be referenced externally.

The *`preferredName`* attribute is of `BooleanType` data type defined in the AdsML Type Library. The *`preferredName`* attribute identifies whether a name is the preferred name in the event that multiple names are present. A value of '`true`' identifies the value as the preferred name for that entry.

The *`codeAuthority`* attribute is of `ShortStringType` data type defined in the AdsML Type Library and identifies the party or body that created and manages the code value.

The *`codeSchemeName`* attribute is of `ShortStringType` data type defined in the AdsML Type Library and identifies the name of the particular code scheme to which the code value belongs and by which the code gains its meaning.

The *`codeSchemeVersion`* attribute is of `ShortStringType` data type defined in the AdsML Type Library and identifies the version of the code scheme that is being used.

The *`codeDisplayName`* attribute is of `ShortStringType` data type defined in the AdsML Type Library and can be used to record a human-readable rendition of the code value for display purposes if desired.

```
<xs:element name="Value" type="ValueTextType"/>

<xs:complexType name="ValueTextType">
 <xs:simpleContent>
  <xs:extension base="LongString">
   <xs:attribute name="id" type="adsml:IDType"/>
   <xs:attribute name="preferredName" type="adsml:BooleanType"/>
   <xs:attribute name="codeAuthority" type="adsml:ShortStringType"/>
   <xs:attribute name="codeSchemeName" type="adsml:ShortStringType"/>
   <xs:attribute name="codeSchemeVersion" type="adsml:ShortStringType"/>
   <xs:attribute name="codeDisplayName" type="adsml:ShortStringType"/>
  </xs:extension>
 </xs:simpleContent>
</xs:complexType>
```

# 5 Structured Descriptions schema vocabulary

The elements and attributes of the structured descriptions schema are defined below.

## 5.1 AdsMLStructuredDescriptions element

The `AdsMLStructuredDescriptions` element is the root element of an AdsML structured descriptions instance document and provides a container for individual structured description(s) of ad objects. Uniqueness constraints are specified in the `AdsMLStructuredDescriptions` element to ensure that,

- The value of the *`objectDescriptionID`* attribute of the `AdObjectDescription` element is unique amongst all instances of that element within the `AdsMLStructuredDescriptions` file.
- The value of the *`propertyID`* attribute of the `Property`, `PropertyGroup`, or `PropertyModifier` elements is unique amongst all instances of those elements within the `AdsMLStructuredDescriptions` file.

```
<xs:element name="AdsMLStructuredDescriptions"
type="StructuredDescriptionsType">
 <xs:unique name="adsMLobjectDescriptionIDUniqueConstraint">
  <xs:selector xpath="adsml-sd:AdObjectDescription"/>
  <xs:field xpath="@objectDescriptionID"/>
 </xs:unique>
 <xs:unique name="advertisedItemPropertyIDUniqueConstraint">
  <xs:selector xpath=".//adsml-sd:Property|.//adsml-
sd:PropertyGroup|.//adsml-sd:PropertyModifier"/>
  <xs:field xpath="@propertyID"/>
 </xs:unique>
</xs:element>
```

The `AdsMLStructuredDescriptions` element enables a standalone instance document of structured descriptions to be created should this be required. In most use cases, structured descriptions data is likely to be embedded in other instance documents, either `by` referencing the `StructuredDescriptions` element or by locally creating a new element declared as the `StructuredDescriptionsType`, both of which are defined in the AdsMLStructuredDescriptions Public Type Library. It is in this mode that the other AdsML standards use AdsMLStructuredDescriptions in their schema, importing the AdsMLStructuredDescriptions Public Type Library. Note that when reusing the `StructuredDescriptionsType` it is **RECOMMENDED** that any element declared as the `StructuredDescriptionsType` is named '`StructuredDescriptions`'.

For the definition of the `StructuredDescriptions` element and the `StructuredDescriptionsType` see Section 5.2 StructuredDescriptions element.

## 5.2 StructuredDescriptions element

The `StructuredDescriptions` element is the root element of structured descriptions instance data and provides a container for individual structured description(s) of ad objects. Uniqueness constraints are specified in the `StructuredDescriptions` element to ensure that,

- The value of the *`objectDescriptionID`* attribute of the `AdObjectDescription` element is unique amongst all instances of that element within the `StructuredDescriptions` instance data.
- The value of the *`propertyID`* attribute of the `Property`, `PropertyGroup`, or `PropertyModifier` elements is unique amongst all instances of those elements within the `StructuredDescriptions` instance data.

```
<xs:element name="StructuredDescriptions" type="StructuredDescriptionsType">
 <xs:unique name="objectDescriptionIDUniquenessConstraint">
  <xs:selector xpath="adsml-sd:AdObjectDescription"/>
  <xs:field xpath="@objectDescriptionID"/>
 </xs:unique>
 <xs:unique name="advertisedItemPropertyIDUniquenessConstraint">
  <xs:selector xpath=".//adsml-sd:Property|.//adsml-
sd:PropertyGroup|.//adsml-sd:PropertyModifier"/>
  <xs:field xpath="@propertyID"/>
 </xs:unique>
</xs:element>
```

The `StructuredDescriptions` element contains a sequence of optional and repeatable `IndustryCodeSet` and `AdObjectDescription` element(s).

The `IndustryCodeSet` element(s) record an industry classification of the advertisement content.

An `AdObjectDescription` element is used to record a structured description of each individual advertisement object in the advertisement content, allowing for the scenario where advertisement content can consist of multiple 'advertisement object'(s).

The optional and multiple occurrence of `IndustryCodeSet` and `AdObjectDescription` element(s) allows a structured descriptions instance to provide basic or sophisticated levels of metadata about the ad content that it describes. At a basic level, a structured descriptions instance could contain no more than an industry classification of ad content using a single `IndustryCodeSet` element. In a more sophisticated scenario, composite ads consisting of more than one 'ad object' can be classified and described using a combination of `IndustryCodeSet` and `AdObjectDescription` element(s).

```
<xs:complexType name="StructuredDescriptionsType">
 <xs:sequence>
  <xs:element ref="IndustryCodeSet" minOccurs="0" maxOccurs="unbounded"/>
  <xs:element ref="AdObjectDescription" minOccurs="0"
maxOccurs="unbounded"/>
 </xs:sequence>
</xs:complexType>
```

## 5.3 IndustryCodeSet element

The `IndustryCodeSet` element records an industry classification of ad content.

A required `IndustryCode` element records an industry classification of the ad content described by the structured descriptions instance. See definition of `IndustryCode` element for more information about industry code.

An optional `PercentageAllocation` element can be used to indicate the percentage weighting of the advertisement content that is classified by the `IndustryCode` element. The percentage weighting is recorded as an integer value using the `IntegerType` data type defined in the AdsML Type Library.

