Open Geospatial Consortium

Publication Date: 2015-02-24

Approval Date: 2015-02-22

Submission Date: 2014-11-04

External identifier of this OGC® document: http://www.opengis.net/doc/IS/arml/2.0

Normative URL for this OGC® document: http://docs.opengeospatial.org/is/12-132r4/12-132r4.html

Internal reference number of this OGC® document: OGC 12-132r4

Version: 1.0

Category: OGC® Implementation Standard

Editor: Martin Lechner

OGC Augmented Reality Markup Language 2.0 (ARML 2.0)

Encoding Standard

Warning

This document is an OGC Member approved international standard, however, this version is informative. The normative version can be found at: http://docs.opengeospatial.org/is/12-132r4/12-132r4.html. This document is available on a royalty free, non-discriminatory basis. Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document type:	OGC® Publicly Available Standard
Document subtype:	Encoding
Document stage:	Approved for public release
Document language:	English

i. Abstract

This OGC® Standard defines the Augmented Reality Markup Language 2.0 (ARML 2.0). ARML 2.0 allows users to describe virtual objects in an Augmented Reality (AR) scene with their appearances and their anchors (a broader concept of a *location*) related to the real world. Additionally, ARML 2.0 defines ECMAScript bindings to dynamically modify the AR scene based on user behavior and user input.

ii. Keywords

The following are keywords to be used by search engines and document catalogues.

ogcdoc ar augmented reality virtual objects arml virtual reality mixed reality 3d graphics model

iii. Preface

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

iv. Submitting organizations

The following organizations submitted this Document to the Open Geospatial Consortium Inc. as a Request For Adoption:

a) Wikitude GmbH.
b) Georgia Tech
c) SK Telecom
d) Laramie Range Innovations, LLC

v. Submitters

All questions regarding this submission should be directed to the editor or the submitters:

Name	Company
Martin Lechner	Wikitude GmbH.
Blair MacIntyre	Georgia Tech
Sang Chul Kang	SK Telecom
Scott Simmons	Laramie Range Innovations, LLC

Contents

1 Scope

The scope of the ARML 2.0 standard is to provide an interchange format for Augmented Reality (AR) applications to describe an AR scene, with a focus on vision-based AR (as opposed to AR relying on audio etc.). The format describes the virtual objects that are placed into an AR environment, as well as their registration in the real world. ARML 2.0 is specified as an XML grammar. Both the specification as well as the XSD schema are provided.

Additionally, ARML 2.0 provides ECMAScript bindings to allow dynamic modification of the scene, as well as interaction with the user. The ECMAScript bindings, described in JSON, use the same core object models as the XML grammar and include event handling and animations.

The goal of ARML 2.0 is to provide an extensible standard and framework for AR applications to serve the AR use cases currently used or developed. With AR, many different standards and computational areas developed in different working groups come together. ARML 2.0 needs to be flexible enough to tie into other standards without actually having to adopt them, thus creating an AR-specific standard with connecting points to other widely used and AR-relevant standards.

As a requirement, a device running an AR implementation using ARML 2.0 must have a component (screen, see-through display etc.) where the virtual objects are projected on. The device must have sensors such as a camera, GPS, and orientation to analyze the real world - .

Users interact with the virtual scene by moving around in the real world. Based on the movement of the user, the scene on the screen is constantly updated. A user can also interact with the scene by selecting virtual objects, typically by touching them on the screen. However, how a user can select a virtual object is application- and device-specific and out of scope for ARML 2.0.

The plan is to extend ARML in the future to support non-visual virtual objects, such as sound and haptic feedback. The current specification of ARML 2.0, however, focuses on visual objects.

2 Normative References

The following normative documents contain provisions that, through reference in this text, constitute provisions of this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

XML Schema Part 1: Structures Second Edition. W3C Recommendation (28 October 2004)
http://www.w3.org/TR/xmlschema-1/

ECMAScript Language Specification
http://www.ecma-international.org/ecma-262/5.1/ECMA-262.pdf

Web IDL Specification
http://www.w3.org/TR/WebIDL/

OpenGIS® Geography Markup Language (GML) Encoding Standard version 3.2.1 (2007)
http://www.opengeospatial.org/standards/gml

COLLADA Specification
http://www.khronos.org/collada/

XML Path Language (XPath) 2.0
http://www.w3.org/TR/xpath20/

3 Terms and Definitions

Terms and definitions used in this document are reused form the AR Glossary developed by the International AR Standards Community *[AR Glossary]* where applicable. The glossary is a public document (insert URL); the community's chairperson gave specific permission for use.

The following definitions are used within the document:

3.1 (AR) Implementation or **AR Application**: Any service that provides *Augmentations* to an AR-ready device or system.

3.2 Device: the hardware unit the *AR implementation* is running on.

3.3 Augmentation: A relationship between the real world and a *digital asset*. The realization of an *augmentation* is a *composed scene*. An augmentation may be formalized through an authoring and publishing process where the relationship between real and virtual is defined and made discoverable.

3.4 Digital Asset: Data that is used to augment users' perception of reality and encompasses various kinds of digital content such as text, image, 3d models, video, audio and haptic surfaces. A digital asset is part of an *augmentation* and therefore is rendered in a *composed scene*. A digital asset can be scripted with behaviors. These scripts can be integral to the object (for example, a GIF animation) or separate code artifacts (for example, browser markup). A digital asset can have styling applied that changes its default appearance or presentation. **Visual Assets** are *digital assets* that are represented visually. As ARML in its current version focuses on visual representations of augmentations, only Visual Assets are allowed.

3.5 Composed Scene: Produced by a system of sensors, displays and interfaces that creates a perception of reality where *augmentations* are integrated into the real world. A composed scene in an augmented reality system is a manifestation of a real world environment and one or more rendered *digital assets*. It does not necessarily involve 3D objects or even visual rendering. The acquisition of the user (or device)'s current pose is required to align the composed scene to the user's perspective. Examples of composed scenes with visual rendering (AR in camera view) include a smartphone application that presents visualization through the handheld video display, or a webcam-based system where the real object and augmentation are displayed on a PC monitor.

3.6 Camera View or **AR View**: the term used to describe the presentation of information to the user (the *augmentation*) as an overlay on the camera display.

4 Conventions

4.1 Abbreviated terms

AR	Augmented Reality
ARML	Augmented Reality Markup Language
GML	Geography Markup Language
JSON	JavaScript Object Notation
KML	Keyhole Markup Language
OGC	Open Geospatial Consortium
XML	Extensible Markup Language
XSD	W3C XML Schema Definition Language

4.2 Schema language

The XML implementation specified in this Standard is described using the XML Schema language (XSD) [*XML Schema Part 1: Structures*].

4.3 Scripting Components

The Scripting components described are based on the ECMAScript language specification [ECMAScript Language Specification] and are defined using Web IDL [Web IDL Specification].

5 Introduction

Even though AR has been researched for decades, no formal definition of AR exists. Below are two descriptions/definitions of AR:

[Wikipedia AR Definition (insert date pulled)]: Augmented reality (AR) is a live, direct or indirect, view of a physical, real-world environment whose elements are *augmented* by computer-generated sensory input such as sound, video, graphics or GPS data. As a result, the technology functions by enhancing one's current perception of reality. AR is about augmenting the real world environment with virtual information by improving people's senses and skills. AR mixes virtual characters with the actual world.

[Ronald Azuma AR Definition]: Augmented Reality is a system that has the following three characteristics:

- Combines real and virtual
- Interactive in real time
- Registered in 3-D

5.1 History of ARML - ARML 1.0

ARML 2.0's predecessor, ARML 1.0 [ARML 1.0 Specification], was developed in 2009 as a proprietary interchange format for the Wikitude World Browser. ARML 2.0 does not extend ARML 1.0. Instead ARML 2.0 is a complete redesign of the format. ARML 1.0 documents are not expected to work with implementations based on ARML 2.0. ARML without a version number implicitly stands for ARML 2.0 in this document.

ARML 1.0 is a descriptive, XML based data format, specifically targeted for mobile AR applications. ARML focuses on mapping geo-referenced Points of Interest (POIs) and their metadata, as well as mapping data for the POI content providers publishing the POIs to the AR application. The creators of the Wikitude World Browser defined ARML 1.0 in late 2009, to enable developers to create content for AR Browsers. ARML 1.0 combines concepts and functionality typically shared by AR Browser, reuses concepts defined in OGC's KML standard and is already used by hundreds of AR content developers around the world.

ARML 1.0 is fairly restrictive and focuses on functionality Wikitude required back in 2009. Thus, ARML 2.0, while still using ideas coming from ARML 1.0, is targeted to be a complete redesign of the 1.0 format, taking the evolution of the AR industry, as well as other concepts and ideas into account.

6 ARML 2.0 - Object Model (normative)

Requirements Class	
http://www.opengis.net/spec/arml/2.0/req/core	
Target Type	Software Implementation

6.1 General Concepts

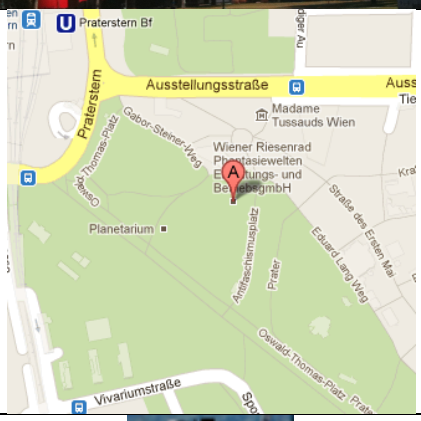
6.1.1 Features, Anchors and VisualAssets

In ARML 2.0, a Feature represents a real world object that should be augmented. Using the Ferris Wheel below as an example, the Feature to augment is the Ferris Wheel itself. Technically speaking, a Feature consists of some metadata on the real world object, as well as one or more Augmentations that describe where a Feature is located in the composed scene. In ARML 2.0 terms, an Augmentation is called an Anchor.

Anchors define the link between the digital and the physical world (a broader concept of a *location*). An Anchor describes where a particular Feature is located in the real world. An Anchor can be either a spatial location that is tracked using location and motion sensors on the device, or a visual pattern (such as markers, QR codes or any sort of reference image) that can be detected and tracked in the camera stream using computer vision technology. In the Ferris Wheel example, the Anchor is the geospatial location of the Ferris Wheel in Vienna.

Finally, VisualAssets describe how a particular Anchor should be represented in the Composed Scene. VisualAssets can either be 2-dimensional (such as text or images) or 3-dimensional. The icon and the text in the example below represent VisualAssets that are attached to the Anchor of the Ferris Wheel, causing the Ferris Wheel to be augmented with the Visual Assets as soon as the Anchor is visible in the scene.

Examples:

Geospatial AR		
Feature		The physical object: The Riesenrad (Ferris Wheel) in Vienna, including Metadata
Anchor		Its location: 48.216581,16.395847
VisualAsset		The digital object that is used to represent the Feature in the scene.
Result		As soon as the location of the Ferris Wheel is detected to be in the field of vision (typically using GPS, motion sensors, magnetometers etc.), the VisualAsset is projected onto the corresponding position on the screen.

Computer Vision-based AR		
Feature	The security features of a 10-dollar-note	
Anchor		A US 10 Dollar-note (along with the location of the security features on the note).
VisualAsset		Some buttons that can be pressed to get more information on a particular security feature
Result		As soon as the 10 Dollar note is detected in the scene, the VisualAssets are projected onto the note in the correct positions.

6.1.2 Declarative and Scripting Specification

ARML 2.0 comes with a declarative specification, serialized in XML, describing the objects in the AR scene (section 7), as well as a scripting specification allowing dynamically modifying the scene and reacting on user-triggered events (section 8). The scripting specification uses ECMAScript for the scripting parts and a JSON serialization of the objects for accessing the objects' properties.

The scripting specification declares hooks to the descriptive spec, so both specs, while existing separately from another, work together for a dynamic experience. An implementation can chose to only support the declarative spec (for instance in case scripting parts cannot be implemented on the platform the implementation is running on).

The scripting specification contains sections, which are intended for advanced users only. These sections are clearly marked as *Advanced ARML* in the title and are intended for those already familiar with the basic concepts of ARML.

6.1.3 Document Structure

An ARML document is grouped into three parts: The declarative part (AR Elements), the styling part and the scripting part.

- The *ARElements* element contains a list of *ARElement* objects, as specified in the ARML specification below.
- The optional *style* element contains styles (typically CSS) used for styling the virtual objects in the scene.
- The optional *script* part contains scripting code (typically ECMAScript or JavaScript).

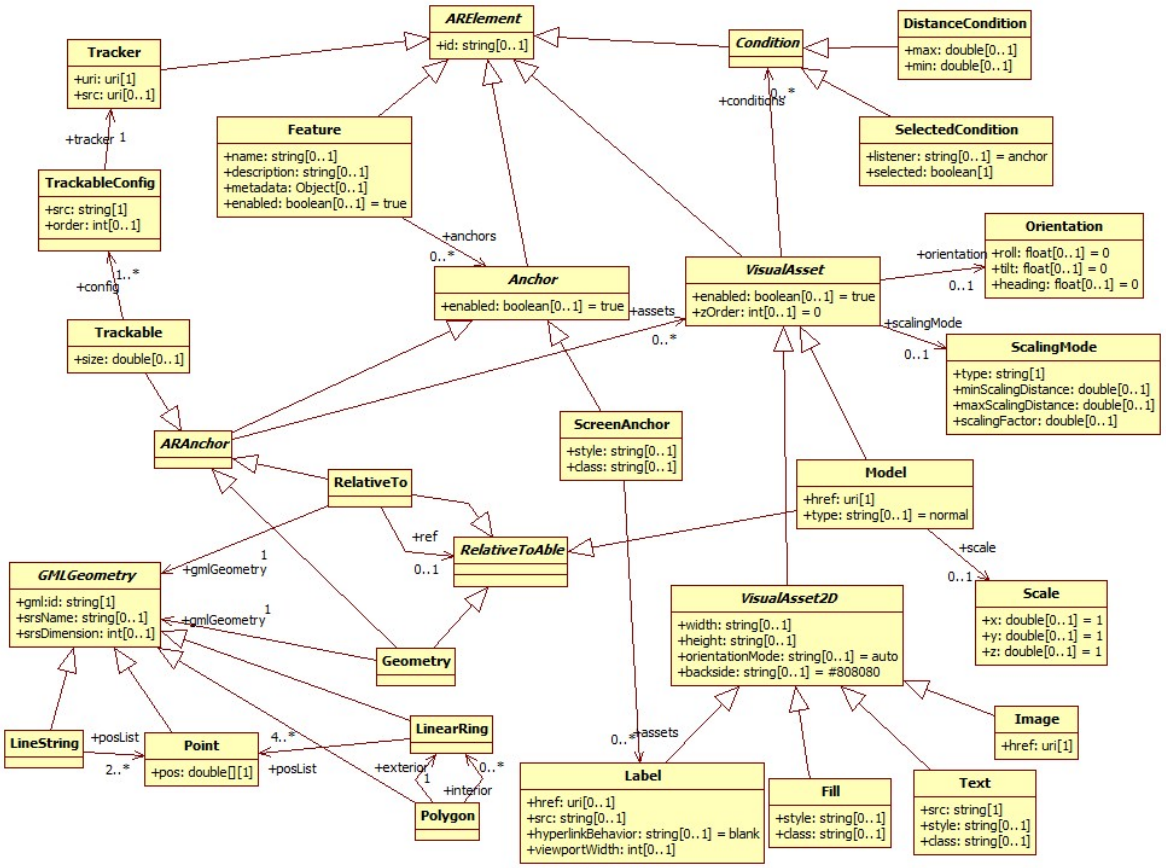
The following section will describe the ARElements section.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/parse_encoding
An implementation shall be able to parse valid ARML 2.0 XML encodings, as defined in section 7.

6.1.4 ARML 2.0 Object Model Diagram

ARML 2.0 is built on top of a generic object model to allow future serializations in different languages, as well as good extensibility for future needs.

The diagram below shows the generic object model in ARML 2.0.



6.1.5 Units

Units in ARML are given in meters. Whenever any virtual object in ARML has a size of x meters, the size of this object on the screen is equal to a real world object of the same size and the same distance in the camera view.

Remark: The actual size on the screen is dependent on certain camera parameters on the device.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/units
All units are specified in meters. The specified size of a virtual object corresponds to the size of a real world object of the same size at the same distance.

6.2 interface ARElement

Most classes specified in ARML 2.0 are derived from *ARElement*. An *ARElement* has an optional *id* property, which uniquely identifies the object. The *user* id is pre-assigned by the system and must not be used in the encoding. If *user* is used, the attribute must be ignored.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/ARElement/id_user
In case an ARElement's id property is set to *user*, the property shall be ignored.

Properties:

Name	Description	Type	Multiplicity
id	The unique ID of the ARElement	string	0 or 1

id

The unique ID of the *ARElement* which makes it uniquely accessible and referenceable.

6.3 class Feature

Inherits From ARElement.

A *Feature* is an abstraction of a real world phenomenon *[GML Specification]*. In ARML, a Feature has one or more *Anchors*, which describe how the *Feature* is registered in the real world. Each of these *Anchors* has one or more *VisualAssets* attached to it, which visually represent the Feature in the composed scene.

Example:

The physical object: The Riesenrad (Ferris Wheel) in Vienna, including Metadata

Properties:

Name	Description	Type	Multiplicity
name	The name of the Feature	string	0 or 1
description	A description of the Feature	string	0 or 1
enabled	A boolean flag controlling the state of the Feature	boolean	0 or 1

Name	Description	Type	Multiplicity
metadata	Arbitrary metadata	Any XML	0or 1
anchors	A list of anchors the Feature is referenced with	Anchor[]	0 or 1

name
The optional name of the Feature. Can be reused in Label and Text VisualAssets by using $[name] in the Label or Text. Additionally, the name of the Feature is used as a Text-VisualAsset when an Anchor of the Feature has no VisualAsset attached to it. The property can be omitted.

description
The optional description of the Feature. Can be reused in Label and Text VisualAssets by using $[description] in the Label or Text.

metadata
Allows the storage of arbitrary metadata for the Feature. Any XML content can be used. The content may or may not conform to a custom scheme.

enabled
Setting the boolean flag to true (enabled) means that VisualAssets attached to the Anchors of the Feature are part of the composed scene, setting it to false (disabled) causes all Assets attached to the Feature to be ignored for the composed scene (i.e. they are never visible in the AR View). Defaults to true if not given.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/Feature/enabled
Setting a Feature's enabled property to false causes any Anchors and VisualAssets attached to the Feature to be ignored in the scene (i.e. they will not be visible at any time).

anchors
contains a list of Anchors describing the Anchors of the Feature in the real world.

6.4 interface Anchor

Inherits From ARElement.

An *Anchor* describes the registration (location) of a *Feature* in the real world or on the screen. Two different types of Anchors are used in ARML:

- *ARAnchor* describes the location of a Feature in the real world. This Anchor is used for virtual objects that are registered in the real world and move around on the screen as the user moves around.
- *ScreenAnchor* describes a fixed location of a Feature on the screen. This Anchor is used for objects that have a fixed location on the screen (similar to HTML components inside a HTML page). The objects associated with a ScreenAnchor will not move when the user is moving

around, but remains static on the screen. Typical use cases are game HUDs or static informational displays on certain Features.

Properties:

Name	Description	Type	Multiplicity
enabled	The state of the anchor	boolean	0 or 1

enabled
Setting the boolean flag to true (enabled) means that VisualAssets attached to the Anchor are part of the composed scene (if the Feature the Anchor is attached to is also enabled), setting it to false (disabled) causes all VisualAssets attached to the Anchor to be ignored in the composed scene (i.e. they are never visible in the AR View). Defaults to true if not given.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/Anchor/enabled
Setting an Anchor's enabled property to false causes any VisualAssets attached to the Anchor to be ignored in the scene (i.e. they will not be visible at any time).

Remark: Anchors are typically used within Features; however, an Anchor can also exist outside a Feature. Regardless if it is located within a Feature or was defined separately (immediately within the *ARElements* section), it is part of the composed scene.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/Anchor/anchor_without_feature
An Anchor is part of the composed scene even when it is not attached to a Feature, i.e. it is an immediate child of the ARElements tag.

6.4.1 interface ARAnchor

Inherits From Anchor.

An ARAnchor describes the registration (location) of a Feature in the real world. An ARAnchor might be declared using spatial coordinates, i.e. a *location* in a (geo-) spatial sense, or an image or marker that is recognized in the live camera video stream and even a sound that is recognized over the microphone.

ARAnchor is an abstract class, which must not be instantiated directly. We define the following concrete types of ARAnchors in ARML:

Geometry		Either a Point, LineString or Polygon, described with spatial coordinate tuples

Trackable		A visual pattern that is detected in the camera stream
RelativeTo		An Anchor relative to other objects (e.g. another ARAnchor); useful to create large scenes relative to another Trackable

Properties:

Name	Description	Type	Multiplicity
assets	The assets representing the anchor in the live scene	Asset[]	1

assets
A list of VisualAssets attached to the ARAnchor. These VisualAssets will represent the ARAnchor.

If no VisualAsset is supplied, a *Text* VisualAsset, with its text set to the *name* of Feature the ARAnchor is attached to, is used as the default VisualAsset. In case even the *name* property is omitted for the Feature, no VisualAsset is attached as default.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/ARAnchor/no_visual_asset
In case no valid VisualAsset is attached to an ARAnchor in the encoding, the behavior depends on whether the ARAnchor is attached to a Feature with its name property set. In case it is set, an auto-generated VisualAsset of type Text is added automatically. The text of the Text is set to the value of the name property of the Feature. In case no such name is set, no VisualAsset is attached.

6.4.1.1 Local Coordinate System and Dimensions

Any ARAnchor type, with LineStrings being the only exception, specifies its own local coordinate system. This allows VisualAssets to be placed on top of any ARAnchor (see section 6.5.2 for details), and RelativeTo Anchors created relative to an underlying ARAnchor. For each ARAnchor type, it is explicitly stated how the coordinate system is defined for this particular type of ARAnchor. Additionally, each ARAnchor has a dimension associated with it. As VisualAssets take on different dimensions (a Text is 2D, while a 3D model is 3D), it is important to define the dimension of an ARAnchor as well, to allow a high level definition of how an n-dimensional Visual Asset will be rendered on top of an m-dimensional ARAnchor, without having to specifically consider each ARAnchor and VisualAsset combination.

Wherever a concrete ARAnchor is defined, the dimension and coordinate system is defined as well, except for ARAnchors with a dimension of 1 (Lines). Due to their nature, these ARAnchors do not define a local coordinate system.

6.4.1.2 class Geometry

Inherits from ARAnchor.

A Geometry Anchor is used when a Feature is registered in the real world using spatial coordinates (such as geospatial locations). The Geometry Anchor serves as a wrapper for GMLGeometries, which essentially describe the spatial location of the Feature. A Geometry Anchor contains all properties inherited from ARAnchor, as well as an additional element, which describes the wrapped GMLGeometry and the spatial coordinates.

The following GMLGeometries are allowed in ARML 2.0 and are described below:

- *gml:Point* (a single position)
- *gml:LineString* (a list of positions, connected to form a line)
- *gml:Polygon* (a list of positions, connected to form a planar area)

Remark: Geometry anchors can only be considered if an implementation is capable of detecting the user's current position and is thus capable of calculating spatial relationships between the user and the Geometry anchors.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/Geometry/no_position
In case the implementation is not able to detect the current position of the user in the coordinate reference system used in the encoding, any Geometry anchor shall be ignored.

6.4.1.2.1 interface GMLGeometries

The GMLGeometries Point, LineString and Polygon are reused from the GML specification *[GML Specification]*.

Complying with the GML specification, each GMLGeometry must have an id property.

The default coordinate reference system (CRS) for Geometries is WGS84 (EPSG code 4326; "longitude latitude"; decimal numbers; no altitude). Alternative CRSes can be specified using *srsName*, either by supplying the EPSG code *[EPSG Codes]*, or by pointing to an OGC WKT CRS definition. Implementations are required to at least support WGS84. If a certain CRS used in an encoding is unknown to an implementation, the entire Geometry Anchor must be ignored.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/GMLGeometries/crs
Any implementation shall support at least the WGS84 CRS. Any Geometry Anchor using a GMLGeometry with an alternative CRS, which the implementation cannot support, shall be ignored by the implementation.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/GMLGeometries/default_crs
In case no *srsName* attribute is specified, the CRS for the GMLGeometry defaults to WGS84 (format: longitude latitude).

If altitude values are provided, the CRSes dimension must be set to 3 (see *srsDimension*), and values must be provided in "longitude latitude altitude" format (altitude in meters). If no altitude is supplied, the altitude of every position will be set to the user's current altitude.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/GMLGeometries/no_altitude
If no altitude information is supplied for a Particular GMLGeometry, the altitude value is set to the same elevation of the user's current altitude.

GML Geometries also allow other attributes, such as axis labels etc., which are not relevant in the context of ARML and can thus be safely omitted.

Properties:

Name	Description	Type	Multiplicity
gml:id	A unique ID for the geometry	string	1
gml:srsName	The link to a well-known CRS or an EPSG code	string	0 or 1
gml:srsDimension	The dimension of the CRS specified	Positive Integer	0 or 1

gml:id
A unique ID, required as per the GML specification.

srsName
optionally specifies either a link to an OGC WKT CRS, or an EPSG code. If *srsName* is omitted, WGS84 is implicitly assumed to be the default CRS.

srsDimension
The optional attribute srsDimension specifies the number of coordinate values in a position (i.e. the dimension of the underlying CRS). srsDimension should be used when srsName is specified, and must be a positive integer. If both srsName and srsDimension are not given, srsDimension defaults to 2.

6.4.1.2.2 class Point

Derived from [GML Specification].

A Point specifies a position in the referenced coordinate reference system by a single coordinate tuple.

Properties:

Name	Description	Type	Multiplicity
pos	The list of doubles, specifying the position of the Point	List of double values	1

pos
Specifies the coordinate vector describing the position of the Point, in a blank-separated list.

Remark: GML allows the specification of a custom *srsName* and *srsDimension* also on the *pos*-level, but states that it is unlikely that this will be used in a useful way. The same applies for ARML 2.0.

6.4.1.2.3 class LineString

Derived from [GML Specification].

A LineString is defined by two or more coordinate tuples, with linear interpolation between them. The number of direct positions in the list shall be at least two. The segments created by interpolation between the coordinate tuples are called *LineString segments*.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/GMLGeometries/LineString/definition
A LineString that does not consist of at least two coordinate tuples shall be ignored.

Properties:

Name	Description	Type	Multiplicity
posList	The list of doubles, specifying the vector of positions of the LineString	list of double values	1
pointProperty	A List of gml:Point elements that make up the LineString	List of gml:Point element	2 .. *

posList
Specifies the list coordinate vectors describing the vertices of the LineString, in a blank-separated list.

Remark: GML allows the specification of a custom *srsName* and *srsDimension* also on the *posList*-level, but states that it is unlikely that this will be used in a useful way. The same applies for ARML 2.0.

pointProperty
Specifies the list of Points describing the vertices of the LineString. Must appear at least twice.

6.4.1.2.4 class Polygon

Derived from [GML Specification].

A Polygon is a planar object defined by an outer boundary and 0 or more inner boundaries. The boundaries are specified using the *exterior* and *interior* elements. The boundaries, in turn, are defined by LinearRings.

A LinearRing is a closed LineString (with at least 4 coordinates) that should not cross itself. It is defined in the exact same way as a LineString, except the element tag is called *LinearRing*. Simplified, a LinearRing is a LineString where the last position equals the first position.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/GMLGeometries/LinearRing/definition
A LinearRing that does not consist of at least 4 coordinate tuples, or its starting point is not equal to its endpoint, shall be ignored.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/GMLGeometries/Polygon/definition
A Polygon shall consist of one and only one exterior LinearRing, and zero or more interior LinearRings. In case the exterior LinearRing is invalid (see http://www.opengis.net/spec/arml/2.0/req/core/GMLGeometries/LinearRing/definition), the entire Polygon shall be ignored. In case one of the interior LinearRings is invalid, only the particular LinearRing is ignored.

Properties:

Name	Description	Type	Multiplicity
exterior	A LinearRing forming the outer boundary of the Polygon	LinearRing	1
interior	A LinearRing forming a hole in the interior of the Polygon	LinearRing	0 .. *

exterior
A LinearRing forming the outer boundary of the Polygon

interior
A LinearRing forming a hole in the Polygon

6.4.1.2.5 Advanced ARML: Coordinate Reference System and Dimensions

Dimensions:

The dimensions of Geometries are defined as specified in GML (Point: 0, LineString : 1, Polygon: 2). The coordinate systems defined below are all of Cartesian type (i.e. orthogonal axes).

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/GMLGeometries/local_cs/cs_type
The local coordinate system of any GMLGeometry is of Cartesian type.

Local Coordinate Systems:

Point
The ground plane is defined by the projected earth's surface at the specified Point. In case the Point is used relative to a Trackable, the ground plane is formed by the Trackable's surface. The x and y axes run within the ground plane.
Origin: The point itself
x-axis: pointing east (or right, parallel to the Trackable's lower and upper edges, when used relative to a Trackable, see RelativeTo Anchor for details)
y-axis: pointing north (or towards the top edge, running parallel to the left and right edges of the Trackable when used relative to a Trackable)
z-axis: pointing up, perpendicular to earth's (or Trackable's) surface
Unit: Meters

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/GMLGeometries/local_cs/cs_type/Point
A Point's local coordinate reference system shall be defined as stated in paragraph 6.4.1.2.5, section Point.

LineString
Due to their nature, LineStrings do not define their own local coordinate system. Refer to section 6.5.2 for details how to map VisualAssets onto LineStrings. Consequently, LineStrings cannot be used as an originating Anchor for RelativeTo-Geometries (see section 6.4.1.2.5).

Polygon
A Polygon's local coordinate system is derived from the (uniquely defined) bounding rectangle (the smallest rectangle fully enclosing the Polygon) having two of the four edges parallel to the earth's surface (or Trackable's surface when used relative to a Trackable, see RelativeTo Anchor for details).

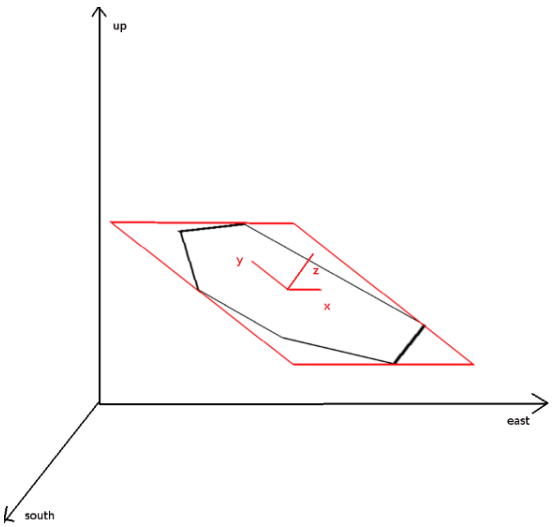
To calculate the BoundingRectangle, take the lowest and highest point (in relation to the altitude) of the Polygon and draw the two lines through these points in the polygon's plane, parallel to the earth's surface. Now, take the easternmost and westernmost point and draw the two lines through these points in the polygon's plane, perpendicular to the earth's surface. The resulting rectangle is the bounding rectangle of the Polygon.
If the Polygon is used relative to a Trackable, take the topmost, bottommost, rightmost and leftmost point relative to the Trackable, as well as the Trackable's surface for the ground plane instead.

This ensures that the bounding rectangle is aligned with the (earth's or Trackable's) surface. The bounding rectangle forms the ground plane of the coordinate system, x and y axis run within the ground plane.
Origin: The point marking the center of the bounding rectangle
x-axis running parallel to the edges of the bounding rectangle which run parallel to the surface. When the origin of the coordinate system is viewed from the center of the lower edge (the one edge parallel to the surface which is closer to the earth's or Trackable's surface) of the bounding rectangle, the x-axis points right
y-axis running perpendicular to x- and z-axis, creating a left-handed coordinate system

z-axis is equal to the Polygon's normal vector
Unit: Meters

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/GMLGeometries/local_cs/cs_type/Polygon
A Polygon's local coordinate reference system shall be defined as stated in paragraph 6.4.1.2.5, section Polygon.

Illustration:



Special case:
In case the Polygon is placed parallel to the earth's (or Trackable's) surface (that means altitude is equal for each vertex), the bounding rectangle cannot be determined in the above definition. In this case, the bounding rectangle's edges are aligned with the vectors pointing north/south and east/west from the first vertex of the Polygon (or up/down and left/right when used relative to a Trackable), and the southern/down edge form the lower edge of the Bounding Rectangle (which is used to determine the x axis).

6.4.1.3 Trackable and Tracker

Trackables are a more general concept of a *location* of a Feature in the real world. Instead of specifying an exact, well known set of coordinates somewhere within a well-known coordinate reference system by using the geometry types specified in the previous section, a Trackable describes something that is tracked in the real world (typically by a camera) and serves as the Anchor of a Feature. As an example, a Trackable could be a 2D image, QR code or 3D model; however, Trackables are not restricted to visual objects. An application could also track Sounds coming in from the microphone. As Trackables are mostly visual in AR implementations, we will put a focus on those.

Two classes are required to specify a Trackable:

- *Trackable*: The Trackable describes the trigger (in whatever form) that should be tracked in the scene. A Trackable might be an artificial game marker, the reference image or reference 3D model, the description of a face, the referenced song etc.
- *Tracker:* A Trackable is always linked to one or more specific Trackers, which references the framework(s) that needs to be used to track the referenced Trackable. For instance, if the

Trackable is a generic image, the Tracker needs to reference a generic image tracking capability the implementation needs to be bundled with. If the implementation uses face tracking and the Trackable describes a specific face, the Tracker needs to reference an underlying face tracking functionality, which is exposed by the implementation.

6.4.1.3.1 class Tracker

Inherits From ARElement.

The Tracker describes the tracking framework to be used to track the Trackables associated with this Tracker.

A Tracker is uniquely and globally identified by a URI. It is not required that any meaningful content is accessible via the URI, however, a developer of a Tracker is encouraged to expose some descriptions about the Tracker when the URI is called from a standard web browser. A definition of the exposed content is beyond the scope of ARML 2.0.

Properties:

Name	Description	Type	Multiplicity
uri	The URI identifying the Tracker	string	1
src	The container the Tracker is operating in	string	0 or 1

uri

To reference the framework used to track the associated Trackables, a Tracker specifies a uri property that uniquely identifies the underlying tracking software. The URI might be registered in a Tracker dictionary that assigns a unique URI to any publicly used Tracker, so AR implementations using the standard can use this as a reference to what tracking framework should be used. The URI might also point to a custom tracker implementation that is used just within the specific implementation. If the URI cannot be resolved to any of the Trackers available on the implementation, the Tracker cannot be used and must be gracefully ignored along with any associated Trackables.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/Trackable_And_Tracker/unknown_tracker
In case an encoding uses a Tracker unknown to the implementation, the implementation shall ignore the Tracker and all its associated Trackables.

src

Optionally specifies a URI which references the container the Tracker is operating in, and the associated Trackables can be found in. This mechanism allows a two-level location of the actual Trackable in case it is contained within a container. *src* must be set if the Trackable is not directly accessible via some sort of URI or any other identifier, but is located in any sort of container, such as a zip file or a proprietary binary container containing all targets.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/Trackable_And_Tracker/contained_trackable
In case the Trackables of a particular Tracker are located in any sort of container (zip file etc.), the *src* property of the Tracker shall point to the container the Trackables are stored in.

The following generic tracker URI is defined for every implementation:

- http://www.opengis.net/arml /tracker/genericImageTracker hosting a tracker, which takes jpeg, png or gif images as image targets. The Trackables can be zipped; the src property must then point to the zip file containing the Trackables.

6.4.1.3.2 class Trackable

Inherits From ARAnchor.

A Trackable represents the object that will be tracked. It provides the actual Anchor of the Feature in the real world.

Conceptually, a Trackable consists of a digital file that describes the Trackable (marker, image etc.), and a Tracker that is used to track the particular Trackable. This linkage between a digital file and the Tracker is defined in a TrackableConfig. Typically, only one TrackableConfig will be supplied per Trackable (one Trackable is attached to a particular Tracker which can read the digital file provided and track it in the camera), however, if a Trackable can be tracked in multiple ways with multiple Trackers (typically requiring a specific digital file to be provided per Tracker), multiple TrackableConfigs can be supplied.

Properties:

Name	Description	Type	Multiplicity
config	Linking the Trackable with the Tracker	TrackableConfig	1 .. *
size	The real world size of the Trackable, in meters	double	0 or 1

config

The config provides the mapping between the Tracker and the Trackable. Each Trackable must have at least one config, but might have more in case the Trackable can be tracked using different Trackers. See TrackableConfig below for details.

size

The size property allows to specify the size of the real world object that is tracked with the Trackable. If the Trackable is any sort of 2-dimensional object (such as images, face descriptions etc.), the size specifies the width of the Trackable in meters. For example, if a billboard advertisement sized 5 by 10 meters in the real world should be tracked, the image representing the Trackable should be in the same aspect ratio as the real object (1:2), and the size property needs to be set to 5. If the Trackable is a 3-dimensional object, the size property specifies the meters representing one unit in the 3D mesh. For example, if the model is using meters as the unit, set size to 1, if it is using feet, set it to 0.3048.

Certain Trackables might already contain information on the actual size of the Trackable within the referenced file. Examples include 3D models in COLLADA file format *[COLLADA Specification]*. In this case, the size property of the Trackable can be omitted. However, the usage of the *size* element is encouraged even in these cases. The size property overrules any size-properties implicitly set in the file format. A Trackable without any defined size (either in the file or with the *size* property) by the implementation must be ignored.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/Trackable_And_Tracker/trackable_2D_size
In case the Trackable is any sort of 2D object, the size in the encoding specifies the width of the tracked object in meters.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/Trackable_And_Tracker/trackable_3D_size
In case the Trackable is any sort of 3D object, the size in the encoding specifies the meters representing one unit in the 3D mesh.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/Trackable_And_Tracker/trackable_size_preset
In case the Trackable's binary representation includes size information and *size* is set in the encoding, the setting in the encoding takes precedence.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/Trackable_And_Tracker/trackable_missing_size
In case the *size* property is not set in the encoding, and no size information is available in the binary representation of the Trackable, the entire Trackable shall be ignored.

TrackableConfig

Name	Description	Type	Multiplicity
tracker	The URI of the Tracker that is used to track the Trackable	string	1
src	The identification of the Trackable	string	1
order	An order of the TrackableConfigs	int	0 .. 1

tracker
The tracker property holds the URI to the referenced Tracker the Trackable will be tracked with (format: #id).

src
The src property references the digital file that contains the description of the Trackable (a marker, an image etc.). Depending on the src property of the Tracker, the src property of the Trackable must be of different formats:

- If *src* of the referenced Tracker is not set, *src* of the Trackable must contain a URI pointing to the Trackable.
- If *src* of the referenced Tracker is set (e.g. pointing to a zip file), *src* of the Trackable must be set to a String that uniquely identifies the Trackable for the given Tracker (e.g. the path to the Trackable in a zip file, or any unique ID in another container)

order
An optional attribute that can be used to set a rank for the config in case multiple configs are available for a particular Trackable. The one with the lowest number is checked first, and only if the referenced Tracker is not available on the implementation, the next configs are considered. If two or more configs have the same order set, it is up to the implementation to decide on an order. If the attribute is not set, it defaults to the maximum integer on the platform, causing these configs to be considered last.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/Trackable_And_Tracker/config_order_max
In case the order property is not set for a particular TrackableConfig, the value is implicitly set to the maximum integer value available for the implementation.

6.4.1.3.3 Advanced ARML: Coordinate Reference System and Dimension

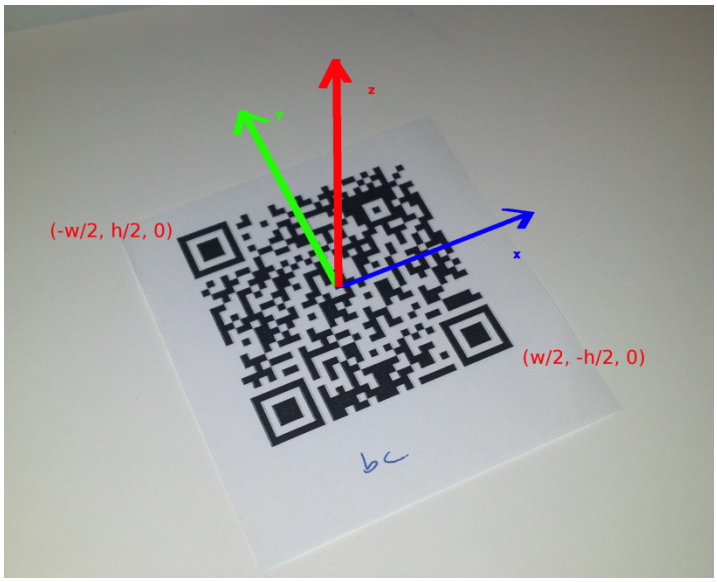
Dimensions:

The *center* (see Local Coordinate Systems below for details) of the Trackable will be tracked, resulting in a 0-dimensional ARAnchor (similar to a Geometry ARAnchor of type *Point*). Other areas of the Trackable (such as Outline etc.) can be tracked using RelativeTo locations, see *RelativeTo* section for details.

Local Coordinate Systems:

2D Trackables (QR Codes, Markers, Images etc.):

origin: the intersection of the diagonals of the bounding rectangle of the marker (for rectangular markers, this is the natural "center" of the image).
x-axis: pointing right with respect to the Trackable, running parallel to the top and bottom edge of the marker
y-axis: pointing up, parallel to the left and right edge of the marker
z-axis: perpendicular to x and z axis (i.e. the plane the Trackable is forming), pointing upwards (out of the marker)
Unit: Meters



w := width of Trackable
h := height of Trackable (calculated based on aspect ratio)

3D Trackables (tracked 3D models):

origin: the origin of the model.
x, y and z axis are reused from the model
Unit: As specified in the size property of the model (or any implicit size detected in the model file itself)

Other Trackables:

Trackables that do not fall into or cannot be mapped onto one of the above categories must specify their local coordinate system on their own.

6.4.1.4 Advanced ARML: class RelativeTo

Inherits From ARAnchor.

RelativeTo Anchors are defined relative to another ARAnchor (except LineStrings), to the user or relative to a Model. RelativeTo allows ARAnchors to be defined relative to other objects, regardless of where they are actually located. A Trackable, for example, defaults to a 0-dimensional ARAnchor. RelativeTo can be used to track the outline or any specific area in the Trackable without having to specify the Trackable again. The area can be specified using the local coordinate system of a Trackable.

RelativeTo is specified using GMLGeometryElements. The coordinate system is calculated according to the rules set forth in Local Coordinate Systems of GMLGeometryElements, based on the underlying ARAnchor or Model (in which case the model's x/z plane serves as the surface plane for coordinate system calculations).

While it is technically possible to define RelativeTo anchors relative to another RelativeTo anchor, usage of this construct is discouraged due to complex local coordinate system handling. It is advised to always base a RelativeTo-Anchor directly on a non-RelativeTo ARAnchor, a Model or the user.

Properties:

Name	Description	Type	Multiplicity
ref	The ARAnchor or Model the RelativeTo Anchor is referencing	string	1
GMLGeometry	The geometry describing the RelativeTo ARAnchor	GMLGeometryElement	1

ref
Specifies the URI to the object the Anchor is referencing. Either another ARAnchor (except LineStrings) or Model, or #user is allowed as reference. If an ARAnchor is specified as *ref*, the ARAnchor's local coordinate system is used to calculate the relative location (based on the GMLGeometryElement of the RelativeTo Anchor). If a Model is used, the engineering coordinate system of the Model is used as coordinate system for the calculation of the relative location.
If #user is provided as reference, the current location of the user is considered a Point-Anchor (with its local coordinate system set accordingly).