If present, an optional *`objectDescriptionIDRef`* attribute of `ShortTokenType` data type defined in the AdsML Type Library allows an `IndustryCodeSet` element to be associated with an individual ad object in a structured descriptons instance by referencing that `AdObjectDescription` element's *`objectDescriptionID`* attribute.

```
<xs:element name="IndustryCodeSet" type="IndustryCodeSetType"/>

<xs:complexType name="IndustryCodeSetType">
 <xs:sequence>
  <xs:element ref="IndustryCode"/>
  <xs:element ref="PercentageAllocation" minOccurs="0"/>
 </xs:sequence>
 <xs:attribute name="objectDescriptionIDRef" type="adsml:ShortTokenType"/>
</xs:complexType>

<xs:element name="PercentageAllocation" type="adsml:IntegerType"/>
```

## 5.4 AdObjectDescription element

The `AdObjectDescription` element records a structured description of the characteristics of an advertisement object as simple name-value pairs using a required and repeatable choice between `Property` and `PropertyGroup` element(s).

A required *`objectDescriptionID`* attribute records an identifier for the `AdObjectDescription` using the `ShortTokenType` data type defined in the AdsML Type Library.

A required *`objectDefinitionURIRef`* attribute is used to associate the structured description instance with the object definition that defines the set of properties used to describe that type of ad object. The *`objectDefinitionURIRef`* makes the association in the form of a URI using the `URIType` data type defined in the AdsML Type Library, identifying the object definition by referencing its *`objectDefinitionURI`* attribute.

```
<xs:element name="AdObjectDescription" type="AdObjectDescriptionType"/>

<xs:complexType name="AdObjectDescriptionType">
 <xs:choice maxOccurs="unbounded">
  <xs:element ref="Property"/>
  <xs:element ref="PropertyGroup"/>
 </xs:choice>
 <xs:attribute name="objectDescriptionID" type="adsml:ShortTokenType"
use="required"/>
 <xs:attribute name="objectDefinitionURIRef" type="adsml:URIType"
use="required"/>
</xs:complexType>
```

### 5.4.1 Property element

The `Property` element records a structured description of a characteristic of an advertisement object using a combination of attribute and element data content.

The `Property`’s data is recorded using attributes defined in the *`PropertyAttributes`* attribute group.

If present, optional and repeatable `PropertyModifier` children element(s) record data that ‘modifies’ or ‘qualifies’ the `Property` in which they appear.

See definition of the *`PropertyAttributes`* attribute group and the `PropertyModifier` element.

```
<xs:element name="Property" type="PropertyType"/>

<xs:complexType name="PropertyType">
 <xs:sequence>
  <xs:element ref="PropertyModifier" minOccurs="0" maxOccurs="unbounded"/>
 </xs:sequence>
 <xs:attributeGroup ref="PropertyAttributes"/>
</xs:complexType>
```

### 5.4.2 PropertyModifier element

The `PropertyModifier` element is used to record data that ‘modifies’ or ‘qualifies’ the `Property` element in which it occurs using attributes defined in the *`PropertyAttributes`* attribute group.

See definition of the *`PropertyAttributes`* attribute group.

```
<xs:element name="PropertyModifier" type="PropertyModifierType"/>
```

```
<xs:complexType name="PropertyModifierType">
 <xs:attributeGroup ref="PropertyAttributes"/>
</xs:complexType>
```

### 5.4.3 PropertyGroup element

The `PropertyGroup` element is used to record a structured description of a characteristic of an advertisement object when that characteristic requires more than one property to describe it. A `PropertyGroup` can itself contain a `PropertyGroup` to allow nested complex properties to be recorded.

The `PropertyGroup` records data using a required and repeatable choice between `Property` and `PropertyGroup` elements. The *`CommonPropertyAttributes`* attribute group records information specific to the `PropertyGroup` itself.

See definition of the *`CommonPropertyAttributes`* attribute group and the `Property` element.

```
<xs:element name="PropertyGroup" type="PropertyGroupType"/>

<xs:complexType name="PropertyGroupType">
 <xs:choice maxOccurs="unbounded">
  <xs:element ref="Property"/>
  <xs:element ref="PropertyGroup"/>
 </xs:choice>
 <xs:attributeGroup ref="CommonPropertyAttributes"/>
</xs:complexType>
```

### 5.4.4 CommonPropertyAttributes attribute group

The *`CommonPropertyAttributes`* attribute group provides a set of attributes for describing characteristics common to the `Property`, `PropertyModifier` and `PropertyGroup` elements.

The following attributes are defined:

- A required *`propertyID`* attribute. Records a unique identifier using the `ShortTokenType` data type defined in the AdsML Type Library. The *`propertyID`* is constrained by the uniqueness constraint defined in Section 5.2 StructuredDescriptions element.
- A required *`internalName`* attribute. Records an 'internal' name for the property used for processing. The internal name is intended to be 'machine-readable' and so does not have to be 'human-readable'. The internal name is recorded using the `ShortStringType` data type defined in the AdsML Type Library.
- An optional *`displayName`* attribute. If present, records an optional user-oriented and 'human-readable' name for the property. The display name is recorded using the `ShortStringType` data type defined in the AdsML Type.

```
<xs:attributeGroup name="CommonPropertyAttributes">
 <xs:attribute name="propertyID" type="adsml:ShortTokenType"
use="required"/>
 <xs:attribute name="internalName" type="adsml:ShortStringType"
use="required"/>
 <xs:attribute name="displayName" type="adsml:ShortStringType"/>
</xs:attributeGroup>
```

### 5.4.5 PropertyAttributes attribute group

The *`PropertyAttributes`* attribute group provides a set of attributes for recording property data common to the `Property` and `PropertyModifier` elements.

The attribute group references the following attributes:

- The *`CommonPropertyAttributes`* attribute group.

The attribute group defines the following attribute:

- A required *`value`* attribute. Records the value of the property using the `StringType` data type defined in the AdsML Type Library.

```
<xs:attributeGroup name="PropertyAttributes">
 <xs:attributeGroup ref="CommonPropertyAttributes"/>
 <xs:attribute name="value" type="adsml:StringType" use="required"/>
</xs:attributeGroup>
```

# 6 Common component listing

This section lists the element, attribute and type definitions that are reused across both the *advertisement object definitions* and the *structured descriptions* schema structures.

## 6.1 Common content model definitions

The following common content model definitions are defined or reused.

### 6.1.1 Description element

The `Description` element records a description as a string. The `Description` element is reused from the AdsML Type Library.