GMLGeometry
The GMLGeometry describes the location relative to the object specified in *ref*. Thus, the resulting RelativeTo-Anchor can either be a gml:Point, gml:LineString or gml:Polygon, and the coordinates are given with respect to the underlying coordinate system of the ARAnchor or the coordinate system of the Model.
srsName and *srsDimension* for the GMLGeometryElement are ignored, *srsDimension* is implicitly set to 3. The local coordinate system of the underlying ARAnchor or Model will be used.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/RelativeTo/GMLGeometry_properties
The GMLGeometry properties *srsName* and *srsDimension* are ignored when used within RelativeTo anchors. *srsDimension* is implicitly set to 3.

6.4.2 class ScreenAnchor

Inherits From Anchor.

A *ScreenAnchor* describes a fixed location on the screen which can be used to draw HTML components on the screen which are not registered in the real world and will not move on the screen as the user moves through the environment. A ScreenAnchor describes a rectangular area on the screen, aligned with the edges of the screen.

Properties:

Name	Description	Type	Multiplicity
style	inline styling for the element	String	0 or 1
class	References a CSS class	String	0 or 1
assets	The Labels representing the anchor in the live scene	Label[]	1

style and *class*
see CSS styling for details

CSS Styles are used to position the ScreenAnchor on the screen, similar to absolute positioning of an iframe in a HTML page. The following CSS properties are available for ScreenAnchor:

- *top* specifies how far the top edge of the ScreenAnchor is offset below the top edge of the screen
- *bottom* specifies how far the bottom edge of the ScreenAnchor is offset above the bottom edge of the screen
- *left* specifies how far the left edge of the ScreenAnchor is offset to the right of the left edge of the screen
- *right* specifies how far the right edge of the ScreenAnchor is offset to the left of the right edge of the screen
- *width* specifies the width of the ScreenAnchor
- *height* specifies the height of the ScreenAnchor

top, bottom, left, right, width and height can either be non-negative integer values (representing pixels on the screen) or percentage values (top, bottom and height in percentage of screen height, left, right and width in percentage of screen width). Only one value of top and bottom should be set.

In case of conflicting top/bottom/height values, top takes precedence over height, which takes precedence over bottom. In case of conflicting left/right/width values, left takes precedence over width, which takes precedence over right.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/ScreenAnchor/property_conflicts
In case of conflicting top/bottom/height values, top takes precedence over height, which takes precedence over bottom. In case of conflicting left/right/width values, left takes precedence over width, which takes precedence over right.

If neither top, nor bottom is given, the ScreenAnchor will be placed as if top would be set to 0. If neither left, nor right is given, the ScreenAnchor will be placed as if left would be set to 0. width and height default to 100% if not given.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/ScreenAnchor/missing_properties
If neither top, nor bottom is given, top is set to 0. If neither left, nor right is given, left is set to 0; width and height default to 100% if not given.

It is advised that out of top/bottom/height and left/right/width respectively, 2 out of the 3 values are always specified.

assets
A list of Labels attached to the ScreenAnchor which will be projected on the screen, see *Anchor.assets* for details.

When Labels are attached to a ScreenAnchor, the following properties of the Label will be ignored:

- width and height
- Orientation
- orientationMode
- ScalingMode
- any DistanceConditions

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/ScreenAnchor/ignored_properties
An implementation shall ignore the following properties for any Asset attached to a screen anchor: - width and height - Orientation - orientationMode - ScalingMode - any DistanceConditions

Additionally, the distance from the user to any ScreenAnchor is always 0, causing Labels attached to ScreenAnchors to occlude any other VisualAsset with a lesser or equal zOrder. Two overlapping ScreenAnchors should never have the same zOrder value set.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/ScreenAnchor/default_properties
The distance from the user to any ScreenAnchor is always 0.

Absolute width and height values of a Label attached to a ScreenAnchor represent pixels on the screen. Percentage values represent the length in percent of the total screen width or height. If the content of the Label does not fit in the specified ScreenAnchor, the content should be made scrollable.

6.5 interface VisualAsset

Inherits From ARElement.

Visual Assets are the visual representations of the Features (and their Anchors) on the screen. The following VisualAssets are defined:

- 2-dimensional
  - Label: a VisualAsset specified through HTML elements
  - Fill: a colored area
  - Text: plain text
  - Image: an image

- 3-dimensional
  - Model: a 3D model

Properties:

Name	Description	Type	Multiplicity
enabled	The state of the VisualAsset	boolean	0 or 1
zOrder	Defines the Drawing order	int	0 or 1
conditions	Conditions in which the VisualAsset will be projected	Condition[]	0 or 1
Orientation	An Orientation object that describes how the VisualAsset is oriented in the Anchor's coordinate system	Orientation	0 or 1
ScalingMode	The scaling mode of the VisualAsset	ScalingMode	0 or 1

enabled

Setting the boolean flag to true (enabled) means that the VisualAsset is part of the composed scene (if the corresponding Anchor and Feature is enabled as well), setting it to false (disabled) causes the VisualAsset to be ignored in the composed scene. Defaults to true if not given.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/VisualAsset/enabled
Setting a VisualAsset's *enabled* property to false causes the VisualAsset to be ignored in the scene (i.e. they will not be visible at any time).

zOrder

Visual Assets are projected onto the screen according to their distance, with Assets of closer Anchors occluding assets of Anchors further away. To customize the drawing order, any VisualAsset has a *zOrder* property. Assets with higher zOrder values will occlude assets with lower zOrder values, independent on their distance. Only if the zOrder values of two assets are equal, the distance is taken into account again. If not given, zOrder defaults to 0.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/VisualAsset/projection_order
Visual Assets are projected onto the screen according to their distance, with Assets of closer Anchors occluding assets of Anchors further away.

conditions
A list of conditions controlling when the VisualAsset will be drawn. This is particularly useful for a Level Of Detail (LOD) control over how an anchor is represented. From further away, an Anchor might have a Label representation, when the user gets closer, the representation might change to a 3D Model. Refer to Conditions for details.

Orientation
A VisualAsset's orientation can be manually configured using an Orientation object. See Orientation-class for details.

ScalingMode
Defines how the VisualAsset will be scaled, see *Scaling VisualAssets* for details.

6.5.1 VisualAsset Types

6.5.1.1 interface VisualAsset2D

Inherits From VisualAsset.

VisualAsset2D is an abstract class that provides common properties for every concrete instance of 2-dimensional VisualAssets.

Properties:

Name	Description	Type	Multiplicity
width	The width of the VisualAsset	string	0 or 1
height	The height of the VisualAsset	string	0 or 1
orientationMode	defines how VisualAssets are automatically aligned in the underlying Anchor	string	0 or 1
backside	Customization of the back side of the VisualAsset2D	string	0 or 1

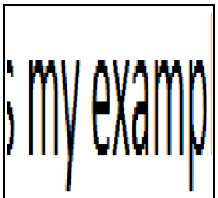
width and height
2-dimensional VisualAssets like Images do not have an implicit width and height in the composed scene. Thus, width and height can be explicitly set for 2-dimensional VisualAssets.

Both width and height can be set in absolute values (representing meters in the real world), as well as percentage values (the percentage of the total area of the underlying ARAnchor covered by the VisualAsset). If only one of width and height is set, the other value is implicitly calculated based on the aspect ratio of the VisualAsset (for Fill where an aspect ratio is not applicable, the unset value is always implicitly set to 100%). If neither width, nor height is set, width is implicitly set to 100% and height is calculated based on the aspect ratio. If both width and height are set, the VisualAsset is stretched accordingly.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/VisualAsset2D/width_and_height
If only one of width and height is set, the other value is implicitly calculated based on the aspect ratio of the VisualAsset. For Fill where an aspect ratio is not applicable, the unset value is always implicitly set to 100%. If neither width, nor height is set, width is implicitly set to 100% and height is calculated based on the aspect ratio.

Examples:

The Anchor used in the examples below is a flat polygon with a real world width of 20 meters and height of 18 meters. The Visual Asset projected onto it is a simple Text with content "This is my example Text". The examples showcase different settings of width and height; the actual measures are only approximate to show the effects of different settings.

Image	This is my example Text				
Setting	-	<width> 100% </width> <height> 100% </height>	<height> 100% </height>	<width> 5 </width>	<width> 5 </width> <height> 2 </height>
Automatically Calculated	width = 100%; height according to aspect ratio		width according to aspect ratio	height according to aspect ratio	-

If the underlying Anchor does not have an extent in width and/or height direction (like a Point (no width and height) or a LineString (no height)), the Anchor's extent in the affected direction is set to 1 meter. For example, when an Image is projected onto a Point Anchor, and the Image's width is set to 100%, the Image is rendered 1 meter wide. Height is calculated according to the aspect ratio of the Image.

orientationMode

This property controls how the VisualAsset2D is initially oriented in the Anchor's coordinate system (before roll, tilt and heading are applied) and can take on three different values: *auto* (default)*, user* and *absolute*.

Setting the value to *user* orients the VisualAsset2D towards the user. *absolute* positions the VisualAsset2D according to the coordinate system specification of the VisualAsset and the Anchor. *auto* sets the orientationMode implicitly to *absolute* when the VisualAsset2D is attached to a Trackable (or a RelativeTo Anchor referencing a Trackable), and sets it to *user* for all other cases. See *Orienting VisualAssets* for details on how this affects the orientation of a VisualAsset.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/VisualAsset2D/orientationMode
Setting the *orientationMode* value to *user* orients the VisualAsset2D towards the user. *absolute* positions the VisualAsset2D according to the coordinate system specification of the VisualAsset and the Anchor. *auto* sets the orientationMode implicitly to *absolute* when the VisualAsset2D is attached to a Trackable (or a RelativeTo Anchor referencing a Trackable), and sets it to *user* for all other cases.

backside
Backside defines how the back side of the VisualAsset should appear. Naturally, this is only relevant in case orientationMode is set to absolute.
The following values are possible for backside:

- any hex value: paints the back side of the VisualAsset2D in the color referenced by the hex value. The hex value start with # and must be given in RGBA. If backside is not given, the value defaults to #808080.
- *mirrored*: the front face of the VisualAsset is mirrored onto the back face. This effect gives the impression of the front face shining through.
- *copied*: the front side is copied onto the back side (making the front- and backside indistinguishable.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/VisualAsset2D/backSide
Setting *backside* to a hex value (#RBGA) paints the back side in that color, setting it to mirrored mirrors the front side onto the back side, setting it to copied copies the front side onto the backside.

6.5.1.1.1 class Label

Inherits From VisualAsset2D .

A Label is a VisualAsset representing a HTML view, and its content is specified in HTML. The content can either be specified using a URI pointing to a HTML file, or specified with inline HTML. Any HTML5 content is allowed, and implementations are encouraged to support the full feature set of HTML5, ECMAScript and CSS.

Properties:

Name	Description	Type	Multiplicity
href	A link to a HTML page that describes the rendered content	String	0 or 1
src	Inline HTML that will be used to describe the content	String	0 or 1
hyperlinkBehavior	A flag indicating how the implementation should handle clicks on hyperlinks in the Label	String	0 or 1
viewportWidth	An optional viewport setting	positive integer	0 or 1

href and *src*
href and src describe the content of the Label; href is a URI pointing to a HTML page that is rendered in the Label, src holds inline HTML content. If both properties are set, *src* takes precedence over *href*. At least one of the properties must be set; otherwise, the Label must be ignored.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/Label/href_and_src_precedence
src takes precedence over *href* when set.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/Label/href_and_src_required
At least one of *src* and *href* are required, otherwise the Label shall be ignored.

hyperlinkBehavior
hyperlinkBehavior allows to control how the implementation should handle clicks on hyperlinks in the Label, as well as any other location changes to the HTML document. The value can be set to either *block*, *blank* or *self*.

- *block*: Hyperlinks are not followed.
- *blank*: Hyperlinks are followed, the resulting page is opened full-screen in a new browser window. This is the default.
- *self*: Hyperlinks are followed, the resulting page is opened within the Label, replacing the original content of the Label.

The hyperlinkBehavior is independent from any onClick-event listeners set on the Label (see section 8.3.28), or the selected state of the Label (see section 6.5.4.2).

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/Label/hyperlinkBehavior
Any Label that has the *hyperlinkBehavior* set to *block* does not follow hyperlinks when clicked. The property set to *blank* causes any hyperlinks to open in full screen windows. The property set to *self* causes the content of the Label to change to the content behind the hyperlink when clicked.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/Label/hyperlinkBehavior_default
If not set, *hyperlinkBehavior* defaults to *blank*.

viewportWidth
An optional setting to control the viewport width of the Label, in pixels. This setting effectively controls the size of the content in the Label (contrary to width and height of the Label, which only describe the size of the Label itself), as well as how much space is available in the Label. If not set or set to a non-positive value, viewportWidth defaults to 256. The larger the value, the smaller the content is rendered. Implementations are allowed to set an implicit maximum threshold for viewportWidth.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/Label/viewportWidth_default
If not set or set to a negative value, *viewportWidth* defaults to *256*.

Consider an image, 256 pixels wide. Setting the viewport to 256 pixels causes the Image to horizontally span across the entire Label. Setting viewportWidth to 512 causes the Image to span across the first half of the Label, with the right half of the Label being blank.

Accessing metadata through src and href

The Feature element in ARML 2.0 allows the definition of metadata (in the name, description and metadata tag, see section 6.3). In *src* and *href* of a Label, that metadata can be referenced supplying special character sequences in the HTML. *$[name]* and *$[description]* will be replaced by the name and description of the Feature, or an empty string if not specified. To reference metadata in the metadata tag, XPath 2.0 expressions *[XML Path Language (XPath) 2.0]* enclosed in $[and] must be used (see examples below). The root node for the XPath evaluation is the metadata-tag in the Feature section. The character sequence is only replaced with the resulting node's value if an XPath evaluation returns a single TextNode, and is replaced with an empty string otherwise.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/Label/metadata_name_description
Any occurrences of *$[name]* and *$[description]* in the Label's *src* property (or the content behind *href*) shall be replaced with the name and description of the Feature's metadata the Label is linked to. If a property is missing, it shall be replaced with an empty string.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/Label/metadata_general
Any occurrences of *$[(XPath-Expression)]* shall be replaced with the corresponding metadata in the Feature the Label is linked to. *(XPath-Expression)* shall be a valid XPath 2.0 expression with the Feature's metadata node the Label is linked to as root node. The replacement will only be done when the XPath evaluation returns a single TextNode, and is replaced with an empty string otherwise.

6.5.1.1.2 class Fill

Inherits From VisualAsset2D.

Fill is used when an Anchor should appear colored. It is most useful for coloring LineStrings and Polygons. Fill can be styled using CSS styles.

Properties:

Name	Description	Type	Multiplicity
style	inline styling for the element	String	0 or 1
class	References a CSS class	String	0 or 1

style and *class*

see CSS styling for details

The following CSS properties are available for Fill:

- *color* defines the fill color of the Fill, in #RGB or #RGBA; defaults to #000000 (black)

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/Fill/color_default
If the color CSS property of Fill is not set, it defaults to #000000 (black).

6.5.1.1.3 class Text

Inherits From VisualAsset2D.

Text allows plain text to be rendered. Contrary to Label, where HTML styling can be used, Text only allows a limited set of styling options. Developers are encouraged to use Text when no HTML content is necessary, as Text does not need viewport settings to be correctly set. The size of the text is dependent on the *width* and *height* settings of the Text and will be automatically calculated.
Text can be styled using CSS styles.

Properties:

Name	Description	Type	Multiplicity
src	The text that will be rendered	String	1
style	Achieve inline styling for the element	String	0 or 1
class	References a CSS class	String	0 or 1

src
The text to be rendered. Implementations use the platform's primary font style to render the text. No control sequences such as \n or \t are available, use Label in these cases.
Replacement tags such as *$[name]* etc. can be used in the same way as for Labels (see section 6.5.1.1.1, paragraph *Accessing metadata through src and href* for details).

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/Text/metadata_name_description
Any occurrences of *$[name]* and *$[description]* in the Text's *src* property shall be replaced with the name and description of the Feature's metadata the Text is linked to. If a property is missing, it shall be replaced with an empty string.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/Text/metadata_general
Any occurrences of *$[(XPath-Expression)]* shall be replaced with the corresponding metadata in the Feature the Text is linked to. *(XPath-Expression)* shall be a valid XPath 2.0 expression with the Feature's metadata node the Text is linked to as root node. The replacement will only be done when the XPath evaluation returns a single TextNode, and is replaced with an empty string otherwise.

style and *class*
see CSS styling for details

The following CSS properties are available for Text:

- *font-color* defines the font color of the Text, in #RGB or #RGBA; defaults to black
- *background-color* defines the color of the background, in #RGB or #RGBA; defaults to transparent

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/Text/font-color_default
If the font-color CSS property of Text is not set, it defaults to #000000 (black).

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/Text/background-color_default
If the background-color CSS property of Text is not set, it defaults to #00000000 (transparent).

6.5.1.1.4 class Image

Inherits From VisualAsset2D.

Image allows an image to be rendered. Developers are encouraged to use Image instead of Label when only an image should be displayed, as Image does not need viewport settings to be correctly set. The size of the image is dependent on the *width* and *height* settings of the Image and will be automatically calculated.

Properties:

Name	Description	Type	Multiplicity
href	A URI to an image	string	1

href
A URI to the image that will be displayed on the screen. The format of the image is not restricted. If an implementation cannot render the image format of a particular image, it must ignore the entire image.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/Image/formats
In case an implementation cannot support the type of a particular image, it shall ignore the entire image.

6.5.1.2 class Model

Inherits From VisualAsset.

A Model is a Visual Asset representing a 3D Model. The format of the model is not restricted. If an implementation cannot parse the format of a particular Model, it must ignore the entire Model. Implementations are encouraged to make sure that COLLADA Common Profile is fully supported as a minimum, however this is not a requirement. Implementations are also allowed to support additional file formats, however, these will not be standardized.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/Model/formats
In case an implementation cannot support the type of a particular Model, it shall ignore the entire Model.

Properties:

Name	Description	Type	Multiplicity
href	A URI to a model file	string	1
type	The type of the Model, either normal or infrastructure	string	0 or 1
Scale	Setting the scale of the Model	Scale	0 or 1

href
The Model file itself is specified using a URI containing the source of the Model.

type
defines the role of the model in the augmented scene. Type can take on two different values, *normal* (default) and *infrastructure*.
Models with type *normal* are rendered in the composed scene. Infrastructure models are declared in the scene and used for occlusion detection, but are not visible in the scene (for example, a real world building might be modeled as an infrastructure model, so it's not rendered on the screen, but it is used to virtually occlude other VisualAssets behind the real world building).

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/Model/type
Models of type *normal* are visible in the scene. Models of type *infrastructure* are not rendered in the scene and are solely used for occlusion detection.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/Model/type_default
If the type property of a Model is not set, it defaults to *normal*.

Scale
allows scaling of the Model; see class Scale for details.

6.5.1.2.1 class Scale

Scale allows scaling of the Model along the x-, y- and z-axis. The values default to 1 if not specified. As with orientations, applying scales does not affect the axes of the Model itself, only the object is scaled.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/Scale/defaults
If not set, the x, y and z values default to 1.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/Scale/axis
Applying scales does not affect the axis and their dimensions, only the object itself is scaled.

6.5.2 Orienting VisualAssets

Depending on the dimension of the VisualAsset and dimension of the ARAnchor it is attached to, different rules apply how VisualAssets are rendered on ARAnchors. The orientation is also dependent on the properties *orientationMode* (VisualAsset2D only) and *Orientation*.

6.5.2.1 Orienting VisualAsset2Ds

VisualAsset2Ds come with an *orientationMode* property (see interface VisualAsset2D), which controls how the VisualAsset is oriented.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/AutomaticOrientation_VisualAssets/2D
The *orientationMode* of a VisualAsset2D and the dimension of the underlying ARAnchor control the automatic orientation of VisualAsset2Ds. Any implementation shall implement the orientation of VisualAsset2Ds as specified in section 6.5.2.1.

Case 1: Underlying ARAnchor is of Dimension 0, orientationMode = "user"

In this case, the center point of the VisualAsset2D is placed right onto the position of the ARAnchor in 3D space (either the geospatial location for Point-Geometries, or the center point of the Trackable for Trackables). The upper face of the VisualAsset2D is always oriented towards the user's current location. The upper and lower edges of the VisualAsset2D run parallel to the earth's surface in case of a Point-Geometry, and parallel to the Trackable's surface in case of a Trackable.

Case 2: Underlying ARAnchor is of Dimension 1, orientationMode = "user"

The VisualAsset2D runs along the defined LineString. The horizontal centerline of the Asset (the line being equidistant from the top and bottom of the VisualAsset) is placed onto the defined LineString. The horizontal center of the 2-dimensional VisualAsset (the point being equidistant from the center point of the left and right edge of the VisualAsset) is placed on the point being equidistant from the left and right end of the LineString (the origin of the coordinate system of the Anchor). This ensures that the VisualAsset expands from the center of the LineString, equally in both directions.

For each line segment of the LineString (the lines between the positions that form the LineString), the VisualAsset is directly facing the user. For each segment, the shortest distance from the LineString segment to the user's viewing point is calculated, the resulting vector forms the normal vector of the plane the VisualAsset will be drawn into for this particular LineString segment (see figure below).

Remark: In case a LineString consists of more than one segment, the tie points of the segments might cause issues when the VisualAsset is rendered onto them. It is up to the implementation to smooth these artifacts.

Case 3: Underlying ARAnchor is of Dimension 2, orientationMode = "user"

The center of the VisualAsset2D is placed in the center of the BoundingRectangle of the ARAnchor, which can be considered the center of the Polygon forming the ARAnchor (see Local Coordinate System of a Polygon for details). The lower and upper edges and the left and right edges of the VisualAsset respectively are parallel to the lower and upper edges and the left and right edges of the BoundingRectangle of the Polygon respectively. The front face of the VisualAsset2D faces the user.

In case the Polygon and the VisualAsset are not of the same shape, the Polygon's boundaries will cut off any areas of the VisualAsset that do not lie within the Polygon's boundaries. This also applies to any holes in the Polygon defined by *interior LinearRings*.

Case 4: Underlying ARAnchor is of Dimension 0, orientationMode = "absolute"

This is handled in the same way as case 1, with the exception that the VisualAsset is placed into the x/z plane of the coordinate system of the Anchor, regardless of the user's position. The top and bottom edges of the VisualAsset are parallel to the x-axis; the left and right edges of the VisualAsset are parallel to the y-axis of the ARAnchor’s coordinate system. The top edge of the VisualAsset is located in the positive y-half; the right edge of the VisualAsset is located in the positive x-half.

Case 5: Underlying ARAnchor is of Dimension 1, orientationMode = "absolute"

The basic setup is equal to case 2. However, instead of calculating the plane as facing the user, the VisualAsset's left and right edges are placed parallel to the earth's surface for LineStrings associated with a Geometry, and parallel to the Trackable's surface for LineStrings associated with Trackables. This ensures the VisualAsset appears to be *lying flat on top of the LineString* when viewed from above.

The VisualAsset's front face is always facing up, whereat up is defined as:
When viewing the first LineSegment in a way that the first specified vertex is on the left side, and the second vertex is on the right side, the side facing the viewer is the upper side.

Case 6: Underlying ARAnchor is of Dimension 2, orientationMode = "absolute"

This is handled in the same way as in case 3, with the exception that the VisualAsset's front face is always facing up (depending on the order the vertices of the Polygon were specified).

6.5.2.2 Orienting 3D VisualAssets

Case 1: Underlying ARAnchor is of Dimension 0

3-dimensional assets are projected into the coordinate system of a 0-dimensional location. Both the Model and the ARAnchor use the same coordinate system origin and the same axis alignment.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/AutomaticOrientation_VisualAssets/3D_dim_0
Any implementation shall implement the orientation of 3D VisualAssets (Models) attached to 0-dimensional ARAnchors as specified in section 6.5.2.2.

Case 2: Underlying ARAnchor is of Dimension 1 or 2

3-dimensional assets cannot be attached to 1- or 2-dimensional Anchors and must be ignored in these cases.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/AutomaticOrientation_VisualAssets/3D_dim_1_2
3D VisualAssets (Models) attached to 1- or 2-dimensional ARAnchors shall be ignored.

6.5.2.3 class Orientation - Manual Orientation of VisualAssets

The Orientation class allows to manually adjusting the orientation of a VisualAsset in 3D space after it was automatically oriented according to the above rules.

Properties:

Name	Description	Type	Multiplicity
roll	rotation around a certain rotation axis, see below for details	double	0 or 1
tilt	rotation around a certain rotation axis, see below for details	double	0 or 1
heading	rotation around a certain rotation axis, see below for details	double	0 or 1

The orientation object has 3 properties, *roll*, *tilt* and *heading*, which define rotations of the VisualAsset in 3 directions. The following rules apply:

- The rotation is applied using static axes (meaning that the axes are not transformed, only the object inside the coordinate system is rotated)
- The orientation steps are executed in the following order: roll - tilt - heading
- roll, tilt and heading are specified in degrees from -180 to 180.
-

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/ManualOrientation_VisualAssets/order
The orientation is executed in the following order: roll – tilt – heading

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/ManualOrientation_VisualAssets/axes
Applying orientation does not affect the axis and their dimensions, only the object itself is rotated.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/ManualOrientation_VisualAssets/application
The *orientationMode* of a VisualAsset and the dimension of the underlying ARAnchor control the manual orientation of VisualAssets. Any implementation shall implement the manual orientation of VisualAssets as specified in section 6.5.2.3.

Depending on the orientationMode and the type of the Anchor, the rotations are applied slightly differently:

Case 1: 0-dimensional Anchor, orientationMode absolute or VisualAsset is 3-dimensional

- roll rotates the VisualAsset about the y axis. A positive rotation is clockwise around the y-axis when viewed from the origin of the coordinate system looking along the positive axis.
- tilt rotates the VisualAsset about the *x* axis. A positive rotation is clockwise around the *x*-axis when viewed from the origin of the coordinate system looking along the positive axis.
- heading rotates the VisualAsset about the *z* axis. A positive rotation is clockwise around the *z*-axis when viewed from the origin of the coordinate system looking along the positive axis.

Case 2: 0-dimensional Anchor, orientationMode user

- tilt rotates the VisualAsset about the line parallel to the (earth's or Trackable's) surface, running through the center of the VisualAsset (the user will see the VisualAsset flipping towards or away from him). A positive rotation moves the top towards the user at first.
- heading rotates the VisualAsset about the line connecting the center of the screen with the center of the VisualAsset (the user will see the VisualAsset rotating in the plane that is facing him). A positive rotation is clockwise when viewed from the user looking towards the VisualAsset.
- roll rotates the VisualAsset about the axis that is perpendicular to the other two axes specified above, pointing away from the surface. A positive rotation moves the right edge of the VisualAsset towards the user first.

Case 3: 1-dimensional Anchor

- roll does not apply
- tilt rotates the VisualAsset about each LineSegment of the LineString. A positive rotation is to the right when viewed from the start of each LineSegment towards the end of the LineSegment.
- heading does not apply

Case 4: 2-dimensional Anchor

- roll does not apply
- tilt does not apply
- heading rotates the VisualAsset inside the plane the Polygon is forming around the center of the VisualAsset (and the coordinate system of the Anchor). A positive rotation is clockwise when viewed from above the Polygon.

6.5.3 class ScalingMode - Scaling VisualAssets

VisualAssets appear smaller when their attached Anchors are further away, and appear bigger when the user moves towards the Anchor.

Consider, for example, a Polygon geometry representing a billboard on the street, which measures 20x10 meters, where a Label is attached to it (with width set to 100%). As the Anchor (and thus the Label) is scaled naturally, the further away the user, the smaller the Label is rendered, so it always fits the billboard. This is called *natural scaling*.

However, as the user walks away from the billboard, pretty soon the Label will become almost invisible, as a width of 20 meters, seen from a distance of 1000 meters, will appear very tiny. Contrary, if standing right in front of the billboard, the Label will obstruct the entire screen, occluding any other objects.

To overcome this, a Visual Asset can be scaled in *custom* mode. In custom scaling mode, a *minScalingDistance* and *maxScalingDistance* are supplied along with a scalingFactor. *min*- and *maxScalingDistance* specify the distance of the user to the anchor of the VisualAsset (precisely: the distance of the origin of the coordinate system of the anchor) where custom scaling should start and stop. Outside of those boundaries, no scaling will apply. *scalingFactor* controls how much VisualAssets are scaled between those two boundaries.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/Scaling_VisualAssets/minMaxScalingDistance
In *custom* scaling mode, *minScalingDistance* and *maxScalingDistance* specify the distance of the user to the origin of the coordinate system of the ARAnchor the VisualAsset is attached to when scaling should start (min) and stop (max). Before *minScalingDistance* and behind *maxScalingDistance*, no scaling applies.

For example, setting a minScalingDistance to 10 meters causes the Label to be rendered as if the billboard would be 10 meters away, even if the user were standing closer. Similarly, setting a maxScalingDistance to 100 meters causes the Label to be rendered as if the billboard would be 100 meters away, even if the user were standing a lot further away. Between 10 and 100 meters, natural scaling is applied if no scalingFactor is set.

The amount of scaling between min and maxScalingDistance can be controlled using a scaling factor. scalingFactor specifies the size of the VisualAsset at maxScalingDistance in percentage of the size of the VisualAsset at minScalingDistance. Setting scalingFactor to 0.5, minScalingDistance to 10 meters and maxScalingDistance to 100 meters causes the VisualAsset to lose 50% of its size (on the screen) between 10 and 100 meters distance.

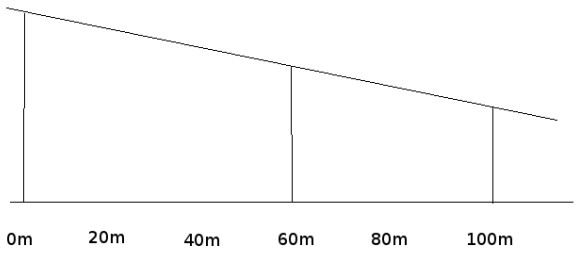
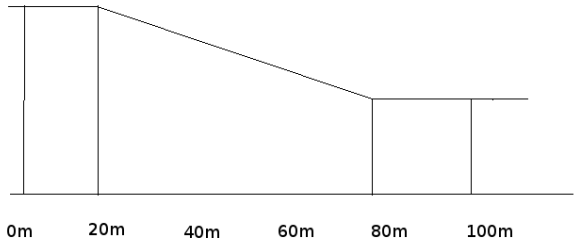
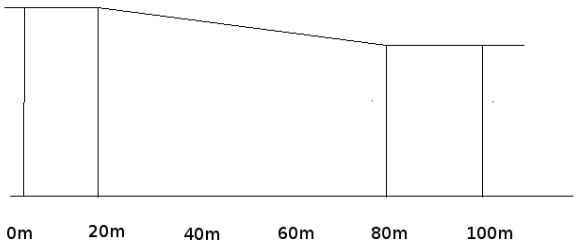
Requirement
http://www.opengis.net/spec/arml/2.0/req/core/Scaling_VisualAssets/scalingFactor
In *custom* scaling mode, *scalingFactor* specifies the size of the VisualAsset at *maxScalingDistance* in

percentage of the size of the VisualAsset at *minScalingDistance*.

If both *minScalingDistance* and *maxScalingDistance* are set to the same value and *scalingFactor* is omitted, the VisualAsset will appear at the same size on the screen, regardless of the distance. If *scalingFactor* is set to 50%, the VisualAsset has a constant size until *maxScalingDistance* is reached, then drops to 50% of the size and keeps this size for any distances further than *maxScalingDistance*.

Illustration:

The following three diagrams show the effect of applying different settings to the size of a VisualAsset on the screen. The horizontal axis of the diagram shows the distance from the user to the Visual Asset, the vertical axes shows the size of the VisualAsset on the screen.

	Natural Scaling: The Visual Asset has a certain natural size at distance 0 and is scaled according to natural scaling.
	Custom Scaling: minScalingDistance: 20 maxScalingDistance: 80 scalingFactor: not set The VisualAsset is not scaled between 0 and 20 meters distance (at 20 meters, it is at its natural size), then naturally scaled between 20 and 80 meters. At 80 meters distance, it takes on its natural size. Beyond 80 meters distance, the VisualAsset does not change its screen size.
	Custom Scaling: minScalingDistance: 20 maxScalingDistance: 80 scalingFactor: 0.75 Scaling behaves similar to the previous example. The main difference is that at 80 meters distance, the VisualAsset is now at 75% of its size at 20 meters.

Example:

Natural scaling of a Label, with viewing distances 10 meters, 20 meters, 30 meters and 40 meters	
Custom scaling of a Label, with viewing distances 10 meters, 20 meters, 30 meters and 40 meters, minScalingDistance set to 20, maxScalingDistance set to 30 and scalingFactor omitted.	

In the second example, natural scaling applies between 20 and 30 meters distance. If the user is closer than 20 meters, the Label is rendered on the screen as if the Anchor would be 20 meters away (minScalingDistance set to 20 meters). Similarly, if the user is further than 30 meters away, the Label is rendered on the screen as if the Anchor would be 30 meters away (maxScalingDistance set to 30).

The scaling mode calculations are applied after the VisualAsset was positioned, scaled (according to width and height for VisualAsset2D, and Scaling for Model) and aligned according to the orientation settings.

Properties:

Name	Description	Type	Multiplicity
type	The type of the scaling mode, either "natural" or "custom"	string	1
minScalingDistance	The distance the natural scaling effect should start	double	0 or 1
maxScalingDistance	The distance the natural scaling effect should stop	double	0 or 1
scalingFactor	The size of the VisualAsset at maxScalingDistance in percentage of the size of the VisualAsset at minScalingDistance	double	0 or 1

type
Either *natural* or *custom*

minScalingDistance
The distance the natural scaling effect should start. Should only be specified when type is set to custom and is ignored for natural. If not specified or set to a negative value, custom scaling acts as if the value would be set to 0. Must be less than or equal to *maxScalingDistance*.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/Scaling_VisualAssets/minScalingDistance_default
If not set, *minScalingDistance* is implicitly set to 0. *minScalingDistance* shall be less than or equal to *maxScalingDistance*.

maxScalingDistance
The distance the natural scaling effect should stop. Must be specified when type is set to custom and is ignored for natural. Must be greater than or equal to *minScalingDistance*.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/Scaling_VisualAssets/maxScalingDistance_ignored
maxScalingDistance shall be ignored when *natural* scaling is used, and shall be greater than or equal to *minScalingDistance*.

scalingFactor
scalingFactor is a percentage value (0 <= scalingFactor <= 1) that defines how rapidly the VisualAssets should be scaled between min and maxScalingDistance. It specifies the size of the VisualAsset at maxScalingDistance in percentage of the size of the VisualAsset at minScalingDistance. If minScalingDistance is not supplied, it must be temporarily set to 1 meter for the purpose of the scalingFactor calculations. scalingFactor should only be specified when type is set to custom and maxScalingDistance is set, and is ignored otherwise.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/Scaling_VisualAssets/scalingFactor_ignored
scalingFactor shall be ignored when *natural* scaling is used.

6.5.4 interface Condition

Inherits from ARElement.

Depending on the situation, certain VisualAssets might be visible on the screen at different times. Consider a mountain with a mountain hut on its summit, which should be remodeled. The mountain hut has a representation as a 3D model, showing the shape of the mountain hut in the future. However, from further away, the 3D model is not visible at all. Hikers starting at the valley ground, however, want to see a big Label indicating where the Mountain hut is actually located.

The following conditions are available:

- *distance* (min and max distance)
- *selected* (true/false)

If multiple conditions are supplied for a particular VisualAsset, all these conditions must yield true for the VisualAsset to be visible.

Requirement

http://www.opengis.net/spec/arml/2.0/req/core/Condition/multiple
If multiple Conditions are set for a VisualAsset, they all shall yield true for the VisualAsset to be rendered.

Remark: To achieve a "condition1 or condition2" situation, the VisualAsset must be duplicated (asset1 and asset2), where asset1 is tied to condition1, and asset2 is tied to condition2.

6.5.4.1 class DistanceCondition

Inherits from Condition.

DistanceCondition allows VisualAssets to be activated and deactivated based on the distance of the user to the anchor (precisely: the origin of the coordinate system of the anchor).

Properties:

Name	Description	Type	Multiplicity
max	The maximum distance the VisualAsset will be visible for	double	0 or 1
min	The minimum distance the VisualAsset will be visible for	double	0 or 1

max

denotes the maximum distance the VisualAsset will be visible at, in meters. For example, if it is set to 100, VisualAssets attached to Anchors with a distance of more than 100 meters are not visible.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/Condition/Distance/max
VisualAssets further away from the user than the distance specified in *max* shall not be visible.

min

denotes the minimum distance the VisualAsset will be visible at, in meters. For example, if it is set to 100, VisualAssets attached to Anchors with a distance of less than 100 meters are not visible.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/Condition/Distance/min
VisualAssets closer to the user than the distance specified in *min* shall not be visible.

If both min and max are set, both conditions must yield true for the visual asset to be rendered

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/Condition/Distance/min_and_max
If both *min* and *max* are set, both conditions shall yield true for the VisualAsset to be visible.

6.5.4.2 class SelectedCondition

Inherits from Condition.

The selected condition allows VisualAssets to be activated and deactivated based on the selected-status of the Feature or Anchor.

An Anchor is considered *selected* if one of its VisualAssets has been selected in the most recent selection process. It is expected that for most implementations, a click or touch on a VisualAsset will be considered a selection of that VisualAsset, however, the definition is implementation- and platform-specific. In turn, a Feature is considered selected if one of its Anchors is selected.

Properties:

Name	Description	Type	Multiplicity
listener	The element type the selected-condition is listening for, either "feature" or "anchor"	String	0 or 1
selected	The selected state the VisualAsset should be visible	boolean	1

listener
One of *feature* or *anchor*, defaults to *anchor*.
If *listener* is set to *feature*, the selected-condition listens on the selected state of the Feature the VisualAsset is attached (i.e. also the selection of another Anchor that is attached to the same Feature can trigger the select-state of the particular Feature). If *listener* is set to *anchor*, the selected-condition listens on the selected state of the Anchor the VisualAsset is attached to.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/Condition/Selected/listener_default
If *listener* is not set, it defaults to *anchor*.

selected
If set to true, the VisualAsset is only visible when the Anchor or Feature (see *listener*) is currently selected; if set to false, it is only visible when the Anchor or Feature is not currently selected.

Requirement
http://www.opengis.net/spec/arml/2.0/req/core/Condition/Selected/selected
If selected is set to true, the VisualAsset is only visible when the listener is in selected state. When set to false, the VisualAsset is only visible when the listener is not in selected state.

7 ARML 2.0 – XSD and XML Serialization (normative)

Requirements Class	
http://www.opengis.net/spec/arml/2.0/req/model	
Target Type	Encoding

Requirement
http://www.opengis.net/spec/arml/2.0/req/model/general/xsd_verification
Any ARML 2.0 XML encoding shall validate correctly against the XSD defined in section 7.

This chapter defines an XML serialization of the ARML 2.0 object model. The XML serialization is defined in XSD. The following XSD header, namespaces and imports are used:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.opengis.net/arml/2.0"
  xmlns="http://www.opengis.net/arml/2.0"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:gml=" http://www.opengis.net/gml/3.2"
  elementFormDefault="qualified"
  version="2.0">

  <!-- Xlink import -->
  <xsd:import namespace="http://www.w3.org/1999/xlink"
    schemaLocation="http://www.w3.org/1999/xlink.xsd" />
  <!-- GML import -->
  <xsd:import namespace="http://www.opengis.net/gml/3.2"
    schemaLocation="http://schemas.opengis.net/gml/3.2.1/gml.xsd" />
```

7.1 Document Structure

The root element of the document is <*arml*>, which contains the following elements:

- The <*ARElements*> element, containing a list of *ARElement* objects.
- Multiple optional <*style*> elements, including and optional *type*-attribute that allows the specification of the style-mime type (typically *text/css*).
- Multiple optional <*script*> elements, including and optional *type*-attribute that allows the specification of the script-mime type (typically *text/javascript*).

Requirement
http://www.opengis.net/spec/arml/2.0/req/model/general/root_element
<arml> is the root element of the encoding and shall be present.

Requirement
http://www.opengis.net/spec/arml/2.0/req/model/ARElement/container
Only objects derived from ARElement are allowed as immediate child elements in the <ARElements> tag.

XML Example (shortest possible ARML document):

```
<?xml version="1.0" encoding="UTF-8"?>
<arml xmlns=http://www.opengis.net/arml/2.0
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/arml/2.0
                       http://schemas.opengis.net/arml/2.0/arml.xsd">
  <ARElements>
  </ARElements>
</arml>
```

XML Example:

```
<arml xmlns="http://www.opengis.net/arml/2.0">
  <ARElements>
    <Feature id="myFeature">
      <name>My first Feature</name>
      <anchors>
        <gml:Point gml:id="myPoint">
          <gml:pos>48.123 13.456</gml:pos>
        </gml:Point>
      </anchors>
    </Feature>
  </ARElements>

  <style type="text/css">
    <![CDATA[
      ... CSS style definitions of any Visual Assets
    ]]>
  </style>

  <script type="text/ecmascript">  <!--might also be javascript and other
derivatives -->
    <![CDATA[
      ... ECMAScript goes here ...     ]]>
  </script>
</arml>
```

XSD:

```
<xsd:complexType name="ArmlType">
  <xsd:sequence>
    <xsd:element name="ARElements" maxOccurs="1" minOccurs="1">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element ref="ARElement" minOccurs="0" maxOccurs="unbounded"
/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="style" maxOccurs="unbounded" minOccurs="0">
      <xsd:complexType>
        <xsd:simpleContent>
          <xsd:extension base="xsd:string">
            <xsd:attribute name="type" type="xsd:string" use="optional" />
            <xsd:attribute ref="xlink:href" use="optional" />
          </xsd:extension>
        </xsd:simpleContent>
```

```
      </xsd:complexType>
    </xsd:element>

    <xsd:element name="script" maxOccurs="unbounded" minOccurs="0">
      <xsd:complexType>
        <xsd:simpleContent>
          <xsd:extension base="xsd:string">
            <xsd:attribute name="type" type="xsd:string" use="optional" />
            <xsd:attribute ref="xlink:href" use="optional" />
          </xsd:extension>
        </xsd:simpleContent>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>

<xsd:element name="arml" type="ArmlType" />
```

7.2 interface ARElement

Requirement
http://www.opengis.net/spec/arml/2.0/req/model/ARElement/interface
ARElement only serves as an interface for other objects defined in the encoding and may not be used directly in the encoding.

Requirement
http://www.opengis.net/spec/arml/2.0/req/model/ARElement/id
Any object derived from ARElement may contain an optional id attribute used to uniquely reference the ARElement by other objects. In case the attribute is set, it shall be unique within the document.

XSD:

```
<xsd:complexType name="ARElementType" abstract="true">
  <xsd:attribute name="id" type="xsd:ID" use="optional" />
</xsd:complexType>

<xsd:element name="ARElement" abstract="true" type="ARElementType" />
```

7.3 class Feature

A Feature contains a list of Anchors, which can either be defined directly in the *anchors*-tag, or referenced using the *anchorRef* tag. Both ways can be mixed within one Feature, and a Feature can have an arbitrary number of Anchors.
If an Anchor is referenced with *anchorRef*, the URI to the Anchor is specified in the xlink:href attribute.

Requirement
http://www.opengis.net/spec/arml/2.0/req/model/Feature/anchors/relative
Anchors attached to a Feature by reference (i.e. using *anchorRef*) shall be referenced via a URI using the xlink:href attribute.

XSD:

```
<xsd:complexType name="FeatureType">
  <xsd:complexContent>
    <xsd:extension base="ARElementType">
      <xsd:sequence>
        <xsd:element name="name" type="xsd:string" maxOccurs="1"
minOccurs="0" />
        <xsd:element name="description" type="xsd:string" maxOccurs="1"
minOccurs="0" />
        <xsd:element name="enabled" type="xsd:boolean" maxOccurs="1"
minOccurs="0" />
        <xsd:element name="metadata" maxOccurs="1" minOccurs="0">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:any processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="anchors" maxOccurs="1" minOccurs="0">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element ref="Anchor" minOccurs="0" maxOccurs="unbounded"
/>
              <xsd:element name="anchorRef" maxOccurs="unbounded"
minOccurs="0">
                <xsd:complexType>
                  <xsd:attribute ref="xlink:href" use="required" />
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="Feature" type="FeatureType"
substitutionGroup="ARElement" />
```

XML Example:

```
<Feature id="ferrisWheel">
  <name> Ferris Wheel</name>
  <enabled>true</enabled>
  <metadata>
    <constructed>1896-1897</constructed>
    <height>64,75</height>
  </metadata>
```

```
  <anchors>
    <!-- either defined directly in the tag -->
    <Geometry>
      ...
    </ Geometry>
    <!-- or referenced -->
    <anchorRef xlink:href="#myAnchor" />
  </anchors>
</Feature>
```

7.4 interface Anchor

Requirement
http://www.opengis.net/spec/arml/2.0/req/model/Anchor/interface
Anchor only serves as an interface for other objects defined in the encoding and may not be used directly in the encoding.

XSD:

```
<xsd:complexType name="AnchorType" abstract="true">
  <xsd:complexContent>
    <xsd:extension base="ARElementType">
      <xsd:sequence>
        <xsd:element name="enabled" type="xsd:boolean" maxOccurs="1"
minOccurs="0" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="Anchor" type="AnchorType" abstract="true"
substitutionGroup="ARElement" />
```

7.4.1 interface ARAnchor

Requirement
http://www.opengis.net/spec/arml/2.0/req/model/ARAnchor/interface
ARAnchor only serves as an interface for other objects defined in the encoding and may not be used directly in the encoding.

An ARAnchor contains a list of VisualAssets, which can either be defined directly in the *assets*-tag, or referenced using the *assetRef* tag. Both ways can be mixed within one ARAnchor, and an ARAnchor can have an arbitrary number of VisualAssets.
In case VisualAssets are referenced with *assetRef*, the URI to the VisualAsset is specified in the xlink:href attribute.

Requirement
http://www.opengis.net/spec/arml/2.0/req/model/ARAnchor/relative
VisualAssets attached to an Anchor by reference (i.e. using *assetRef*) shall be referenced via a URI using the xlink:href attribute.

XSD:

```
<xsd:complexType name="ARAnchorType" abstract="true">
  <xsd:complexContent>
    <xsd:extension base="AnchorType">
      <xsd:sequence>
        <xsd:element name="assets" maxOccurs="1" minOccurs="1">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element ref="VisualAsset" maxOccurs="unbounded"
minOccurs="0" />
              <xsd:element name="assetRef" maxOccurs="unbounded"
minOccurs="0">
                <xsd:complexType>
                  <xsd:attribute ref="xlink:href" use="required" />
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="ARAnchor" type="ARAnchorType" abstract="true"
substitutionGroup="Anchor" />
```

7.4.1.1 class Geometry

XSD:

```
<xsd:complexType name="GeometryType">
  <xsd:complexContent>
    <xsd:extension base="ARAnchorType">
      <xsd:choice>
        <xsd:element ref="gml:Point" />
        <xsd:element ref="gml:LineString" />
        <xsd:element ref="gml:Polygon" />
      </xsd:choice>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="Geometry" type="GeometryType"
substitutionGroup="ARAnchor" />
```

Example:

```
<Feature id="myFeature">
  <anchors>
    <Geometry>
      <enabled>true</enabled>
```

```
      <assets>
        ...
      </assets>
      <gml:Point gml:id="point1">
        <gml:pos>1 2</gml:pos>
      </gml:Point>
    </Geometry>
  </anchors>
</Feature>
```

7.4.1.1.1 interface GMLGeometries

Requirement
http://www.opengis.net/spec/arml/2.0/req/model/GMLGeometries/interface
GMLGeometries only serves as an interface for other objects defined in the encoding and may not be used directly in the encoding.

7.4.1.1.2 class Point

Requirement
http://www.opengis.net/spec/arml/2.0/req/model/Point/xsd
The Point class in ARML reuses the Point model, including the XSD, from the [GML Specification].

XML Example:

```
<gml:Point gml:id="myPointWithAltitudeOfUser">
  <gml:pos>
    47.48 13.14
  </gml:pos>
</gml:Point>

<gml:Point gml:id="myPointWithExplicitAltitude" srsDimension="3">
  <gml:pos>
    47.48 13.14 520
  </gml:pos>
</gml:Point>
```

7.4.1.1.3 class LineString

Requirement
http://www.opengis.net/spec/arml/2.0/req/model/LineString/xsd
The LineString class in ARML reuses the LineString model, including the XSD, from the [GML Specification].

XML Example:

```
<gml:LineString gml:id="myLineString">
  <gml:posList>
    47.48 13.14 48.49 14.15
  </gml:posList>
</gml:LineString>
```

7.4.1.1.4 class Polygon

Requirement
http://www.opengis.net/spec/arml/2.0/req/model/Polygon/xsd
The Polygon class in ARML reuses the Polygon model, including the XSD, from the [GML

Specification].

As a convention, the vertices of the Polygon (especially the vertices of the exterior LinearRing) should be specified in counter-clockwise direction to correctly define the VisualAsset's front face. See Orienting VisualAssets for details.

Requirement
http://www.opengis.net/spec/arml/2.0/req/model/GMLGeometries/LinearRing/order
A LinearRing's coordinate tuples shall be specified in counter-clockwise order.

XML Example:

```
<gml:Polygon gml:id="myPolygon">
  <gml:exterior>
    <gml:LinearRing>
      <gml:posList>
        47.48 13.14 48.49 14.15 48.49 14.13 47.48 13.14
       </gml:posList>
    </gml:LinearRing>
  </gml:exterior>
  <gml:interior>
    <gml:LinearRing>
      <gml:posList>
        48.00 14.00 48.01 14.01 48.01 13.99 48.00 14.00
       </gml:posList>
    </gml:LinearRing>
  </gml:interior>
  <gml:interior>
    <gml:LinearRing>
      ...
    </gml:LinearRing>
  </gml:interior>
</gml:Polygon>
```

7.4.1.2 Trackable and Tracker

7.4.1.2.1 class Tracker

The src and uri properties are specified in xlink:href attributes, see the XSD below.

XSD:

```
<xsd:complexType name="TrackerType">
  <xsd:complexContent>
    <xsd:extension base="ARElementType">
      <xsd:sequence>
        <xsd:element name="uri" maxOccurs="1" minOccurs="1">
          <xsd:complexType>
            <xsd:attribute ref="xlink:href" use="required" />
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="src" maxOccurs="1" minOccurs="0">
          <xsd:complexType>
```



```
            <xsd:attribute ref="xlink:href" use="required" />
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="Tracker" type="TrackerType"
substitutionGroup="ARElement" />
```

XML Example:

```
<!-- a generic image Tracker -->
<Tracker id="myGenericImageTracker">
  <uri xlink:href="http://www.opengis.net/arml/tracker/genericImageTracker"
/>
</Tracker>

<!-- a generic image Tracker operating on a set of image targets supplied
via a zip file -->
<Tracker id="myGenericImageTrackerWithZip">
  <uri xlink:href="http://www.opengis.net/arml/tracker/genericImageTracker"
/>
  <src xlink:href="http://www.myserver.com/myTargets/myTargets.zip" />
</Tracker>

<!-- a custom Tracker -->
<Tracker id="myCustomTracker">
  <uri xlink:href="http://www.myServer.com/myTracker" />
  <src xlink:href="http://www.myServer.com/myTrackables/binary.file" />
</Tracker>
```

7.4.1.2.2 class Trackable

XSD:

```
<xsd:complexType name="TrackableType">
  <xsd:complexContent>
    <xsd:extension base="ARAnchorType">
      <xsd:sequence>
        <xsd:element name="config" maxOccurs="unbounded" minOccurs="1">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="tracker" maxOccurs="1" minOccurs="1">
                <xsd:complexType>
                  <xsd:attribute ref="xlink:href" use="required" />
                </xsd:complexType>
              </xsd:element>
              <xsd:element name="src" type="xsd:string" maxOccurs="1"
minOccurs="1" />
            </xsd:sequence>
            <xsd:attribute name="order" type="xsd:int" use="optional" />
```

```
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="size" type="xsd:double" maxOccurs="1"
minOccurs="0" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="Trackable" type="TrackableType"
substitutionGroup="ARAnchor" />
```

XML Example:

```
<!-- using the trackers specified above -->
<!-- a png image tracked with the generic image tracker -->
<Trackable id="myBirdTrackable">
  <config>
    <tracker xlink:href="#myGenericImageTracker" />
    <src>http://www.myserver.com/myTrackables/bird.png</src>
  </config>
  <size>0.2</size> <!-- in real word dimensions, the bird image is 20 cm
wide -->
</Trackable>

<!-- a jpg image tracked with the generic image tracker operating on a zip
file-->
<Trackable id="myBirdTrackableInZip">
  <config>
    <tracker xlink:href="#myGenericImageTrackerWithZip" />
    <src>/images/bird.png</src>
  </config>
  <size>0.2</size>
</Trackable>

<!-- a jpg image tracked with the generic image tracker operating on a zip
file-->
<Trackable id="myCustomBirdTrackable">
  <config>
    <tracker xlink:href="#myCustomTracker" />
    <src>bird</src> <!-- the custom tracker is supposed to understand the
ID "bird" in the Tracker's binary container -->
  </config>
  <size>0.2</size>
</Trackable>

<!—a Trackable that can be tracked in two different ways, preferably with a
custom implementation that takes a binary file, and if this configuration
is not available, a generic imagetracker should be used-->
<Trackable id="myTrackable">
  <config order="1">
    <tracker xlink:href="#myCustomSuperSpeedyTracker" />
    <src>http://www.myserver.com/myTrackables/bird.dat</src>
  </config>
```

```
  <!–fallback -->
  <config order="2">
    <tracker xlink:href="#myGenericImageTracker" />
    <src>http://www.myserver.com/myTrackables/bird.png</src>
  </config>
  <size>0.2</size>
</Trackable>
```

7.4.1.3 class RelativeTo

Requirement
http://www.opengis.net/spec/arml/2.0/req/model/RelativeTo/ref
A RelativeTo element shall have its *ref* property set to either reference another ARAnchor (except LineStrings), another Model or the user's position (using #user), using the xlink:href attribute.

XSD:

```
<xsd:complexType name="RelativeToType">
  <xsd:complexContent>
    <xsd:extension base="ARAnchorType">
      <xsd:sequence>
        <xsd:element name="ref" maxOccurs="1" minOccurs="1">
          <xsd:complexType>
            <xsd:attribute ref="xlink:href" use="required" />
          </xsd:complexType>
        </xsd:element>
        <xsd:choice>
          <xsd:element ref="gml:Point" />
          <xsd:element ref="gml:LineString" />
          <xsd:element ref="gml:Polygon" />
        </xsd:choice>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="RelativeTo" type="RelativeToType"
substitutionGroup="ARAnchor" />
```

Example (to mark the outline of a Trackable):

```
<Trackable id="myTrackable">
   <size>5</size> <!-- assuming a square Trackable for this example-->
   ...
</Trackable>

<RelativeTo>
  <ref xlink:href="#myTrackable" />
  <gml:LineString gml:id="trackableOutline">
    <gml:posList dimension="3"> <!-- will describe the outline of the
square marker (2.5 meters from origin to top, bottom, left and right edge -
->
      2.5 2.5 0 2.5 -2.5 0 -2.5 -2.5 0 -2.5 2.5 0 2.5 2.5 0
    </gml:posList>
```



```
  </gml:LineString>
</ RelativeTo>
```

7.4.2 class ScreenAnchor

XSD:

```
<xsd:complexType name="ScreenAnchorType">
  <xsd:complexContent>
    <xsd:extension base="AnchorType">
      <xsd:sequence>
        <xsd:element name="style" type="xsd:string" maxOccurs="1"
minOccurs="0" />
        <xsd:element name="class" type="xsd:string" maxOccurs="1"
minOccurs="0" />
        <xsd:element name="assets" maxOccurs="1" minOccurs="1">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element ref="Label" maxOccurs="unbounded" minOccurs="0"
/>
              <xsd:element name="assetRef" maxOccurs=" unbounded "
minOccurs="0">
                <xsd:complexType>
                  <xsd:attribute ref="xlink:href" use="required" />
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="ScreenAnchor" type="ScreenAnchorType"
substitutionGroup="Anchor" />
```

Example (a Feature also contains a ScreenAnchor showing some information on the Feature):

```
<Feature id="myPlacemark">
  <anchors>
    <ScreenAnchor style="bottom:0; left:0; width: 100%;">
      <!-- area spans the entire screen width, and is located at the bottom
of the screen; top is dynamic -->
      <assets>
        <Label>
          <src><![CDATA[<div><b>My Restaurant</b> is wonderful, come in and
have a seat!</div>]]></src>
        </Label>
      </assets>
    </ScreenAnchor>
  </anchors>
</Feature>
```

7.5 interface VisualAsset

Requirement
http://www.opengis.net/spec/arml/2.0/req/model/VisualAsset/interface
VisualAsset only serves as an interface for other objects defined in the encoding and may not be used directly in the encoding.

XSD :

```
<xsd:complexType name="VisualAssetType" abstract="true">
  <xsd:complexContent>
    <xsd:extension base="ARElementType">
      <xsd:sequence>
        <xsd:element name="enabled" type="xsd:boolean" maxOccurs="1"
minOccurs="0" />
        <xsd:element name="zOrder" type="xsd:int" maxOccurs="1"
minOccurs="0" />
        <xsd:element name="conditions" maxOccurs="1" minOccurs="0">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element ref="Condition" maxOccurs="unbounded"
minOccurs="1" />
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="Orientation" type="OrientationType"
maxOccurs="1" minOccurs="0" />
        <xsd:element name="ScalingMode" type="ScalingModeType"
maxOccurs="1" minOccurs="0" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="VisualAsset" type="VisualAssetType" abstract="true"
substitutionGroup="ARElement" />
```

Example:

```
<VisualAsset id="myVisualAsset">
  <enabled>true</enabled>
  <zOrder>0</zOrder>
  <Orientation>
    <roll>90</roll>
    <tilt>90</tilt>
    <heading>90</heading>
  </Orientation>
  <Conditions>
    ...
  </Conditions>
</VisualAsset>
```

7.5.1 VisualAsset Types

7.5.1.1 interface VisualAsset2D

Requirement
http://www.opengis.net/spec/arml/2.0/req/model/VisualAsset2D/interface
VisualAsset2D only serves as an interface for other objects defined in the encoding and may not be used directly in the encoding.

XSD:

```
<xsd:complexType name="VisualAsset2DType" abstract="true">
  <xsd:complexContent>
    <xsd:extension base="VisualAssetType">
      <xsd:sequence>
        <xsd:element name="width" type="xsd:string" maxOccurs="1"
minOccurs="0" />
        <xsd:element name="height" type="xsd:string" maxOccurs="1"
minOccurs="0" />
        <xsd:element name="orientationMode" maxOccurs="1" minOccurs="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="user" />
              <xsd:enumeration value="absolute" />
              <xsd:enumeration value="auto" />
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="backside" type="xsd:string" maxOccurs="1"
minOccurs="0" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="VisualAsset2D" type="VisualAsset2DType" abstract="true"
substitutionGroup="VisualAsset" />
```

7.5.1.1.1 class Label

XSD:

```
<xsd:complexType name="LabelType">
  <xsd:complexContent>
    <xsd:extension base="VisualAsset2DType">
      <xsd:sequence>
        <xsd:element name="href" maxOccurs="1" minOccurs="0">
          <xsd:complexType>
            <xsd:attribute ref="xlink:href" use="required" />
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="src" type="xsd:anyType" maxOccurs="1"
minOccurs="0" />
        <xsd:element name="hyperlinkBehavior" maxOccurs="1" minOccurs="0">
          <xsd:simpleType>
```

```
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="block"></xsd:enumeration>
              <xsd:enumeration value="blank"></xsd:enumeration>
              <xsd:enumeration value="self"></xsd:enumeration>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="viewportWidth" type="xsd:positiveInteger"
maxOccurs="1" minOccurs="0" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="Label" type="LabelType"
substitutionGroup="VisualAsset2D" />
```

Example:

```
<Label id="mySrcLabel">
  <src>
    <div>Here's my Label in a div</div>
  </src>
</Label>

<Label id="myHrefLabel">
  <href xlink:href="http://www.myserver.com/myLabel.html" />
</Label>

<!-- Example of replacing name and metadata fields -->
<Feature id="empireStateBuilding">
  <name>The Empire State Building</name>
  <metadata>
    <constructed>1929-1931</constructed>
    <height>381m</height>
  </metadata>
  ...
</Feature>
<!-- The Label could be attached to multiple buildings conforming with the
same metadata-layout -->
<Label id="myBuildingLabel">
  <src>
    $[name]<br/>Constructed: $[/constructed]<br/>height: $[/height]
  </src>
</Label>
```

7.5.1.1.2 class Fill

XSD:

```
<xsd:complexType name="FillType">
  <xsd:complexContent>
    <xsd:extension base="VisualAsset2DType">
```

```
      <xsd:sequence>
        <xsd:element name="style" type="xsd:string" maxOccurs="1"
minOccurs="0" />
        <xsd:element name="class" type="xsd:string" maxOccurs="1"
minOccurs="0" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="Fill" type="FillType" substitutionGroup="VisualAsset2D"
/>
```

Example:

```
<Fill id="myFill" style="color:#FF0000;" />

<!-- the same can be achieved with -->
<!-- style-section in arml document -->
<style type="text/css">
  Fill.redFill {
    color : #FF0000;
  }
</style>

<!-- ARElements section of arml document -->
<Fill id="myFill" class="redFill" />
```

7.5.1.1.3 class Text

XSD:

```
<xsd:complexType name="TextType">
  <xsd:complexContent>
    <xsd:extension base="VisualAsset2DType">
      <xsd:sequence>
        <xsd:element name="src" type="xsd:string" maxOccurs="1"
minOccurs="1" />
        <xsd:element name="style" type="xsd:string" maxOccurs="1"
minOccurs="0" />
        <xsd:element name="class" type="xsd:string" maxOccurs="1"
minOccurs="0" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="Text" type="TextType" substitutionGroup="VisualAsset2D"
/>
```

Example:

```
<Text id="myText" style="font-color:#FF0000;">
  <src>This text will be displayed</src>
</Text>
```

7.5.1.1.4 class Image

XSD:

```
<xsd:complexType name="ImageType">
  <xsd:complexContent>
    <xsd:extension base="VisualAsset2DType">
      <xsd:sequence>
        <xsd:element name="href" maxOccurs="1" minOccurs="1">
          <xsd:complexType>
            <xsd:attribute ref="xlink:href" use="required" />
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="Image" type="ImageType"
substitutionGroup="VisualAsset2D" />
```

Example:

```
<Image id="myImage">
  <href xlink:href="http://www.myserver.com/myImage.png" />
</Image>
```

7.5.1.2 class Model

XSD:

```
<xsd:complexType name="ModelType">
  <xsd:complexContent>
    <xsd:extension base="VisualAssetType">
      <xsd:sequence>
        <xsd:element name="href" maxOccurs="1" minOccurs="1">
          <xsd:complexType>
            <xsd:attribute ref="xlink:href" use="required" />
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="type" maxOccurs="1" minOccurs="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="normal" />
              <xsd:enumeration value="infrastructure" />
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="Scale" type="ScaleType" maxOccurs="1"
minOccurs="0" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

```
<xsd:element name="Model" type="ModelType" substitutionGroup="VisualAsset"
/>
```

Example:

```
<Model id="myModel">
   <href xlink:href="http://domain.com/myColladaFile.zip" /> <!-- a URI to
a zip file, containing the COLLADA dae file, textures and any other
ressources required -->
   <type>infrastructure</type> <!-- one of normal|infrastructure -->
   <Orientation>
      <roll>0</roll>
      <tilt>0</tilt>
      <heading>0</heading> <!-- Model is oriented towards north -->
   </Orientation>
   <Scale>
      <x>1</x>
      <y>1</y>
      <z>1</z>
   </Scale>
   <zOrder>0</zOrder> <!-- int value controlling the rendering order
(defaults to 0)-->
</Model>
```

7.5.1.2.1 class Scale

XSD:

```
<xsd:complexType name="ScaleType">
  <xsd:sequence>
    <xsd:element name="x" type="xsd:double" maxOccurs="1" minOccurs="0" />
    <xsd:element name="y" type="xsd:double" maxOccurs="1" minOccurs="0" />
    <xsd:element name="z" type="xsd:double" maxOccurs="1" minOccurs="0" />
  </xsd:sequence>
</xsd:complexType>
```

7.5.2 class Orientation

XSD:

```
<xsd:complexType name="OrientationType">
  <xsd:sequence>
    <xsd:element name="roll" type="xsd:double" maxOccurs="1" minOccurs="0"
/>
    <xsd:element name="tilt" type="xsd:double" maxOccurs="1" minOccurs="0"
/>
    <xsd:element name="heading" type="xsd:double" maxOccurs="1"
minOccurs="0" />
  </xsd:sequence>
</xsd:complexType>
```