### 6.1.2 DisplayName element

The `DisplayName` element records a human-readable name intended for display to humans. By default, the `DisplayName` **SHOULD** take the value of the `InternalName` unless specified. The name is recorded as a string using the `LongStringType` data type defined in the AdsML Type Library.

```
<xs:element name="DisplayName" type="adsml:LongStringType"/>
```

### 6.1.3 IndustryCode element

The `IndustryCode` element records an industry classification of the content of an advertisement object using the `CodeType` defined in the AdsML Type Library.

As with all contexts where a code type is used, a controlled vocabulary of specific values can be used if required. Refer to the '*AdsML Type Library*' document for more information about the `adsml:CodeType`, and to the '*AdsML Controlled Vocabularies*' document for more information about the use of controlled vocabularies in AdsML. (See Section 1.7 Accompanying documents.)

```
<xs:element name="IndustryCode" type="adsml:CodeType"/>
```

### 6.1.4 InternalName element

The `InternalName` element records a machine-readable name that is not intended for display to users. `InternalName`'s must be unique within the set of sibling properties, property groups, or property modifiers in which they appear. The internal name is recorded using the `LongStringType` data type defined in the AdsML Type Library.

```
<xs:element name="InternalName" type="adsml:LongStringType"/>
```

### 6.1.5 HeaderType type

The `HeaderType` type provides header-level metadata about the element structure in which it appears using a sequence of a required `InternalName` element followed by optional `DisplayName` and `Description` elements. An optional *`issuedBy`* attribute of `ShortStringType` data type defined in the AdsML Type Library. The *`issuedBy`* attribute identifies the organization issuing or publishing the object definition or value table in which the `HeaderType` is being used. Note that the publisher may or may not be the creator of the published content.

```
<xs:complexType name="HeaderType">
 <xs:sequence>
  <xs:element ref="InternalName"/>
  <xs:element ref="DisplayName" minOccurs="0"/>
  <xs:element ref="adsml:Description" minOccurs="0"/>
 </xs:sequence>
 <xs:attribute name="issuedBy" type="adsml:ShortStringType"/>
</xs:complexType>
```

### 6.1.6 xml:base attribute

The *`xml:base`* attribute is used to define base URIs for parts of XML documents as defined by the XML Base specification. The *`xml:base`* attribute provides a base URI for interpreting any relative URIs that appear in the scope of the element in which it appears. The *`xml:base`* attribute is defined in The "xml" Namespace by the XML Base specification. The *`xml:base`* records its value as an `xs:anyURI` data type.

```
<xs:schema targetNamespace="http://www.w3.org/XML/1998/namespace"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xml:lang="en"
elementFormDefault="unqualified" attributeFormDefault="unqualified">

<xs:attribute name="base" type="xs:anyURI">
 ...
</xs:attribute>

</xs:schema>
```

### 6.1.7 xml:lang attribute

The *`xml:lang`* attribute identifies the human language used in the scope of the element in which it appears as defined by the XML 1.0 Specification. The *`xml:lang`* attribute is defined in The "xml" Namespace by the XML 1.0 Specification. The *`xml:lang`* records its value as an `xs:language` data type.

```
<xs:schema targetNamespace="http://www.w3.org/XML/1998/namespace"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xml:lang="en"
elementFormDefault="unqualified" attributeFormDefault="unqualified">
```

```
<xs:attribute name="lang" type="xs:language">
 ...
</xs:attribute>

</xs:schema>
```

## 6.2 Common data type definitions

Common type definitions are data types derived for common use in the AdsML Structured Descriptions schemas.

### 6.2.1 Specific enumerated value definitions

Specific value definitions are data types derived to provide lists of predefined enumerated values for use in the AdsML Structured Descriptions schema.

#### 6.2.1.1 DataType

Defines a list of data types used to specify the data type of a property or property modifier. `DataType` is derived from `ShortTokenType` data type defined in the AdsML Type Library and used in the *`dataType`* attribute context of the `ObjectProperty` and `ObjectPropertyModifier` elements.

| **DataType** | |
|---|---|
| **Code** | **Definition** |
| Text | A data type used to record a string value. Corresponds to `xs:string` (equivalent to `adsml:StringType`). |
| Value | A data type used to record a specific value recorded in a value table. Corresponds to `xs:string` (equivalent to `adsml:StringType`). |
| Decimal | A data type used to record a numeric value. Corresponds to `xs:decimal`. |
| Integer | A data type used to record a whole numeric value. Corresponds to `xs:integer` (equivalent to `adsml:IntegerType`). |
| Boolean | A data type used to record a Boolean ‘true’ or ‘false’ value. Corresponds to `xs:boolean` (equivalent to `adsml:BooleanType`). |
| DateTime | A data type used to record date and time. Corresponds to `xs:dateTime` (equivalent to `adsml:DateTimeType`). |
| Date | A data type used to record date. Corresponds to `xs:date` (equivalent to `adsml:DateType`). |
| Time | A data type used to record time. Corresponds to `xs:time`. |

#### 6.2.1.2 UsageType

Defines a list of usage types for specifying the usage of a property or property modifier. `DataType` is derived from `ShortTokenType` data type defined in the AdsML Type Library and used in the *`usage`* attribute context of the `Property` and `PropertyModifier` elements.

| **UsageType** | |
|---|---|
| **Code** | **Definition** |
| Required | Indicates that the presence of a property or property modifier is required in an instance structured description. |
| Optional | Indicates that the presence of a property or property modifier is optional in an instance structured description. |
| Encouraged | Indicates that the presence of a property or property modifier is encouraged but not required in an instance structured description. |

### 6.2.2 Root type definitions

AdsML Structured Descriptions does not define any root types.