7.5.3 class ScalingMode

XSD:

```
<xsd:complexType name="ScalingModeType">
  <xsd:complexContent>
    <xsd:extension base="ARElementType">
      <xsd:sequence>
        <xsd:element name="minScalingDistance" type="xsd:double"
maxOccurs="1" minOccurs="0" />
        <xsd:element name="maxScalingDistance" type="xsd:double"
maxOccurs="1" minOccurs="0" />
        <xsd:element name="scalingFactor" type="xsd:double" maxOccurs="1"
minOccurs="0" />
      </xsd:sequence>
      <xsd:attribute name="type" use="required">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="natural" />
            <xsd:enumeration value="custom" />
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

Example:

```
<VisualAsset id="myVisualAsset">
  ... <!-- visual asset definition -->
  <ScalingMode type="custom">
    <minScalingDistance>50</minScalingDistance>
    <maxScalingDistance>5000</maxScalingDistance>
    <scalingFactor>0.75</scalingFactor>
  </ScalingMode>
</VisualAsset>

<VisualAsset id="myVisualAsset2">
  ... <!-- visual asset definition -->
  <ScalingMode type="natural" /> <!-- this is the default behavior -->
</VisualAsset>
```

7.5.4 interface Condition

Requirement
http://www.opengis.net/spec/arml/2.0/req/model/Condition/interface
Condition only serves as an interface for other objects defined in the encoding and may not be used directly in the encoding.

XSD:

```
<xsd:complexType name="ConditionType" abstract="true">
  <xsd:complexContent>
    <xsd:extension base="ARElementType" />
```

```
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="Condition" type="ConditionType" abstract="true"
substitutionGroup="ARElement" />
```

7.5.4.1 class DistanceCondition

XSD:

```
<xsd:complexType name="DistanceConditionType">
  <xsd:complexContent>
    <xsd:extension base="ConditionType">
      <xsd:sequence>
        <xsd:element name="max" type="xsd:double" maxOccurs="1"
minOccurs="0" />
        <xsd:element name="min" type="xsd:double" maxOccurs="1"
minOccurs="0" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="DistanceCondition" type="DistanceConditionType"
substitutionGroup="Condition" />
```

Example:

```
<Model id="myModel">
  ... <!-- representation of the mountain hut as a 3D model
  <conditions>
    <DistanceCondition>
      <min>200</min> <!-- only visible when distance is more than 200
meters -->
    </DistanceCondition>
  </conditions>
</Model>

<Label id="myLabel">
  ... <!-- representation of the mountain hut as a Label
  <conditions>
    <DistanceCondition>
      <max>500</max>
      <min>200</min> <!-- only visible when distance more than 200 meters,
but less than 500 meters -->
    </DistanceCondition>
  </conditions>
</Label>
```

7.5.4.2 class SelectedCondition

XSD:

```
<xsd:complexType name="SelectedConditionType">
  <xsd:complexContent>
    <xsd:extension base="ConditionType">
```

```
      <xsd:sequence>
        <xsd:element name="listener" maxOccurs="1" minOccurs="0">
          <xsd:simpleType>
            <xsd:restriction base="xsd:string">
              <xsd:enumeration value="feature" />
              <xsd:enumeration value="anchor" />
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:element>
        <xsd:element name="selected" type="xsd:boolean" maxOccurs="1"
minOccurs="1" />
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="SelectedCondition" type="SelectedConditionType"
substitutionGroup="Condition" />
```

Example:

```
<Model id="myModel">
  <conditions>
    <SelectedCondition>
      <listener>feature</listener>
      <selected>true</selected> <!-- only visible when the Feature the
VisualAsset is attached to is selected -->
    <SelectedCondition>
  </conditions>
  <href xlink:href="http://myserver.com/myModel.dae" />
</Model>
```

8 ECMAScript Bindings (normative)

Requirements Class	
http://www.opengis.net/spec/arml/2.0/req/scripting	
Target Type	Software Implementation

The following section describes the scripting part of ARML 2.0. Throughout this section, an WebIDL translation of the object model is introduced to define ECMAScript bindings and a JSON serialization of the ARML 2.0 object model.

ARML provides ECMAScript (the standardized version of JavaScript) bindings to allow the dynamic access and modification of objects in the AR scene, as well as event handlers to react on user input. In addition to the XML serialization, each class defined in ARML also has a JSON serialization, which is used to access and modify the properties of the objects in the scene.

Implementations are encouraged to support ARML's ECMAScript bindings to allow the developer dynamic access to the scene. However, if ECMAScript bindings cannot be provided for whatever reason, the implementation must clearly state that only the descriptive ARML specification is supported.

8.1 Accessing ARElements and Modifying the Scene

Implementations must ensure that an *arml* object is injected into the ECMAScript runtime context on startup. This object is the root node for any scripting operations on the AR scene and serves as the namespace for the objects defined in ARML 2.0.

Requirement
http://www.opengis.net/spec/arml/2.0/req/scripting/general/arml_injection
On startup, the *arml* object shall be injected into the ECMAScript runtime.

In addition to serving as the namespace, *arml* has the following properties and methods:

```
module arml {
  readonly attribute ARElement[] arElements;

  ARElement getARElementById(String id);
  void addToScene(ARElement element);
  void removeFromScene(ARElement element);

  void addEventListener(String type, EventListener listener);
  void removeEventListener(String type, EventListener listener);

  ... all interface objects from below
}
```

getARElementById(String id)
returns the object having its *id* property set to the passed String. In case no such object exists, or *id* is empty, the call returns *null*.

addToScene(ARElement element)
adds the given element to the AR scene

removeFromScene(ARElement element)
removes the given element from the AR scene

8.2 Object Creation and Property Access

Each concrete subclass of *ARElement* has its own constructor. To make an object accessible in the scene, *arml.addToScene(element)* must be invoked first, only then is the element accessible via *arml.getARElementById(element.id)*.

Requirement
http://www.opengis.net/spec/arml/2.0/req/scripting/general/object_access
After an *ARElement* was added to the scene (either through descriptive definition or by calling *arml.addToScene*), its ECMAScript representation is accessible through *arml.getARElementById* with the *id* of the *ARElement* as parameter.

An implementation must ensure that properties set in the descriptive spec are always in sync with the matching properties in the scripting spec. For example, if the following feature is defined in the declarative spec:

```
<Feature id="empireStateBuilding">
  <name>The Empire State Building</name>
  <enabled>true</enabled>
  <anchors>
    ...
  </anchors/>
</Tracker>
```

Then the implementation shall ensure that the following object is accessible:

```
var empireState = arml.getARElementById("empireStateBuilding");
```

Furthermore, the implementation shall ensure that the object stored in *empireState* has its properties set to the following values:

```
empireState = {
  "id" : "empireStateBuilding",
  "name" : "The Empire State Building",
  "enabled" : true,
  anchors : [
    ... //the array of Anchors defined for the Feature
  ]
}
```

The properties of empireState can now be accessed and modified using *empireState.name* etc.

Requirement
http://www.opengis.net/spec/arml/2.0/req/scripting/general/synchronization
An implementation shall ensure that properties set in the descriptive spec are always in sync with the matching properties in the scripting spec.

8.3 Object and Constructor Definitions

The ECMAScript bindings of the objects specified in ARML follow some simple principles.

1. Only concrete classes of ARML can be constructed in a valid way.
2. Constructor parameters consist of all mandatory attributes of the class, plus an optional dictionary (key/value JSON object) parameter allowing the population of all optional parameters.
3. Read-only parameters can only be populated at construction time of the object and must not be altered later.

Any misuse of constructors, methods or properties (e.g. wrong number of parameters or illegal values) provided must result in an Exception.

Requirement
http://www.opengis.net/spec/arml/2.0/req/scripting/general/misuse
Any misuse of constructors, methods and properties shall result in an Exception

Remark: All objects defined below are accessible through the *arml* namespace and, in WebIDL terms, belong to the *arml* module. For example, a new Feature can be created with *new arml.Feature();*

```
interface ARElement {
  readonly attribute string id;
};

dictionary ARElementDict {
  string id;
};
```

8.3.1 Feature

```
[Constructor(optional FeatureDict initDict)]
interface Feature : ARElement {
  attribute string name;
  attribute string description;
  attribute boolean enabled;
  attribute object metadata;
  attribute Anchor[] anchors;
};

dictionary FeatureDict : ARElementDict {
  string name;
  string description;
  boolean enabled;
  object metadata;
  Anchor[] anchors;
};
```

8.3.2 Anchor

```
interface Anchor : ARElement {
  attribute boolean enabled;
};
```

```
dictionary AnchorDict : ARElementDict {
  boolean enabled;
};
```

8.3.3 ARAnchor

```
interface ARAnchor : Anchor {
  attribute VisualAsset[] assets;

  void addEventListener(string type, EventListener listener);
  void removeEventListener(string type, EventListener listener);
};

dictionary ARAnchorDict : AnchorDict {
  VisualAsset[] assets;
};
```

8.3.4 ScreenAnchor

```
[Constructor(Label[] assets, optional ScreenAnchorDict initDict)]
interface ScreenAnchor : Anchor {
  attribute string class;
  attribute ScreenAnchorStyleDict style;
  attribute Label[] assets;
};

dictionary ScreenAnchorDict : AnchorDict {
  string class;
  ScreenAnchorStyleDict style;
};

dictionary ScreenAnchorStyleDict {
  string top;
  string bottom;
  string left;
  string right;
  string width;
  string height;
};
```

8.3.5 Geometry

```
interface Geometry : ARAnchor {
  readonly attribute GMLGeometry gmlGeometry;
};

dictionary GeometryDict : ARAnchorDict {
  GMLGeometry gmlGeometry;
};
```

8.3.6 GMLGeometry

```
interface GMLGeometry {
  readonly attribute string id;
};
```

8.3.7 Point

```
[Constructor(string id, double[] pos, optional PointDict initDict)]
interface Point : GMLGeometry {
  attribute double[] pos;
  readonly attribute string srsName;
  readonly attribute string srsDimension;
};

dictionary PointDict {
  string srsName;
  string srsDimension;
};
```

8.3.8 LineString

```
[Constructor(string id, Point[] posList)]
interface LineString : GMLGeometry {
  readonly attribute string id;
  attribute Point[] posList;
};
```

Remark: The descriptive specification allows setting the srsName and srsDimension for the entire LineString, as well as single Points separately. The scripting specification only supports setting the srsName for each single Point. In case the srsName and srsDimension should be set for the entire LineString, the implementation needs to make sure it runs through the entire list of Points and sets the srsName and srsDimension accordingly.

8.3.9 Polygon

```
[Constructor(string id, LineString exterior, optional PolygonDict
initDict)]
interface Polygon : GMLGeometry {
  readonly attribute string id;
  attribute LineString[] interior;
  attribute LineString exterior;
};

dictionary PolygonDict {
  LineString[] interior;
};
```

Remark 1: As *LinearRing*s are closed *LineString*s from a technical perspective, ARML's ECMAScript bindings avoid an additional *LinearRing* type and use *LineString* instead.

Remark 2: The descriptive specification allows setting the srsName and srsDimension for the entire Polygon, as well as single LinearRings and Points separately. The scripting specification only supports setting the srsName for each single Point. In case the srsName and srsDimension should be set for the entire Polygon or LinearRing, the implementation needs to make sure it runs through the entire list of Points and sets the srsName and srsDimension accordingly.

8.3.10 RelativeTo

```
[Constructor(object ref, GMLGeometry gmlGeometry)]
interface RelativeTo : ARAnchor {
```

```
  readonly attribute object ref;
  attribute GMLGeometry gmlGeometry;
};
```

ref can either be an object or a String with its value set to *"#user"*, thus the type has to be a general object.

8.3.11 Tracker

```
[Constructor(string uri, optional TrackerDict initDict)]
interface Tracker : ARElement {
  readonly attribute string uri;
  attribute string src;
};

dictionary TrackerDict : ARElementDict {
  string src;
};
```

8.3.12 Trackable

```
[Constructor(TrackableConfig[] configs, optional TrackableDict initDict)]
interface Trackable : ARAnchor {
  readonly attribute TrackableConfig[] configs;
  attribute double size;

  void addEventListener(string type, EventListener listener);
  void removeEventListener(string type, EventListener listener);
};

dictionary TrackableDict : ARAnchorDict {
  double size;
};


[Constructor(Tracker tracker, string src, optional int order)]
interface TrackableConfig {
  readonly attribute Tracker tracker;
  readonly attribute string src;
  readonly attribute int order;
};
```

8.3.13 VisualAsset

```
interface VisualAsset : ARElement {
  attribute boolean enabled;
  attribute int zOrder;
  attribute Condition[] conditions;
  attribute Orientation orientation
  attribute ScalingMode scalingMode;

  void addEventListener(string type, EventListener listener);
  void removeEventListener(string type, EventListener listener);
};

dictionary VisualAssetDict : ARElementDict {
  boolean enabled;
```

```
  int zOrder;
  Condition[] conditions;
  Orientation orientation;
  ScalingMode ScalingMode;
};
```

8.3.14 Orientation

```
[Constructor(OrientationDict initDict)]
interface Orientation {
  attribute double roll;
  attribute double tilt;
  attribute double heading;
}

dictionary OrientationDict {
  double roll;
  double tilt;
  double heading;
};
```

8.3.15 ScalingMode

```
[Constructor(string type, optional ScalingModeDict initDict)]
interface ScalingMode {
  readonly attribute string type;
  attribute double minScalingDistance;
  attribute double maxScalingDistance;
  attribute double scalingFactor;
};

dictionary ScalingModeDict {
  double minScalingDistance;
  double maxScalingDistance;
  double scalingFactor;
};
```

8.3.16 VisualAsset2D

```
interface VisualAsset2D : VisualAsset {
  attribute string width;
  attribute string height;
  attribute string orientationMode;
  attribute string backside;
};

dictionary VisualAsset2DDict : VisualAssetDict {
  string width;
  string height;
  string orientationMode;
  string backside;
};
```

8.3.17 Label

ARML's *arml* root object is injected into each Label before it is constructed. From the *arml* object a Label can access any object in the scene.

Requirement
http://www.opengis.net/spec/arml/2.0/req/scripting/core/Label/injection
In case ARML 2.0's ECMAScript bindings are supported, ARML's *arml* root object shall be injected into each Label before it is constructed

```
[Constructor(LabelDict initDict)]
interface Label : VisualAsset2D {
  attribute string href;
  attribute string src;
  attribute string hyperlinkBehavior;
  attribute int viewportWidth;
};

dictionary LabelDict : VisualAsset2DDict {
  string href;
  string src;
  string hyperlinkBehavior;
  int viewportWidth;
};
```

8.3.18 Fill

```
[Constructor(FillDict initDict)]
interface Fill : VisualAsset2D {
  attribute FillStyleDict style;
  attribute string class;
};

dictionary FillDict : VisualAsset2DDict {
  FillStyleDict style;
  string class;
};

dictionary FillStyleDict {
  string color;
};
```

8.3.19 Text

```
[Constructor(string src, TextDict initDict)]
interface Text : VisualAsset2D {
  attribute string src;
  attribute TextStyleDict style;
  attribute string class;
};

dictionary TextDict : VisualAsset2DDict {
  TextStyleDict style;
  string class;
};

dictionary TextStyleDict {
  string fontColor;
  string backgroundColor;
};
```

8.3.20 Image

```
[Constructor(string href)]
interface Image : VisualAsset2D {
  attribute string href;
};
```

8.3.21 Model

```
[Constructor(string href, ModelDict initDict)]
interface Model : VisualAsset {
  attribute string href;
  attribute string type;
  attribute Scale scale;

  string start3DAnimation(string id, int loopCount, EventListener
callback);
  void stop3DAnimation(string animationId);
  void pause3DAnimation(string animationId);
  void resume3DAnimation(string animationId);
};

dictionary ModelDict : VisualAssetDict {
  string href;
  string type;
  Scale scale;
};
```

start3DAnimation starts an animation that was declared in the Model's file.

Parameters:

id: The animation to start is referenced by an id with which the animation can be identified in the Model file. In case the animations in the Model file are not referenceable with IDs, the position of the Animation in the file (starting with 1) can be used as a reference. In case no such animation exists, an Exception must be thrown.

Requirement
http://www.opengis.net/spec/arml/2.0/req/scripting/Model/animationId
In case the animationId cannot be resolved, an Exception shall be thrown.

loopCount: An optional parameter specifying how often the animation should loop. If *loopCount* is set to -1, the animation will loop infinitively often. If not set, the value defaults to 1.

Requirement
http://www.opengis.net/spec/arml/2.0/req/scripting/Model/animationLoopCount
If loopCount is not set, it defaults to 1. If it is set to -1, the animation will loop infinitely often.

callback: An optional callback function can be supplied which will be executed right after the animation finished with all the loops provided. The callback will not be executed when the animation was manually stopped (see stop3DAnimation). For more details on EventListeners, see *Event Handling*.

Requirement
http://www.opengis.net/spec/arml/2.0/req/scripting/Model/animationCallback
If set, the callback will be triggered when the animation finished playing all the defined loops. It shall not be triggered when the animation was stopped manually.

Returns:
a string identifying the 3DAnimation. This String can be used to stop the Animation.

stop3DAnimation stops an animation before it regularly finishes.
Parameters:
animationId: The id returned when the animation was started
Returns:
void

pause3DAnimation pauses a currently running animation. Has no effect if the Animation is not running.
Parameters:
animationId: The id returned when the animation was started
Returns:
void

resume3DAnimation resumes a currently paused animation. Has no effect if the Animation is not paused.
Parameters:
animationId: The id returned when the animation was started
Returns:
void

8.3.22 Scale

```
[Constructor(ScaleDict initDict)]
interface Scale {
  attribute double x;
  attribute double y;
  attribute double z;
};

dictionary ScaleDict {
  double x;
  double y;
  double z;
};
```

8.3.23 DistanceCondition

```
[Constructor(DistanceConditionDict initDict)]
interface DistanceCondition : ARElement {
  attribute double max;
  attribute double min;
};

dictionary DistanceConditionDict : ARElementDict {
  double max;
```

```
  double min;
};
```

8.3.24 SelectedCondition

```
[Constructor(boolean selected, SelectedConditionDict initDict)]
interface SelectedCondition : ARElement {
  attribute string listener;
  attribute boolean selected;
};

dictionary SelectedConditionDict : ARElementDict {
  string listener;
  boolean selected;
};
```

8.3.25 Animation

```
interface Animation {
  void addEventListener(string type, EventListener listener);
  void removeEventListener(string type, EventListener listener);

  void start(int loopCount, int delay);
  void stop();
  boolean isRunning();
};
```

Animations cannot be defined in the declarative part of ARML; they can only be declared and controlled in the scripting part. Animations constantly modify the value of a property over a certain time period.
2 different types of Animations are supported in the ECMAScript bindings of ARML: NumberAnimations and GroupAnimations. They all inherit from Animation.

start starts an animation.
Parameters:
loopCount: An optional parameter specifying how often the animation should loop. If *loopCount* is set to -1, the animation will loop infinitively often. If not set, the value defaults to 1.

Requirement
http://www.opengis.net/spec/arml/2.0/req/scripting/Animation/loopCount
If loopCount is not set, it defaults to 1. If it is set to -1, the animation will loop infinitely often.

delay: The number of milliseconds the start of the animation will be delayed. If not set, the value defaults to 0 (immediate start).

Requirement
http://www.opengis.net/spec/arml/2.0/req/scripting/Animation/delay
If delay is not set, it defaults to 0, causing the animation to start immediately.

Returns:
void

stop stops an animation before it regularly finished.
Parameters:
-
Returns:
void

isRunning returns if an animation is currently running.
Parameters:
-
Returns:
true if the Animation is currently running, false otherwise.

8.3.26 NumberAnimation

```
[Constructor(ARElement target, string property, float start, float end,
float duration)]
interface NumberAnimation : Animation {
  readonly attribute ARElement target;
  readonly attribute string property;
  readonly attribute float start;
  readonly attribute float end;
  readonly attribute int duration;
};
```

NumberAnimations constantly modify a numeric value over a certain period of time from a given start value to a specified end value. Between start and end, the value is linearly interpolated.

Properties:
target specifies the ARElement that holds the property that will be animated. Must not be null.
property holds the name of the property that will be animated. The property must hold a numeric value.
start holds the start value of the Animation. If null, the current value of the property is used as start value.

Requirement
http://www.opengis.net/spec/arml/2.0/req/scripting/Animation/Number/start
If no explicit *start* value is set, the animation's start value is equal to the property's current value.

end holds the end value of the Animation. The property will take on this value after the Animation completed.

Requirement
http://www.opengis.net/spec/arml/2.0/req/scripting/Animation/Number/end
After the Animation completed successfully, the animated property takes on the value as specified in the *end* property.

duration, supplied in milliseconds, specifies the duration of one loop of the Animation.

8.3.27 GroupAnimation

```
[Constructor(string type, Animation[] animations)]

interface GroupAniimation : Animation {
  readonly attribute string type;
  readonly attribute Animation[] animations;
};
```

A GroupAnimation groups multiple Animations and runs them depending on the type of the GroupAnimation. Type can either be parallel, causing all Animations in the GroupAnimation to start at the same time, or sequential, causing the Animations to run one after another.

Properties:
type specifies the type of the GroupAnimation, either *parallel* or *sequential*.
animations holds the array of Animations contained in the GroupAnimation.

A parallel GroupAnimation loop has finished when the longest Animation in the group has finished. A sequential GroupAnimation loop has finished when the last Animation in the group has finished.

Requirement
http://www.opengis.net/spec/arml/2.0/req/scripting/Animation/Group/endDefinition
A parallel GroupAnimation loop has finished when the longest Animation in the group has finished. A sequential GroupAnimation loop has finished when the last Animation in the group has finished.

8.3.28 Event Handling

Event handling in ARML is based on concepts of event handling in HTML; see http://www.w3.org/TR/2000/REC-DOM-Level-2-Events-20001113/events.html for details.

Developers can react on certain events by registering *EventListener*s listening on the occurrence of a certain *Event type* on specific *event targets*.

The following ARML classes serve as event targets, with their corresponding Events.

EventTarget	Event Type	Description
arml	locationChanged	Fires when the implementation receives a new geospatial location representing the user's current position
VisualAsset	enterFieldOfVision	Fires when at least one pixel of the VisualAsset becomes visible on the screen
	exitFieldOfVision	Fires when the last pixel of the VisualAsset moves out of the screen
	click	Fires when the VisualAsset was clicked
ARAnchor	enterFieldOfVision	Fires when at least a part of the area the ARAnchor covers becomes visible on the screen

EventTarget	Event Type	Description
	exitFieldOfVision	Fires when the ARAnchor becomes invisible on the screen
Trackable	tracked	Fires when the Trackable was detected in the scene
	trackingLost	Fires when the Trackable cannot be tracked anymore
Animation	start	Fires just before the animation starts
	finish	Fires just after the animation finished

Event Listeners are registered in the event targets using