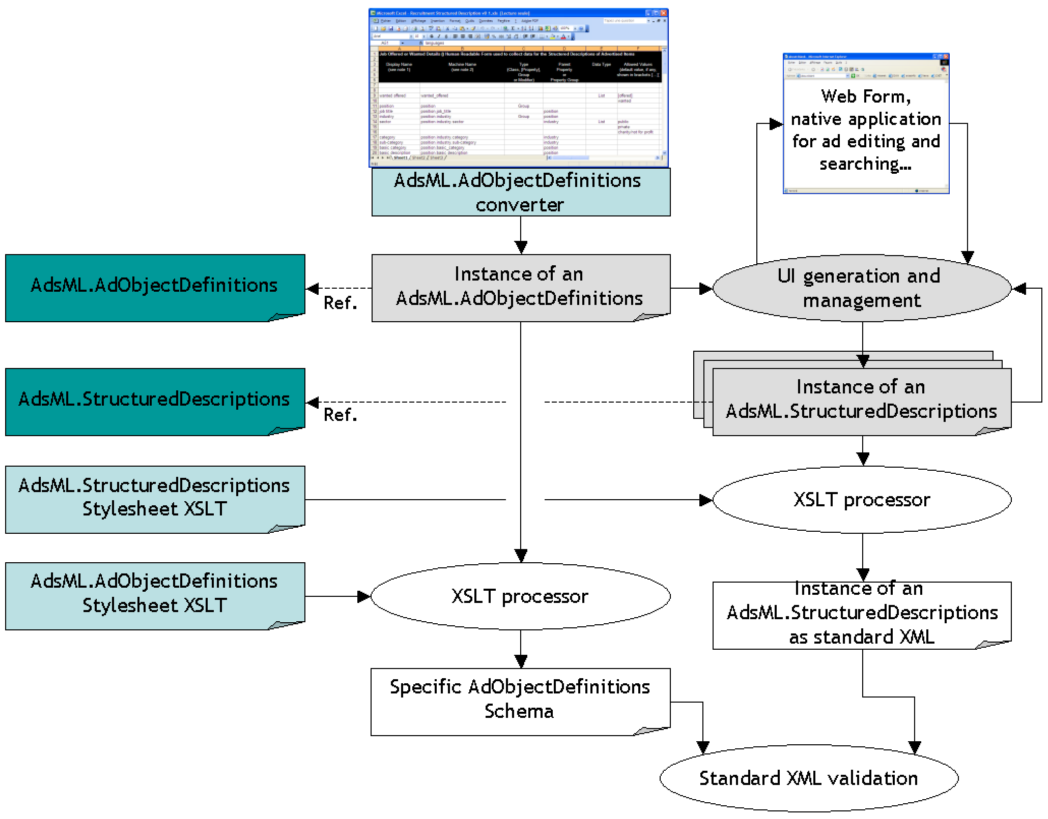
# 7 Creating & maintaining user-defined rule sets for advertisement object definitions

User-defined rule sets for advertisement object definitions are recorded as rule sets. The process for creating and maintaining them is specified in the rule set spreadsheet and its accompanying documentation. For an overview of the process see the '*AdsML Structured Descriptions of Advertisement Objects 1.0 Part 1 Usage Rules & Guidelines*' document. (See Section 1.7 Accompanying documents.)

Note that the form in which trading partners create, maintain, exchange or store rule sets for advertisment object definitions is not prescribed by this specification although it is **RECOMMENDED** that the rule set spreadsheet made available by AdsML for this purpose **SHOULD** be used as described in its accompanying documentation. When using a rule set in AdsML, then to be AdsML schema compliant the rule set **MUST** be represented as an XML instance that conforms to the *advertisement object definitions* schema in order to allow instance structured descriptions data represented using the *structured descriptions* schema to reference the relevant advertisement object definition.

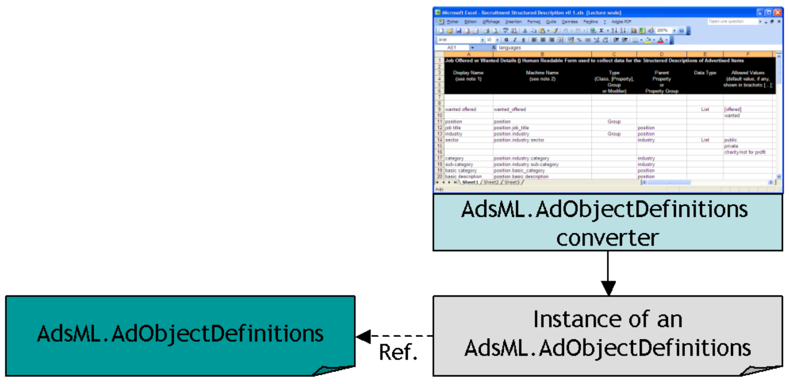
# 8 Model Technical Architecture

This section illustrates a model technical architecture for how AdsML Structured Descriptions can be implemented for use, walking through the process by which the information entered in a rule set is transformed into the XML documents used to assist in the process of capturing, transmitting and validating structured descriptions of advertisement objects. It shows the relationships between the main components of structured descriptions, including both those defined in the AdsML Structured Definitions standard, and those that will be provided by implementers. The following sections then walk through five technical scenarios, highlighting and discussing the portions of this illustration that are relevant to each scenario.


AdsML.AdObjectDefinitions converter
Web Form, native application for ad editing and searching...
AdsML.AdObjectDefinitions
Ref.
Instance of an AdsML.AdObjectDefinitions
UI generation and management
AdsML.StructuredDescriptions
Ref.
Instance of an AdsML.StructuredDescriptions
AdsML.StructuredDescriptions Stylesheet XSLT
XSLT processor
AdsML.AdObjectDefinitions Stylesheet XSLT
XSLT processor
Instance of an AdsML.StructuredDescriptions as standard XML
Specific AdObjectDefinitions Schema
Standard XML validation

## 8.1 Architectural views

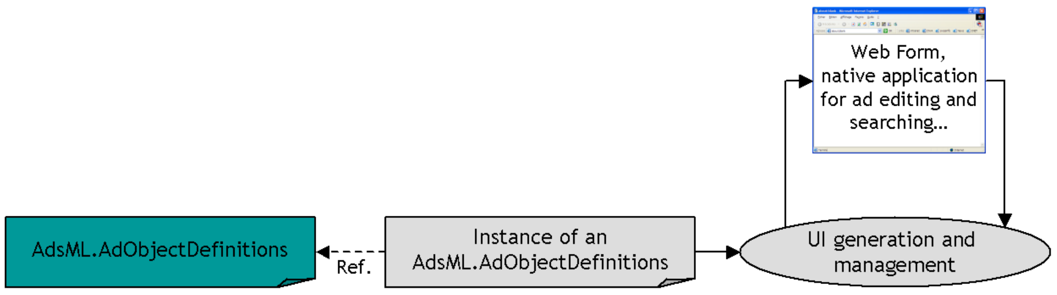
### 8.1.1 User's view


AdsML.AdObjectDefinitions converter
AdsML.AdObjectDefinitions
Ref.
Instance of an AdsML.AdObjectDefinitions

A content specialist defines the data structures and business rules that will be exchanged between these systems using the Rule Set spreadsheet interface that comes from AdsML. The spreadsheet is then converted into an AdsMLAdObjectDefinitions instance, which conforms to the structure defined in the AdsMLAdObjectDefinitions schema.