```
eventTarget.addEventListener(string type, EventListener listener);
```

Event Listeners are removed using

```
eventTarget.removeEventListener(string type, EventListener listener);
```

8.3.28.1 EventListener

```
interface EventListener {
  void handleEvent(Event event);
};
```

handleEvent is called whenever an event occurs of the type for which the EventListener interface was registered. The *event* parameter holds the Event object containing contextual information about the event.

Requirement
http://www.opengis.net/spec/arml/2.0/req/scripting/EventHandling/firing
An EventHandler is triggered when an event occurs of the type for which the EventListener interface was registered.

8.3.28.2 Event

```
interface Event {
  readonly attribute EventTarget target;
};
```

target is used to indicate the Event Target to which the event was originally dispatched.

Example:

```
var clickFunction = function(event){
  var t = event.eventTarget.src;
  //do something
};
```

```
var text = new arml.Text("This is my text");
text.addEventListener("click", clickFunction);
```

Annex A: Conformance Classes

A.1: Conformance Test Class Encoding	
http://www.opengis.net/spec/arml/2.0/conf/model	
Target Type	Encoding
Tests described in this section shall be used to test conformance of an XML encoding with the ARML 2.0 XML serialization.	

A.1.1 Encoding can be validated against XSD	
http://www.opengis.net/spec/arml/2.0/conf/model/general/xsd_verification	
Title	Encoding can be validated against XSD
Abbreviation	xsd_verification
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/model/general/xsd_verification
Test Purpose	To validate whether a given XML encoding complies with the ARML XSD definition.
Test Method	Validate the XML encoding against the XSD and verify that the validation is successful.

A.1.2 Encoding has correct root element	
http://www.opengis.net/spec/arml/2.0/conf/general/root_element	
Title	Encoding has correct root element
Abbreviation	root_element
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/model/general/root_element
Test Purpose	To validate that a given XML encoding has the correct root element set
Test Method	Inspect the encoding and check that the root element is arml.

A.1.3 Only Subclasses of ARElement allowed in ARElements	
http://www.opengis.net/spec/arml/2.0/conf/model/ARElement/container	
Title	Only Subclasses of ARElement allowed in ARElements
Abbreviation	ARElement/container
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/model/ARElement/container
Test Purpose	To validate that a given XML encoding complies with the inheritance structure defined in the ARML 2.0 object model.
Test Method	Inspect the encoding and check that only subclasses of ARElement are immediate children in the <ARElements> tag.

A.1.4 ARElement not allowed in encoding	
http://www.opengis.net/spec/arml/2.0/conf/model/ARElement/interface	
Title	ARElement not allowed in encoding
Abbreviation	ARElement/interface
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/model/ARElement/interface
Test Purpose	To validate that a given XML encoding does not use the abstract ARElement element.
Test Method	Inspect the encoding and check that ARElement is never used in the encoding.

A.1.5 IDs are unique	
http://www.opengis.net/spec/arml/2.0/conf/model/ARElement/id	
Title	IDs are unique
Abbreviation	ARElement/id
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/model/ARElement/id
Test Purpose	To validate that each element with an ID can be uniquely referenced
Test Method	Inspect the encoding and check that ids of ARElements are unique.

A.1.6 Invalid Anchor references	
http://www.opengis.net/spec/arml/2.0/conf/model/Feature/anchors/relative	
Title	Invalid Anchor references
Abbreviation	Feature/anchors/relative
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/model/Feature/anchors/relative
Test Purpose	To avoid having invalid Anchor references in the XML encoding
Test Method	Inspect the encoding and check that Anchors attached to Features by reference are referenced correctly using the xlink:href attribute.

A.1.7 Anchor not allowed in encoding	
http://www.opengis.net/spec/arml/2.0/conf/model/Anchor/interface	
Title	Anchor not allowed in encoding
Abbreviation	Anchor/interface
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/model/Anchor/interface
Test Purpose	To validate that a given XML encoding does not use the abstract Anchor element.
Test Method	Inspect the encoding and check that Anchor is never used in the encoding.

A.1.8 ARAnchor not allowed in encoding	
http://www.opengis.net/spec/arml/2.0/conf/model/ARAnchor/interface	
Title	ARAnchor not allowed in encoding
Abbreviation	ARAnchor/interface
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/model/ARAnchor/interface
Test Purpose	To validate that a given XML encoding does not use the abstract Anchor element.
Test Method	Inspect the encoding and check that ARAnchor is never used in the encoding.

A.1.9 Invalid VisualAsset references	
http://www.opengis.net/spec/arml/2.0/conf/model/ARAnchor/anchors/relative	
Title	Invalid VisualAsset references
Abbreviation	ARAnchor/anchors/relative
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/model/ARAnchor/anchors/relative
Test Purpose	To avoid having invalid VisualAsset references in the XML encoding
Test Method	Inspect the encoding and check that VisualAssets attached to Anchors by reference are referenced correctly using the xlink:href attribute.

A.1.10 GMLGeometries not allowed in encoding	
http://www.opengis.net/spec/arml/2.0/conf/model/GMLGeometries/interface	
Title	GMLGeometries not allowed in encoding
Abbreviation	GMLGeometries/interface
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/model/GMLGeometries/interface
Test Purpose	To validate that a given XML encoding does not use the abstract GMLGeometries element.
Test Method	Inspect the encoding and check that GMLGeometries is never used in the encoding.

A.1.11 GML Point validation	
http://www.opengis.net/spec/arml/2.0/conf/model/Point/xsd	
Title	GML Point validation
Abbreviation	Point/xsd
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/model/Point/xsd
Test Purpose	To validate that a Point defined in an XML encoding also validates according to the GML Point specification.
Test Method	Inspect the encoding and check that all Point elements validate correctly against the Point Type defined in the XSD in the [GML Specification].

A.1.12 GML LineString validation	
http://www.opengis.net/spec/arml/2.0/conf/model/LineString/xsd	
Title	GML LineString validation
Abbreviation	LineString/xsd
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/model/LineString/xsd
Test Purpose	To validate that a LineString defined in an XML encoding also validates according to the GML LineString specification.
Test Method	Inspect the encoding and check that all LineString elements validate correctly against the LineString Type defined in the XSD in the [GML Specification].

A.1.13 GML Polygon validation	
http://www.opengis.net/spec/arml/2.0/conf/model/Polygon/xsd	
Title	GML Polygon validation
Abbreviation	Polygon/xsd
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/model/Polygon/xsd
Test Purpose	To validate that a Polygon defined in an XML encoding also validates according to the GML Polygon specification.
Test Method	Inspect the encoding and check that all Polygon elements validate correctly against the Polygon Type defined in the XSD in the [GML Specification].

A.1.14 GML LinearRing order	
http://www.opengis.net/spec/arml/2.0/conf/model/GMLGeometries/LinearRing/order	
Title	GML LinearRing order
Abbreviation	GMLGeometries/LinearRing/order
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/model/GMLGeometries/LinearRing/order
Test Purpose	To validate that a LinearRing is has a well-defined up- and down-side.
Test Method	Inspect the encoding and check that a LinearRing's coordinates are specified in counter-clockwise order.

A.1.15 Valid RelativeTo references	
http://www.opengis.net/spec/arml/2.0/conf/model/RelativeTo/ref	
Title	Valid RelativeTo references
Abbreviation	RelativeTo/ref
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/model/RelativeTo/ref
Test Purpose	To validate that a RelativeTo element uses only valid elements as reference.
Test Method	Inspect the encoding and check that any RelativeTo element has its ref property set to either - reference an ARAnchor (except a LineString) by using a URI - reference a Model by using a URI - #user

A.1.16 VisualAsset not allowed in encoding	
http://www.opengis.net/spec/arml/2.0/conf/model/VisualAsset/interface	
Title	VisualAsset not allowed in encoding
Abbreviation	VisualAsset/interface
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/model/VisualAsset/interface
Test Purpose	To validate that a given XML encoding does not use the abstract VisualAsset element.
Test Method	Inspect the encoding and check that VisualAsset is never used in the encoding.

A.1.17 VisualAsset2D not allowed in encoding	
http://www.opengis.net/spec/arml/2.0/conf/model/VisualAsset2D/interface	
Title	VisualAsset2D not allowed in encoding
Abbreviation	VisualAsset2D/interface
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/model/VisualAsset2D/interface
Test Purpose	To validate that a given XML encoding does not use the abstract VisualAsset2D element.
Test Method	Inspect the encoding and check that VisualAsset2D is never used in the encoding.

A.1.18 Condition not allowed in encoding	
http://www.opengis.net/spec/arml/2.0/conf/model/Condition/interface	
Title	Condition not allowed in encoding
Abbreviation	Condition/interface
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/model/Condition/interface
Test Purpose	To validate that a given XML encoding does not use the abstract Condition element.
Test Method	Inspect the encoding and check that Condition is never used in the encoding.

A.2: Conformance Test Class Descriptive Implementation

http://www.opengis.net/spec/arml/2.0/conf/core	
Target Type	Implementation
Tests described in this section shall be used to test conformance of an Implementation capable of handling ARML 2.0 files with only descriptive content.	

A.2.1 Parse ARML files	
http://www.opengis.net/spec/arml/2.0/conf/core/parse_encoding	
Title	Parse ARML files
Abbreviation	parse_encoding
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/parse_encoding
Test Purpose	To validate that an implementation can parse valid ARML 2.0 files
Test Method	Verify that the implementation can parse valid ARML 2.0 encodings that pass all tests in http://www.opengis.net/spec/arml/2.0/conf/model.

A.2.2 Default Unit	
http://www.opengis.net/spec/arml/2.0/conf/core/units	
Title	Default Unit
Abbreviation	units
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/units
Test Purpose	To validate that an implementation's interpretation of sizes is in line with the

	specification's definition of sizes.
Test Method	Verify that any VisualAsset2D object with width set to 1 appears the same size on the screen as a real world object of 1-meter width.

A.2.3 Invalid user ID	
http://www.opengis.net/spec/arml/2.0/conf/core/ARElement/id_user	
Title	Invalid user ID
Abbreviation	ARElement/id_user
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/ARElement/id_user
Test Purpose	To validate that an implementation does not allow ARElements with id *user*.
Test Method	Verify that the implementation ignores ARElement ids that are set to *user*.

A.2.4 Disabling Features	
http://www.opengis.net/spec/arml/2.0/conf/core/Feature/enabled	
Title	Disabling Features
Abbreviation	Feature/enabled
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/Feature/enabled
Test Purpose	To validate that an implementation ignores a Feature and its associated objects when the Feature is disabled in the encoding.
Test Method	Verify that the implementation ignores any Feature and its associated Anchors and VisualAssets when its enabled property is set to false.

A.2.5 Disabling Anchors	
http://www.opengis.net/spec/arml/2.0/conf/core/Anchor/enabled	
Title	Disabling Anchors
Abbreviation	Anchor/enabled
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/Anchor/enabled
Test Purpose	To validate that an implementation ignores an Anchor and its associated objects when the Anchor is disabled in the encoding.
Test Method	Verify that the implementation ignores any Anchor and its associated VisualAssets when its enabled property is set to false.

A.2.6 Anchors without Features	
http://www.opengis.net/spec/arml/2.0/conf/core/Anchor/anchor_without_feature	
Title	Anchors without Features
Abbreviation	Anchor/anchor_without_feature
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/Anchor/anchor_without_feature
Test Purpose	To validate that an implementation will handle Anchors even when they are not attached to a Feature.
Test Method	Verify that the implementation adds Anchors to the composed scene even when they are direct descendants of the ARElements tag.

A.2.7 ARAnchors without Visual Assets	
http://www.opengis.net/spec/arml/2.0/conf/core/ARAnchor/no_visual_asset	
Title	ARAnchors without Visual Assets
Abbreviation	ARAnchor/no_visual_asset
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/ARAnchor/no_visual_asset
Test Purpose	To validate that an implementation handles setting default VisualAssets to Anchors without a specific VisualAsset correctly.
Test Method	Verify that, when no valid VisualAsset is attached to an ARAnchor, the implementation - adds an auto-generated VisualAsset of type Text to the ARAnchor, with its text set to the name of the Feature. - Adds no VisualAsset in case the Anchor is not linked to a Feature, or the Feature's name property is empty.

A.2.8 No user position	
http://www.opengis.net/spec/arml/2.0/conf/core/Geometry/no_position	
Title	No user position
Abbreviation	Geometry/no_position
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/Geometry/no_position
Test Purpose	To validate that the implementation does not fail when the user's current position cannot be determined.
Test Method	Verify that the implementation ignores any Geometry anchor in case the user's current position cannot be determined.

A.2.9 Unknown CRS	
http://www.opengis.net/spec/arml/2.0/conf/core/GMLGeometries/crs	
Title	Unknown CRS
Abbreviation	GMLGeometries/crs
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/GMLGeometries/crs
Test Purpose	To validate that the implementation supports the default coordinate reference system (CRS), and does not fail when a CRS is unknown.
Test Method	Verify that the implementation supports at least the WGS84 coordinate reference system (CRS), and verify that the Implementation ignores any Geometry anchor with an unknown alternative CRS.

A.2.10 Default CRS	
http://www.opengis.net/spec/arml/2.0/conf/core/GMLGeometries/default_crs	
Title	Default CRS
Abbreviation	GMLGeometries/default_crs
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/GMLGeometries/default_crs
Test Purpose	To validate that the implementation assumes the default CRS in case a CRS is not specifically set.
Test Method	Verify that the implementation uses WGS84 for any GMLGeometry that does not

	have the srsName attribute set.

A.2.11 Missing Altitude	
http://www.opengis.net/spec/arml/2.0/conf/core/GMLGeometries/no_altitude	
Title	Missing Altitude
Abbreviation	GMLGeometries/no_altitude
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/GMLGeometries/no_altitude
Test Purpose	To validate that the implementation assumes a default altitude in case an altitude is not specifically set.
Test Method	Verify that the implementation sets the user's current altitude for each GMLGeometry not specifying its own altitude value.

A.2.12 LineString definition	
http://www.opengis.net/spec/arml/2.0/conf/core/GMLGeometries/LineString/definition	
Title	LineString definition
Abbreviation	GMLGeometries/LineString/definition
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/GMLGeometries/LineString/definition
Test Purpose	To validate that the implementation does not fail in case a LineString is not valid.
Test Method	Verify that the implementation ignores any LineString that does not consist of at least 2 coordinate tuples.

A.2.13 LinearRing definition	
http://www.opengis.net/spec/arml/2.0/conf/core/GMLGeometries/LinearRing/definition	
Title	LinearRing definition
Abbreviation	GMLGeometries/LinearRing/definition
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/GMLGeometries/LinearRing/definition
Test Purpose	To validate that the implementation does not fail in case a LinearRing is not valid.
Test Method	Verify that the implementation ignores any LinearRing that does not consist of at least 4 coordinate tuples, or which starting point and end point do not match.

A.2.14 Polygon definition	
http://www.opengis.net/spec/arml/2.0/conf/core/GMLGeometries/Polygon/definition	
Title	Polygon definition
Abbreviation	GMLGeometries/Polygon/definition
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/GMLGeometries/Polygon/definition
Test Purpose	To validate that the implementation does not fail in case a Polygon is not valid.
Test Method	Verify that the implementation ignores any Polygon with an invalid exterior LinearRing, and any inner "hole" defined by an invalid interior LinearRing.

A.2.15 Local CS for GMLGeometries	
http://www.opengis.net/spec/arml/2.0/conf/core/GMLGeometries/local_cs/cs_type	
Title	Local CS for GMLGeometries
Abbreviation	GMLGeometries/local_cs/cs_type
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/GMLGeometries/local_cs/cs_type
Test Purpose	To validate that the implementation defines a correct local coordinate reference system for GMLGeometries.
Test Method	Verify that the Implementation defines a local Cartesian coordinate system of each GMLGeometry, except for LineString, which does not define a coordinate system at all.

A.2.16 Local CS for Points	
http://www.opengis.net/spec/arml/2.0/conf/core/GMLGeometries/local_cs/cs_type/Point	
Title	Local CS for Points
Abbreviation	GMLGeometries/local_cs/cs_type/Point
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/GMLGeometries/local_cs/cs_type/Point
Test Purpose	To validate that the implementation defines a correct local coordinate reference system for a Point.
Test Method	Verify that the Implementation defines the local coordinate system of a Point according to paragraph 6.4.1.2.5 section Point.

A.2.17 Local CS for Polygons	
http://www.opengis.net/spec/arml/2.0/conf/core/GMLGeometries/local_cs/cs_type/Polygon	
Title	Local CS for Polygons
Abbreviation	GMLGeometries/local_cs/cs_type/Polygon
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/GMLGeometries/local_cs/cs_type/Polygon
Test Purpose	To validate that the implementation defines a correct local coordinate reference system for a Polygon.
Test Method	Verify that the Implementation defines the local coordinate system of a Polygon according to paragraph 6.4.1.2.5 section Polygon.

A.2.18 Unknown Trackers	
http://www.opengis.net/spec/arml/2.0/conf/core/Trackable_And_Tracker/unknown_tracker	
Title	Unknown Trackers
Abbreviation	Trackable_And_Tracker/unknown_tracker
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/Trackable_And_Tracker/unknown_tracker
Test Purpose	To validate that the implementation does not fail in case a Tracker is unknown.
Test Method	Verify that the Implementation ignores any unknown Tracker and its associated Trackables.

A.2.19 Contained Trackable	
http://www.opengis.net/spec/arml/2.0/conf/core/Trackable_And_Tracker/contained_trackable	
Title	Contained Trackable
Abbreviation	Trackable_And_Tracker/contained_trackable
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/Trackable_And_Tracker/contained_trackable
Test Purpose	To validate that the implementation can handle Trackables stored in containers.
Test Method	Verify that, when the src property of a Tracker is set, the Implementation considers the value as a link to a container that holds multiple Trackables.

A.2.20 Trackable 2D Size	
http://www.opengis.net/spec/arml/2.0/conf/core/Trackable_And_Tracker/trackable_2D_size	
Title	Trackable 2D Size
Abbreviation	Trackable_And_Tracker/Trackable_2D_size
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/Trackable_And_Tracker/trackable_2D_size
Test Purpose	To validate that an implementation's interpretation of a size of a 2D Trackable is in line with the specification's definition of sizes of 2D Trackables.
Test Method	Verify that the Implementation considers the size of the Trackable set in the encoding as the width of the tracked object in meters when the Trackable is a 2D object.

A.2.21 Trackable 3D Size	
http://www.opengis.net/spec/arml/2.0/conf/core/Trackable_And_Tracker/trackable_3D_size	
Title	Trackable 3D Size
Abbreviation	Trackable_And_Tracker/Trackable_3D_size
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/Trackable_And_Tracker/trackable_3D_size
Test Purpose	To validate that an implementation's interpretation of a size of a 3D Trackable is in line with the specification's definition of sizes of 3D Trackables.
Test Method	Verify that the Implementation considers the size of the Trackable set in the encoding as the size of one unit in the engineering coordinate system of the model, in meters, when the Trackable is a 3D object.

A.2.22 Precedence of Trackable Size	
http://www.opengis.net/spec/arml/2.0/conf/core/Trackable_And_Tracker/trackable_size_preset	
Title	Precedence of Trackable Size
Abbreviation	Trackable_And_Tracker/Trackable_size_preset
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/Trackable_And_Tracker/trackable_size_preset
Test Purpose	To validate that an implementation always considers the encoding's size setting to be of higher precedence than the size setting in the binary for Trackables.
Test Method	Verify that the Implementation gives precedence to the size property in the

	encoding over any size information in the binary representation of a Trackable, in case both are set.

A.2.23 Missing Trackable Size	
http://www.opengis.net/spec/arml/2.0/conf/core/Trackable_And_Tracker/trackable_missing_size	
Title	Missing Trackable Size
Abbreviation	Trackable_And_Tracker/trackable_missing_size
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/Trackable_And_Tracker/trackable_missing_size
Test Purpose	To validate that the implementation does not fail in case it cannot determine the correct size of a Trackable.
Test Method	Verify that the Implementation ignores a Trackable that has no size information set (neither in the binary, not in the encoding).

A.2.24 Default order for TrackingConfig	
http://www.opengis.net/spec/arml/2.0/conf/core/Trackable_And_Tracker/config_order_max	
Title	Default order for TrackingConfig
Abbreviation	Trackable_And_Tracker/config_order_max
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/Trackable_And_Tracker/config_order_max
Test Purpose	To validate that the implementation sets the correct default value for a TrackingConfig order, in case it is not set.
Test Method	Verify that the Implementation sets the order property to the maximum integer value when it is not set for a particular TrackingConfig.

A.2.25 RelativeTo src dimension	
http://www.opengis.net/spec/arml/2.0/conf/core/RelativeTo/GMLGeometry_properties	
Title	RelativeTo src dimension
Abbreviation	RelativeTo/GMLGeometry_properties
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/RelativeTo/GMLGeometry_properties
Test Purpose	To validate that the implementation sets the correct srsDimension property in the element referenced by a RelativeTo anchor.
Test Method	Verify that the Implementation ignores the srsDimension and srsName properties of a GMLGeometry, and sets srsDimension to 3, when the GMLGeometry is used within a RelativeTo anchor.

A.2.26 ScreenAnchor property conflicts	
http://www.opengis.net/spec/arml/2.0/conf/core/ScreenAnchor/property_conflicts	
Title	ScreenAnchor property conflicts
Abbreviation	ScreenAnchor/property_conflicts
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/ScreenAnchor/property_conflicts
Test Purpose	To validate that the implementation fulfills the precedence rules for ScreenAnchors
Test Method	Verify that the implementation adheres to the following precedence-rules when conflicting top/bottom/height or left/right/width values are supplied in ScreenAnchors: - Top takes precedence over height, which takes precedence over bottom - Left takes precedence over width, which takes precedence over right

A.2.27 ScreenAnchor missing properties	
http://www.opengis.net/spec/arml/2.0/conf/core/ScreenAnchor/missing_properties	
Title	ScreenAnchor missing properties