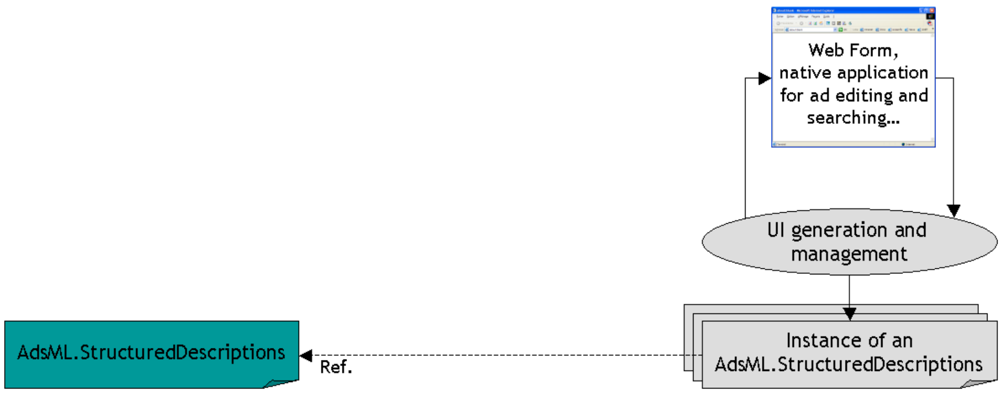
Note that more than one spreadsheet can be defined, if more than one type of advertising information is going to be exchanged between the systems and it is considered appropriate to divide the data definitions into multiple rule sets. Alternatively, a single heterogeneous rule set could be created that contains many different types of information, each on its own page.

### 8.1.2 Developer's view


Web Form, native application for ad editing and searching...
AdsML.AdObjectDefinitions
Ref.
Instance of an AdsML.AdObjectDefinitions
UI generation and management

Based on the definitions in the AdsMLAdObjectDefinitions instance, an application developer can implement a generic or a specific UI for ad editing, searching and displaying.

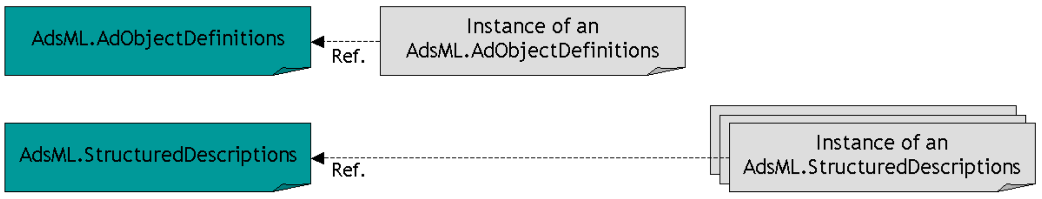
### 8.1.3 Advertiser's view, Consumer's view


Web Form, native application for ad editing and searching...
UI generation and management
AdsML.StructuredDescriptions
Ref.
Instance of an AdsML.StructuredDescriptions

The Advertiser is able to create ads. The Consumer is able to browse or query online ads in order to shop for goods or services.

When placing an ad, the Advertiser will create/edit one AdsMLStructuredDescriptions instance per ad object, using attributes defined in the ad object definition. When looking at an ad, the end user (consumer) would search and browse online ads using those same attributes.

### 8.1.4 Content syndication view


AdsML.AdObjectDefinitions
Ref.
Instance of an AdsML.AdObjectDefinitions
AdsML.StructuredDescriptions
Ref.
Instance of an AdsML.StructuredDescriptions

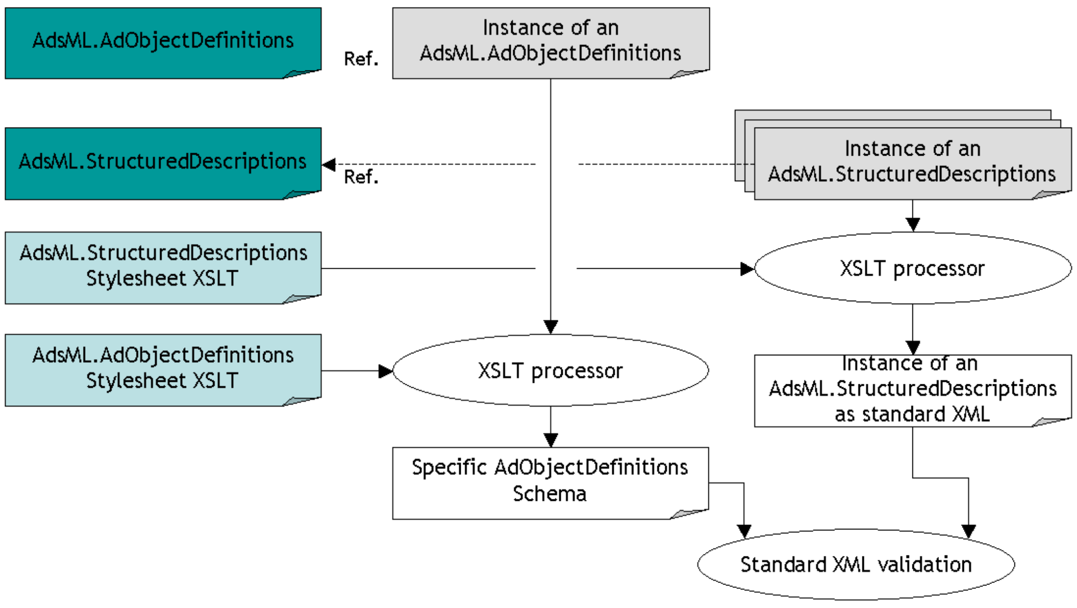
Two sites that agree to exchange content, should:

- Exchange AdsMLAdObjectDefinitions instances for each of the planned advertising categories. This is their formal confirmation and definition of the information formats they will use.

- Exchange the agreed ad instances in the form of AdsMLStructuredDescriptions files that conform to the Ad Object Definitions they previously exchanged.

Note that a single Structured Description message can contain information that is defined in several Ad Object Definitions files.

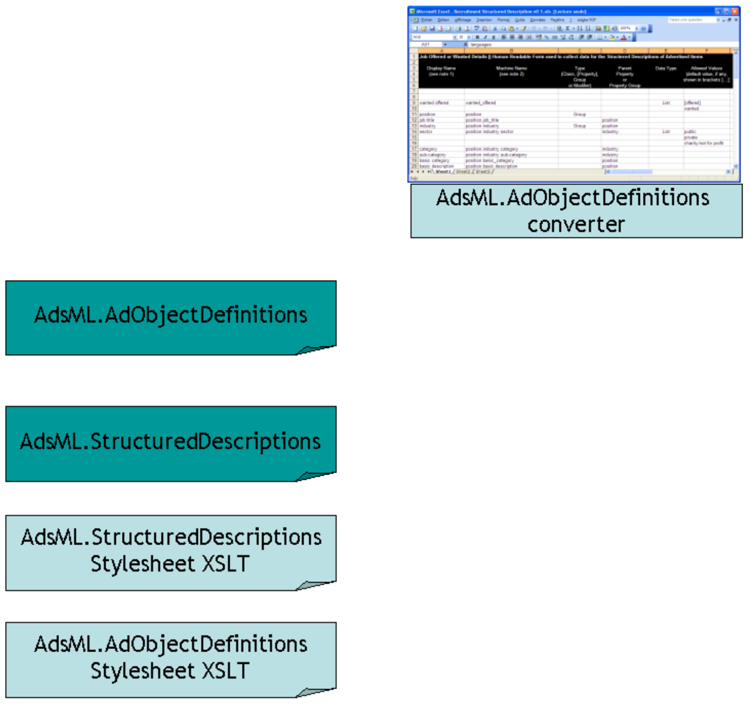
### 8.1.5 Content syndication validation view


AdsML.AdObjectDefinitions
Ref.
Instance of an AdsML.AdObjectDefinitions
AdsML.StructuredDescriptions
Ref.
Instance of an AdsML.StructuredDescriptions
AdsML.StructuredDescriptions Stylesheet XSLT
XSLT processor
AdsML.AdObjectDefinitions Stylesheet XSLT
XSLT processor
Instance of an AdsML.StructuredDescriptions as standard XML
Specific AdObjectDefinitions Schema
Standard XML validation

In addition to the previous syndication scenario, two sites doing syndication could also validate their content using AdsML-provided XSLT, by converting the structured descriptions instances into an XML format that can be validated using a validating parser, and converting the object definitions files into a schema that can perform the necessary validations.

## 8.2 Artifact list

As can be seen in the previous views, the AdsML Structured Descriptions standard defines or provides the following artifacts:


AdsML.AdObjectDefinitions converter
AdsML.AdObjectDefinitions
AdsML.StructuredDescriptions
AdsML.StructuredDescriptions Stylesheet XSLT
AdsML.AdObjectDefinitions Stylesheet XSLT

- AdsML Structured Descriptions **Rule Set** data entry interface – for example, the spreadsheet described above.
- AdsML **Ad Object Definitions Converter** – a tool for converting the rules defined in an AdsML Rule Set spreadsheet into the Ad Object Definitions format.
- AdsML **Ad Object Definitions** (AdsMLAdObjectDefinitions) – an XML format for representing a Rule Set. This is a compact XML structure that can express all of the information content of a Rule Set spreadsheet, plus a little more. The standard provides a schema for this format, and in practice, each rule set will be stored as an XML instance conforming to this schema.
    - NOTE: The extra capabilities that can be expressed in this format but not in the spreadsheet version of the rule set are relatively minor and technical, and will not be required in most markets.
    - NOTE: The format is optimized to express the rule set in a compact form that you can "see at a glance".
- AdsML **Structured Descriptions** (AdsMLStructuredDescriptions) – an XML format for representing the objects in a specific advertisement according to the rules specified in one or more AdsML Ad Object Definitions. Again, this is a compact XML structure that is optimized for processing by generic tools. The standard provides a schema for this format, and in practice, each set of structured descriptions about a given advertisement will be conveyed in an XML instance that conforms to this schema.
- AdsML **Structured Descriptions Validation Stylesheet** (AdsMLStructuredDescriptions Stylesheet XSLT) – converts an AdsMLStructuredDescriptions instance into a format that can more easily be validated by an appropriate XML Schema.
- AdsML **Ad Object Definitions Validation Stylesheet** (AdsMLAdObjectDefinitions Stylesheet XSLT) – converts an

AdsMLObjectDefinitions rule set into an XML Schema that can be used to validate an instance message that has been created using the AdsMLStructuredDescriptions Stylesheet mentioned above.

# 9 Generating the XML definition of a rule set from the spreadsheet version

## 9.1 Overview

Any rule set that has been recorded using the AdsML-defined structured descriptions spreadsheet interface can be converted into a formal Ad Objects Definitions XML document without loss of information. In general, the process of mapping the information from the spreadsheet into the XML format is straightforward, once you are familiar with the XML structures:

- Each page of the spreadsheet (other than the initial "header" page) becomes an `<AdObjectDefinition>` element
- Each row of data on that page becomes either an `<ObjectProperty>`, `<ObjectPropertyGroup>` or `<ObjectPropertyModifier>` element
- Most of the columns of data in the spreadsheet become attributes of the XML element in question
- However, if a set of more than one "Allowed Values" have been defined for a given property or modifier, they are recorded as entries in a `<ValueTable>` element at the bottom of the XML structure, which is then referenced from the '*`allowedValuesURIRef`*' attribute of the property or modifier in question.

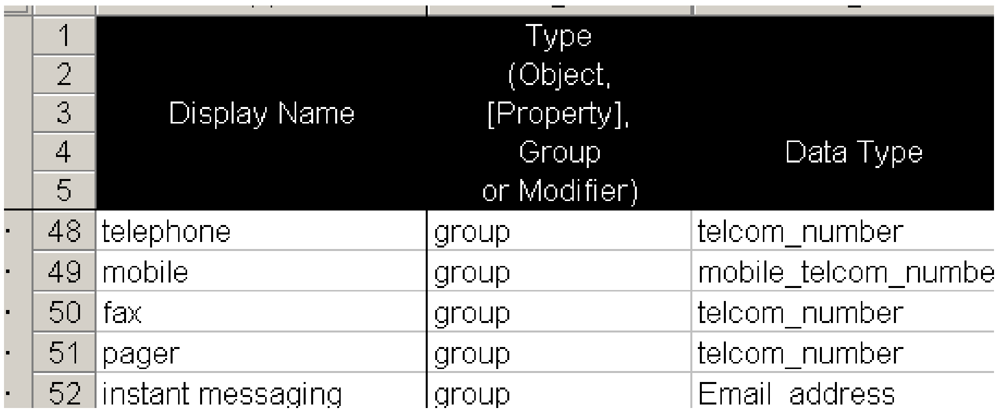
The following sections describe exceptions to the normal practice.

## 9.2 Property re-use

The spreadsheet interface allows for the re-use of property definitions, by creating an empty property group and specifying as its "data type" the name of another page of the spreadsheet where the relevant property definitions can be found.

However, the design of the Ad Objects Definition XML format does not permit this kind of property re-use. Therefore, when generating the XML version of a rule set that contains re-used properties, in each such case the definitions of the re-used properties must be copied into the place where they were referenced, replacing the external reference.

For example, the AdsML model vocabulary for Miscellaneous Goods contains several property groups that re-use the "telcom_number" structure, which is defined on another page:



| | Display Name | Type (Object, [Property], Group or Modifier) | Data Type |
|---|---|---|---|
| 48 | telephone | group | telcom_number |
| 49 | mobile | group | mobile_telcom_numbe |
| 50 | fax | group | telcom_number |
| 51 | pager | group | telcom_number |
| 52 | instant messaging | group | Email address |

Suppose that "telcom_number" contained only two properties: "area_code" and "number". In that case, when the XML for this section was generated, its structure (though not its syntax) would look something like this:

```
When_available (group)
     telephone (group)
       area_code (property)
       number (property
     ...
     fax (group)
       area_code (property)
       number (property)
     pager
       area_code (property)
       number (property)
     etc.
```

As you can see, each property group whose data type was "telcom_number" has now been populated with the property definitions that were found on the referenced page of the spreadsheet. In the XML format, property groups are not allowed to have data types, so the re-use references have disappeared entirely.

## 9.3 Machine names for nested items

In the XML version of a rule set, the machine name for a given property, group or modifier must be unique *within the entire object definition* of which it is a part. No two properties, property groups or even modifiers within an object definition can share the same machine name.

In the spreadsheet version of the rule set, however, the only requirement is that each machine name must be unique *among its immediate siblings*. For example, all of the properties within a given property group must have unique machine names, and all of the modifiers of a given property must have unique machine names, but the same machine name might be used in two different property groups or for two different modifiers. (In fact this will often be the case, since some names such as "name", "size", "currency" and "unit of measure" may appear repeatedly in the rule set.)

In order to ensure that the machine names in the XML version of a rule set are appropriately unique, we recommend the following two principles be applied when converting spreadsheet names into the XML format:

- The machine name for a modifier should always begins with the name of its parent property using 'x.y' notation. For example, the machine name of the "currency" modifier in a spreadsheet might be "currency", but in the

XML version of that rule set it would be expanded to include the name of its parent property, e.g. "price.currency".

- The machine name for a property that is part of a property group always begins with the name of the parent property group. For example, suppose a property group named "Prices" contains a property called "Asking_price". In the spreadsheet, the property's machine name might be just "asking_price", but in the XML version it would have to be expanded to include the parent property group's name, e.g. "prices.asking_price".

These two naming conventions are additive, so that the "currency" modifier of the "asking_price" property in the "prices" property group would be named "prices.asking_price.currency".

# 10 Validation of instance data

As outlined in Section 2.2 AdsML Structured Descriptions overview above, this specification defines a method for expressing an advertisement object definition that defines a set of constraints to which structured description instance data can be validated for conformance.

In principle, the structured description instance data is exchanged as simple name:value pairs in the generic format defined by the *structured descriptions* schema. The structured description instance data is embedded in the file used to exchange the advertisement content described by the structured description data. Recipients of the ad content file are then able to validate the structured description data according to the basic constraints expressed in the *structured descriptions* schema – namely that at a minimum name:value pair(s) are present. The structured description instance data also records a reference to the advertisement object definition that defines the properties of the object described by the instance data. Accordingly, users are able to identify the relevant advertisement object definition and validate the structured description instance data against the constraints specified by that advertisement object definition.

The extent and methods by which any such additional validation is performed is not prescribed by this specification and it is at the discretion of the user to implement it according to their requirements.

This specification does, however, outline a process for achieving such additional validation using standard XML tools in order to,

- Validate that instance data conforms to the advertisement object definition constraints using XML Schema validation
- Report that instance data conforms to the advertisement object definition constraints that cannot be enforced using XML Schema validation

In this 'XML' process, users can validate the structured description instance data against advertisement object definition constraints by using XSLT to transform an XML advertisement object definition instance conformant to the *advertisement object definitions* schema in to a specific XML Schema representation of the advertisement object definition constraints. A second XSLT can be used to transform the structured descriptions instance data from the generic format in to an XML instance of the specific schema generated from the advertisement object definition. Standard XML tools (e.g. XML Schema validation and XSLT) can then be used to validate and report on the instance data.

The conceptual outline of the process used to express advertisement object definition and structured description instance data validation is as follows,

- A rule set for an advertisement object definition is created using the spreadsheet version of the rule set data entry form. The user records the list of properties and modifiers used to define the properties of the advertisement object.
- The rule set is then expressed as an advertisement object definition XML instance that conforms to the *advertisement object definitions* schema.[4]
- The instance ad object description data is exchanged as simple name:value pairs in structured descriptions instance data that conforms to the *structured descriptions* schema. The structured descriptions instance data is embedded in the ad content that it is describing as a plug-in module. This generic name:value format of the structured descriptions metadata provides the common interface for systems.
- On receipt of the ad content, the user can validate the received ad content message using standard XML tools.
- If desired, the user can further validate the structured descriptions instance data by schema validation. For example, using XSLT,

    An XML Schema from the advertisement object definition XML file can be generated using XSLT.

    An XML instance conformant to this generated schema can be populated with data from the structured descriptions instance data using XSLT.

    The structured descriptions instance data can then be validated against this specific XML Schema definition to schema validate the instance ad object description against its defined ad object definition constraints.

    Further validation not possible to enforce using XML Schema could be applied to report on and check if values are present or not using another XSLT. For example, report whether a value that is 'strongly encouraged' to be present is present or not.

Note that the schema language chosen to express a specific schema is not restricted. The user is free to use the schema language (e.g. W3C XML Schema, RELAX NG, DTD) they need in order to achieve their required level of validation.

# 11 Conformance

The conformance requirements for AdsML Structured Descriptions of Advertisement Objects are described throughout this specification document and its' accompanying XML Schema,

- `AdsMLStructuredDescriptions-1.0-Main-AS.xsd`
- `AdsMLStructuredDescriptions-1.0-PublicTypeLibrary-AS.xsd`

The minimal conformance requirement is that instance structured description data is represented using the XML format conformant to the *structured descriptions* schema structure and that it references a rule set represented in the XML format conformant to the *advertisement object definitions* schema structure. The constraints enforced by these schema structures **MUST** be adhered to.

As stated in Section 10 Validation of instance data above, the extent to which a user performs additional validation of structured description instance data against an advertisement object definition beyond the schema constraints imposed by the *structured descriptions* schema structure is at the discretion of the user.

[4] Note: An example conversion process could be a flat XML export from the spreadsheet followed by an XSLT transformation on the flat XML to generate an XML instance conformant to the *advertisement object definitions* schema.

# 12 References

## 12.1 Normative References

- [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)
- [W3C] W3C (World Wide Web Consortium). Ed. Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau. *Extensible Markup Language (XML) 1.0 (Fourth Edition)*. W3C Recommendation, 16 August 2006. (http://www.w3.org/TR/REC-xml)
- [W3C] W3C (World Wide Web Consortium). Ed. Henry S. Thompson, David Beech, Murray Maloney, Noah Mendelsohn. *XML Schema Part 1: Structures Second Edition*. W3C Recommendation, 28 October 2004. (http://www.w3.org/TR/xmlschema-1/)
- [W3C] W3C (World Wide Web Consortium). Ed. Paul V. Biron, Ashok Malhotra. *XML Schema Part 2: Datatypes Second Edition*. W3C Recommendation, 28 October 2004. (http://www.w3.org/TR/xmlschema-2/)
- [W3C] W3C (World Wide Web Consortium). Ed. Tim Bray, Dave Hollander, Andrew Layman. *Namespaces in XML*. W3C Recommendation, 14 January 1999. (http://www.w3.org/TR/REC-xml-names/)
- [W3C] W3C (World Wide Web Consortium). Ed. James Clark, Steve DeRose. *XML Path Language (XPath) Version 1.0*. W3C Recommendation, 16 November 1999. (http://www.w3.org/TR/xpath)
- [W3C] W3C (World Wide Web Consortium). Ed. Jonathan Marsh. *XML Base*. W3C Recommendation, 27 June 2001. (http://www.w3.org/TR/xmlbase/)
- [W3C] W3C (World Wide Web Consortium). *The "xml" Namespace*. W3C namespace for 'xml' prefix, 4 June 2001. (http://www.w3.org/XML/1998/namespace)
- [W3C] W3C (World Wide Web Consortium). Ed. Paul Grosso, Eve Maler, Jonathan Marsh, Norman Walsh. *XPointer Framework*. W3C Recommendation, 25 March 2003. (http://www.w3.org/TR/xptr-framework/)
- [W3C] W3C (World Wide Web Consortium). Ed. Paul Grosso, Eve Maler, Jonathan Marsh, Norman Walsh. *XPointer element() Scheme*. W3C Recommendation, 25 March 2003. (http://www.w3.org/TR/xptr-element/)
- [W3C] W3C (World Wide Web Consortium). Ed. Steven J. DeRose, Ron Daniel Jr., Eve Maler, Jonathan Marsh. *XPointer xmlns() Scheme*. W3C Recommendation, 25 March 2003. (http://www.w3.org/TR/xptr-xmlns/)

## 12.2 Other References (Non-Normative)

- [IETF RFC 1766] H. Alvestrand. *Tags for the Identification of Languages*. Internet Engineering Task Force (IETF), Request for Comments: 1766, March 1995 (http://www.ietf.org/rfc/rfc1766.txt)
- [IETF RFC 2045] N. Freed, N. Borenstein. *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*. Internet Engineering Task Force (IETF), Network Working Group, Request for Comments: 2045, November 1996 (http://www.ietf.org/rfc/rfc2045.txt)
- [IETF RFC 2046] N. Freed, N. Borenstein. *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*. Internet Engineering Task Force (IETF), Network Working Group, Request for Comments: 2046, November 1996 (http://www.ietf.org/rfc/rfc2046.txt)
- [IETF RFC 2646] R. Gellens, Editor. *The Text/Plain Format Parameter*. Internet Engineering Task Force (IETF), Network Working Group, Request for Comments: 2646, August 1999 (http://www.ietf.org/rfc/rfc2646.txt)

- [IETF RFC 2801] D. Burdett. Internet Open Trading Protocol - IOTP Version 1.0. Internet Engineering Task Force (IETF), Network Working Group, Request for Comments: 2801, April 2000 (http://www.ietf.org/rfc/rfc2801.txt)
- [IETF RFC 3023] Eds. M. Murata, S. St.Laurent, D.Kohn. *XML Media Types*. Internet Engineering Task Force (IETF), Network Working Group, Request for Comments: 2396, January 2001. (http://www.ietf.org/rfc/rfc3023.txt)
- [IETF RFC 3986] T. Berners-Lee, R. Fielding, L. Masinter. *Uniform Resource Identifier (URI): Generic Syntax*. Internet Engineering Task Force (IETF), Network Working Group, Request for Comments: 3986, January 2005 http://www.ietf.org/rfc/rfc3986.txt).
- [IETF RFC 4646] A. Phillips, M. Davis. *Tags for Identifying Languages*. Internet Engineering Task Force (IETF), The Internet Society, Request for Comments: 4646, September 2006. (http://www.ietf.org/rfc/rfc4646.txt)
- [IETF RFC 4647] A. Phillips, M. Davis. *Matching of Language Tags*. Internet Engineering Task Force (IETF), The Internet Society, Request for Comments: 4647, September 2006. (http://www.ietf.org/rfc/rfc4647.txt)
- [Ifra] Ifra (INCA-FIEJ Research Association[5]). *IFRA IfraAdConnexion XML Vocabulary Version 1.1, Release 2*. IFRA, 13 May 2003. (http://www.ifra.com/)
- [ISO/IEC 10646] ISO (International Organization for Standardization). *ISO/IEC 10646-1993 (E). Information technology -- Universal Multiple-Octet Coded Character Set (UCS) -- Part 1: Architecture and Basic Multilingual Plane*. International Organization for Standardization, 1993
- [ISO/IEC 10646-2000] ISO (International Organization for Standardization). *ISO/IEC 10646-1:2000. Information technology -- Universal Multiple-Octet Coded Character Set (UCS) -- Part 1: Architecture and Basic Multilingual Plane*. International Organization for Standardization, 2000
- [ISO 639] ISO (International Organization for Standardization). *ISO 639:1988 (E). Code for the representation of names of languages*. International Organization for Standardization, 1988.
- [ISO 3166] ISO (International Organization for Standardization). *ISO 3166-1:1997 (E). Codes for the representation of names of countries and their subdivisions -- Part 1: Country codes*. International Organization for Standardization, 1997.
- [ISO 8601] ISO (International Organization for Standardization). *ISO 8601:2000 Data elements and interchange formats -- Information interchange -- Representation of dates and times (2000-12-21)*. International Organization for Standardization, 2000.
- [NAA] NAA (Newspaper Association of America). Classified Advertising Standards Task Force. *CREST v2.0. NAA Standard for Classified Advertising Data*. (adex version 1.2 document type definition (adex012.dtd)). NAA Classified Advertising Standards Task Force, 24 March 1999. (http://www.naa.org/technology/clsstdtf/)
- [W3C-NOTE-datetime] Misha Wolf, Charles Wicksteed. *Date and Time Formats*. Submitted to W3C 15 September 1997, revised version 27 August 1998. (http://www.w3.org/TR/NOTE-datetime)
- [W3C-NOTE-unicode-xml] Unicode Technical Committee, W3C Internationalization Working Group/Interest Group. *Unicode in XML and other Markup Languages. Unicode Technical Report #20, W3C Note 13 June 2003 and Time Formats*. 13 June 2003. (http://www.w3.org/TR/2003/NOTE-unicode-xml-20030613/)

[5] "INCA" stands for "International Newspaper Colour Association". "FIEJ" stands for "Fédération Internationale des Editeurs de Journaux".

- [W3C Working Group Note] W3C Working Group Note. *Proposal for XML Fragment Identifier Syntax 0.9.* W3C Working Group Note 12 September 2003. (http://www.w3.org/TR/2003/NOTE-xml-fragid-20030912/)

## 13 Appendix A: Acknowledgement for contributions to this document

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

- The AdsML Technical Working Group