Abbreviation	ScreenAnchor/missing_properties
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/ScreenAnchor/missing_properties
Test Purpose	To validate that the implementation sets the correct default values for ScreenAnchors
Test Method	Verify that the Implementation sets the following default values: - top is set to 0 when neither top, nor bottom is given - left is set to 0 when neither left, nor right is given - width and height are set to 100% when not given

A.2.28 ScreenAnchor ignored properties	
http://www.opengis.net/spec/arml/2.0/conf/core/ScreenAnchor/ignored_properties	
Title	ScreenAnchor ignored properties
Abbreviation	ScreenAnchor/ignored_properties
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/ScreenAnchor/ignored_properties
Test Purpose	To validate that the implementation ignores properties in ScreenAnchors as defined in the specification.
Test Method	Verify that the Implementation ignores the following properties for an Asset when it is attached to a ScreenAnchor: - width and height - Orientation - orientationMode - ScalingMode - Any DistanceConditions

A.2.29 ScreenAnchor default properties	
http://www.opengis.net/spec/arml/2.0/conf/core/ScreenAnchor/default_properties	
Title	ScreenAnchor default properties
Abbreviation	ScreenAnchor/default_properties
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/ScreenAnchor/default_properties
Test Purpose	To validate that a ScreenAnchor's distance to the user is always 0.
Test Method	Verify that the Implementation sets the distance from the user to any ScreenAnchor to 0.

A.2.30 Disabling Visual Assets	
http://www.opengis.net/spec/arml/2.0/conf/core/VisualAsset/enabled	
Title	Disabling Visual Assets
Abbreviation	VisualAsset/enabled
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/VisualAsset/enabled
Test Purpose	To validate that an implementation ignores a VisualAsset it is disabled in the encoding.
Test Method	Verify that the implementation ignores any VisualAsset when its enabled property is set to false.

A.2.31 VisualAsset projection order	
http://www.opengis.net/spec/arml/2.0/conf/core/VisualAsset/projection_order	
Title	VisualAsset projection order
Abbreviation	VisualAsset/projection_order
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/VisualAsset/projection_order
Test Purpose	To validate that an implementation occludes objects according to their distance, in a natural way.
Test Method	Verify that the implementation projects the VisualAssets onto the screen according to their distance, with Assets closer to the user occluding Assets further away.

A.2.32 VisualAsset2D width and height	
http://www.opengis.net/spec/arml/2.0/conf/core/VisualAsset2D/width_and_height	
Title	VisualAsset2D width and height
Abbreviation	VisualAsset2D/width_and_height
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/VisualAsset2D/width_and_height
Test Purpose	To validate that an implementation adheres to the calculation rules of width and height for VisualAsset2Ds, as defined in the specification.
Test Method	Verify that the implementation calculates width and height respectively according to the following rules: - If only one of width and height is set, the other value is calculated on the aspect ratio of the VisualAsset2D. - If only one of width and height is set for Fill, the missing value is always set to 100% - If neither width, nor height is set, width is set to 100% and height is

	calculated according to the aspect ratio of the VisualAsset2D

A.2.33 VisualAsset2D orientation mode	
http://www.opengis.net/spec/arml/2.0/conf/core/VisualAsset2D/orientationMode	
Title	VisualAsset2D orientation mode
Abbreviation	VisualAsset2D/orientationMode
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/VisualAsset2D/orientationMode
Test Purpose	To validate that an implementation adheres to the calculation rules of orientations for VisualAsset2Ds, as defined in the specification.
Test Method	Verify that the implementation calculates the orientation of a VisualAsset according to the following rules: - If orientationMode is set to user, the VisualAsset2D is oriented towards the user - If orientationMode is set to absolute, VisualAsset2D is positioned according to the coordinate system specification of the VisualAsset and the Anchor - If orientationMode is set to auto, the orientationMode is implicitely set to absolute when the VisualAsset2D is attached to a Trackable or a RelativeTo Anchor referencing a Trackable, and is set to user otherwise.

A.2.34 VisualAsset2D backside	
http://www.opengis.net/spec/arml/2.0/conf/core/VisualAsset2D/backSide	
Title	VisualAsset2D backside
Abbreviation	VisualAsset2D/backside
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/VisualAsset2D/backSide
Test Purpose	To validate that an implementation adheres to the definitions for the backSide of a VisualAsset2D, as defined in the specification.
Test Method	Verify that the Implementation draws the back side of a VisualAsset2D according to the following rules: - If backside is set to a hex value, the backside is painted in that color - If backside is set to *mirrored*, the front side is mirrored onto the back side. - If backside is set to *copied*, the font side is copied onto the back side.

A.2.35 Label content precedence	
http://www.opengis.net/spec/arml/2.0/conf/core/Label/href_and_src_precedence	
Title	Label content precedence
Abbreviation	Label/href_and_scr_precedence
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/Label/href_and_src_precedence
Test Purpose	To validate that an implementation adheres to the precedence rules for the content of a Label.
Test Method	Verify that the implementation gives src precedence over href in case both properties are set.

A.2.36 Label content required	
http://www.opengis.net/spec/arml/2.0/conf/core/Label/href_and_src_required	
Title	Label content required
Abbreviation	Label/href_and_scr_required
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/Label/href_and_src_required
Test Purpose	To validate that the implementation does not fail in case a Label is invalid.
Test Method	Verify that the implementation ignores Labels that have both src and href unset.

A.2.37 Label hyperlink behavior	
http://www.opengis.net/spec/arml/2.0/conf/core/Label/hyperlinkBehavior	
Title	Label hyperlink behavior
Abbreviation	Label/hyperlinkBehavior
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/Label/hyperlinkBehavior
Test Purpose	To validate that an implementation adheres to the click behavior rules of a Label, as defined in the specification.
Test Method	Verify that the implementation handles clicks on hyperlinks in Labels according to the following rules: - If the Label's hyperlinkBehavior is set to block, links are not followed. - If the Label's hyperlinkBehavior is set to blank, links are opened in full screen browser windows. - If the Label's hyperlinkBehavior is set to self, links are opened inside the Label

A.2.38 Label default hyperlink behavior	
http://www.opengis.net/spec/arml/2.0/conf/core/Label/hyperlinkBehavior_default	
Title	Label default hyperlink behavior
Abbreviation	Label/hyperlinkBehavior_default
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/Label/hyperlinkBehavior_default
Test Purpose	To validate that an implementation sets the correct default value for the hyperlink behavior of a Label.
Test Method	Verify that the Implementation sets hyperlinkBehavior to blank when not set.

A.2.39 Label default viewport	
http://www.opengis.net/spec/arml/2.0/conf/core/Label/viewportWidth_default	
Title	Label default viewport
Abbreviation	Label/viewportWidth_default
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/Label/viewportWidth_default
Test Purpose	To validate that an implementation sets the correct default value for the viewport of a Label.
Test Method	Verify that the Implementation sets viewportWidth to 256 when not set or set to a negative value.

A.2.40 Label name and description replacement	
http://www.opengis.net/spec/arml/2.0/conf/core/Label/metadata_name_description	
Title	Label name and description replacement
Abbreviation	Label/metadata_name_description
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/Label/metadata_name_description
Test Purpose	To validate that an implementation replaces the placeholders for references to the name and description in a Label correctly.
Test Method	Verify that the Implementation replaces any occurrences of $[name] and $[description] in the content of the Label with name and description of the Feature's metadata the Label is linked to. Verify that, in case a property is not set, the implementation replaces the value with an empty string.

A.2.41 Label property replacement	
http://www.opengis.net/spec/arml/2.0/conf/core/Label/metadata_general	
Title	Label property replacement
Abbreviation	Label/metadata_general
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/Label/metadata_general
Test Purpose	To validate that an implementation replaces the placeholders for references to arbitrary properties in a Label correctly.
Test Method	Verify that the Implementation replaces any occurrences of $[(XPath-Expression)] in the content of the Label with the corresponding metadata content of the Feature's metadata the Label is linked to. Verify that the replacement is only performed when the XPath evaluation returns a single text node, and is replaced with an empty string otherwise.

A.2.42 Fill default color	
http://www.opengis.net/spec/arml/2.0/conf/core/Fill/color_default	
Title	Fill default color
Abbreviation	Fill/color_default
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/Fill/color_default
Test Purpose	To validate that an implementation sets the correct default color of a Fill element.
Test Method	Verify that the Implementation sets a Fill's color property to #000000 (black) when not set.

A.2.43 Text name and description replacement	
http://www.opengis.net/spec/arml/2.0/conf/core/Text/metadata_name_description	
Title	Text name and description replacement
Abbreviation	Text/metadata_name_description
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/Text/metadata_name_description
Test Purpose	To validate that an implementation replaces the placeholders for references to the name and description in a Text correctly.

Test Method	Verify that the Implementation replaces any occurrences of $[name] and $[description] in the content of the Text with name and description of the Feature's metadata the Text is linked to. Verify that, in case a property is not set, the implementation replaces the value with an empty string.

A.2.44 Text property replacement	
http://www.opengis.net/spec/arml/2.0/conf/core/Text/metadata_general	
Title	Text property replacement
Abbreviation	Text/metadata_general
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/Text/metadata_general
Test Purpose	To validate that an implementation replaces the placeholders for references to arbitrary properties in a Text correctly.
Test Method	Verify that the Implementation replaces any occurrences of $[(XPath-Expression)] in the content of the Text with the corresponding metadata content of the Feature's metadata the Text is linked to. Verify that the replacement is only performed when the XPath evaluation returns a single text node, and is replaced with an empty string otherwise.

A.2.45 Text default font color	
http://www.opengis.net/spec/arml/2.0/conf/core/Text/font-color_default	
Title	Text default font color
Abbreviation	Text/font-color_default
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/Text/font-color_default
Test Purpose	To validate that an implementation sets the default font color value of a Text element correctly.
Test Method	Verify that the Implementation sets a Text's font-color property to #000000 (black) when not set.

A.2.46 Text default background color	
http://www.opengis.net/spec/arml/2.0/conf/core/Text/background-color_default	
Title	Text default background color
Abbreviation	Text/background-color_default
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/Text/background-color_default
Test Purpose	To validate that an implementation sets the default background color value of a Text element correctly.
Test Method	Verify that the Implementation sets a Text's background-color property to #00000000 (transparent) when not set.

A.2.47 Invalid Image format	
http://www.opengis.net/spec/arml/2.0/conf/core/Image/formats	
Title	Invalid Image format
Abbreviation	Image/formats
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/Image/formats
Test Purpose	To validate that the implementation does not fail in case an Image is invalid.

Test Method	Verify that the Implementation ignores an Image when it cannot support the format of the Image.

A.2.48 Invalid Model format	
http://www.opengis.net/spec/arml/2.0/conf/core/Model/formats	
Title	Invalid Model format
Abbreviation	Model/formats
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/Model/formats
Test Purpose	To validate that the implementation does not fail in case a Model's format is not supported.
Test Method	Verify that the Implementation ignores a Model when it cannot support the format of the Model.

A.2.49 Model Type Handling	
http://www.opengis.net/spec/arml/2.0/conf/core/Model/type	
Title	Model Type Handling
Abbreviation	Model/type
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/Model/type
Test Purpose	To validate that the implementation does not display Models of type infrastructure.
Test Method	Verify that the Implementation displays Models of type normal, and uses Models of type infrastructure solely for occlusion handling.

A.2.50 Model Default Type	
http://www.opengis.net/spec/arml/2.0/conf/core/Model/type_default	
Title	Model Default Type
Abbreviation	Model/type_default
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/Model/type_default
Test Purpose	To validate that the implementation sets the correct type value for a Model in case the property is not set specifically.
Test Method	Verify that the Implementation sets the type of a Model to normal if not set.

A.2.51 Default Scale properties	
http://www.opengis.net/spec/arml/2.0/conf/core/Scale/defaults	
Title	Default Scale properties
Abbreviation	Scale/defaults
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/Scale/defaults
Test Purpose	To validate that the implementation sets the correct default values for a Scale element in case the properties are not set specifically.
Test Method	Verify that the Implementation sets the x, y and z values respectively of Scale to 1 if not set.

A.2.52 Scale 3D Axis Transformation	
http://www.opengis.net/spec/arml/2.0/conf/core/Scale/axis	
Title	Scale 3D Axis Transformation
Abbreviation	Scale/axis
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/Scale/axis
Test Purpose	To validate that the implementation uses the correct 3D transformation mode.
Test Method	Verify that the Implementation does not change the axis and their dimensions when Scale is applied, only the VisualAsset itself is scaled.

A.2.53 2D VisualAssets automatic orientation	
http://www.opengis.net/spec/arml/2.0/conf/core/AutomaticOrientation_VisualAssets/2D	
Title	2D VisualAssets automatic orientation
Abbreviation	AutomaticOrientation_VisualAssets/2D
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/AutomaticOrientation_VisualAssets/2D
Test Purpose	To validate that the implementation adheres to the definitions of automatic orientation for 2D VisualAssets.
Test Method	Verify that the Implementation implements the automatic orientation of a 2D VisualAsset according to the definitions in paragraph 6.5.2.1.

A.2.54 3D VisualAssets dimension 0 automatic orientation	
http://www.opengis.net/spec/arml/2.0/conf/core/AutomaticOrientation_VisualAssets/3D_dim_0	
Title	3D VisualAssets dimension 0 automatic orientation
Abbreviation	AutomaticOrientation_VisualAssets/3D_dim_0
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/AutomaticOrientation_VisualAssets/3D_dim_0
Test Purpose	To validate that the implementation adheres to the definitions of automatic orientation for 3D VisualAssets.
Test Method	Verify that the Implementation implements the automatic orientation of a 3D VisualAsset according to the definitions in paragraph 6.5.2.2.

A.2.55 3D VisualAssets dimension 1 and 2 automatic orientation	
http://www.opengis.net/spec/arml/2.0/conf/core/AutomaticOrientation_VisualAssets/3D_dim_1_2	
Title	3D VisualAssets dimension 1 and 2 automatic orientation
Abbreviation	AutomaticOrientation_VisualAssets/3D_dim_1_2
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/AutomaticOrientation_VisualAssets/3D_dim_1_2
Test Purpose	To validate that the implementation does not project 3D VisualAssets onto 1D and 2D Anchors.
Test Method	Verify that the Implementation ignores any 3D Visual Assets attached to a 1- or 2-dimensional Anchor.

A.2.56 Order of Execution of Manual Orientation for VisualAssets	
http://www.opengis.net/spec/arml/2.0/conf/core/ManualOrientation_VisualAssets/order	
Title	Order of Execution of Manual Orientation for VisualAssets
Abbreviation	ManualOrientation_VisualAssets/order
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/ManualOrientation_VisualAssets/order
Test Purpose	To validate that the implementation executes 3D transformations in the correct order.
Test Method	Verify that the Implementation executes rotations in the following order: roll – tilt – heading

A.2.57 Axes Transformation when manually orientating VisualAssets	
http://www.opengis.net/spec/arml/2.0/conf/core/ManualOrientation_VisualAssets/axes	
Title	Axes Transformation when manually orientating VisualAssets
Abbreviation	ManualOrientation_VisualAssets/axes
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/ManualOrientation_VisualAssets/axes
Test Purpose	To validate that the implementation uses the correct 3D transformation mode.
Test Method	Verify that the Implementation does not change the axis and their dimensions when orientation is applied, only the VisualAsset itself is oriented.

A.2.58 Definition of Manual Orientation of VisualAssets	
http://www.opengis.net/spec/arml/2.0/conf/core/ManualOrientation_VisualAssets/application	
Title	Definition of Manual Orientation of VisualAssets
Abbreviation	ManualOrientation_VisualAssets/application
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/ManualOrientation_VisualAssets/application
Test Purpose	To validate that the implementation adheres to the definitions of manual orientation for VisualAssets.
Test Method	Verify that the Implementation implements the manual orientation of VisualAssets as specified in section 6.5.2.3.

A.2.59 Definition of ScalingDistances	
http://www.opengis.net/spec/arml/2.0/conf/core/Scaling_VisualAssets/minMaxScalingDistance	
Title	Definition of ScalingDistances
Abbreviation	Scaling_VisualAssets/minMaxScalingDistance
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/Scaling_VisualAssets/minMaxScalingDistance
Test Purpose	To validate that the implementation sets correct default values for min and maxScalingDistance in case it is not set explicitly.
Test Method	Verify that the Implementation implements the min and maxScalingDistance of

	VisualAssets as specified in section 6.5.3.

A.2.60 Definition of scalingFactor	
http://www.opengis.net/spec/arml/2.0/conf/core/Scaling_VisualAssets/scalingFactor	
Title	Definition of scalingFactor
Abbreviation	Scaling_VisualAssets/scalingFactor
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/ Scaling_VisualAssets/ scalingFactor
Test Purpose	To validate that the implementation implements the scalingFactor behavior correctly.
Test Method	Verify that the Implementation implements the scalingFactor of VisualAssets as specified in section 6.5.3.

A.2.61 Default minScalingDistance	
http://www.opengis.net/spec/arml/2.0/conf/core/Scaling_VisualAssets/minScalingDistance_default	
Title	Default minScalingDistance
Abbreviation	Scaling_VisualAssets/minScalingDistance_default
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/ Scaling_VisualAssets/minScalingDistance_default
Test Purpose	To validate that the implementation sets correct default value for the scaling factor in case it is not set explicitly.
Test Method	Verify that the Implementation sets minScalingDistance to 0 when not set.

A.2.62 Ignoring maxScalingDistance with Natural Scaling	
http://www.opengis.net/spec/arml/2.0/conf/core/Scaling_VisualAssets/maxScalingDistance_ignored	
Title	Ignoring maxScalingDistance with Natural Scaling
Abbreviation	Scaling_VisualAssets/maxScalingDistance_ignored
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/ Scaling_VisualAssets/maxScalingDistance_ignored
Test Purpose	To validate that the implementation does not take maxScalingDistance into account for natural scaling.
Test Method	Verify that the Implementation ignores maxScalingDistance when natural scaling is used.

A.2.63 Ignoring scalingFactor with Natural Scaling	
http://www.opengis.net/spec/arml/2.0/conf/core/Scaling_VisualAssets/scalingFactor_ignored	
Title	Ignoring scalingFactor with Natural Scaling
Abbreviation	Scaling_VisualAssets/scalingFactor_ignored
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/ Scaling_VisualAssets/scalingFactor_ignored
Test Purpose	To validate that the implementation does not take scalingFactor into account for natural scaling.

Test Method	Verify that the Implementation ignores scalingFactor when natural scaling is used.

A.2.64 And-Concatenation of multiple Conditions	
http://www.opengis.net/spec/arml/2.0/conf/core/Condition/multiple	
Title	And-Concatenation of multiple Conditions
Abbreviation	Condition/multiple
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/Condition/multiple
Test Purpose	To validate that the implementation complies with the AND concatenation rules of Conditions
Test Method	Verify that the Implementation only renders a VisualAsset when it either does not have a Condition attached, or all of the attached Conditions are met.

A.2.65 Maximum render distance	
http://www.opengis.net/spec/arml/2.0/conf/core/Condition/Distance/max	
Title	Maximum render distance
Abbreviation	Condition/Distance/max
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/Condition/Distance/max
Test Purpose	To validate that the implementation does not render VisualAssets that are too far away.
Test Method	Verify that the Implementation ignores all VisualAssets that have a greater distance than the distance specified in max in their attached DistanceCondition.

A.2.66 Minimum render distance	
http://www.opengis.net/spec/arml/2.0/conf/core/Condition/Distance/min	
Title	Minimum render distance
Abbreviation	Condition/Distance/min
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/Condition/Distance/min
Test Purpose	To validate that the implementation does not render VisualAssets that are too close.
Test Method	Verify that the Implementation ignores all VisualAssets that have a smaller distance than the distance specified in min in their attached DistanceCondition.

A.2.67 And-Concatenation of min and max distance	
http://www.opengis.net/spec/arml/2.0/conf/core/Condition/Distance/min_and_max	
Title	And-Concatenation of min and max distance
Abbreviation	Condition/Distance/min_and_max
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/Condition/Distance/min_and_max
Test Purpose	To validate that the implementation complies with the AND concatenation rules of the values in a DistanceConditions
Test Method	Verify that, when both min and max are set in a DistanceCondition, the Implementation only shows a VisualAsset when both conditions yield true.

A.2.68 Default listener for SelectedCondition	
http://www.opengis.net/spec/arml/2.0/conf/core/Condition/Selected/listener_default	
Title	Default listener for SelectedCondition
Abbreviation	Condition/Selected/listener_default
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/Condition/Selected/listener_default
Test Purpose	To validate that the implementation sets the correct default anchor value for a SelectedCondition in case it is not set explicitely.
Test Method	Verify that the Implementation sets the listener to anchor in a SelectedCondition when not set.

A.2.69 SelectedCondition application	
http://www.opengis.net/spec/arml/2.0/conf/core/Condition/Selected/selected	
Title	SelectedCondition application
Abbreviation	Condition/Selected/selected
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/core/Condition/Selected/selected
Test Purpose	To validate that the implementation only renders a VisualAsset when the conditions are met as defined in its SelectedCondition.
Test Method	Verify that the Implementation only renders a VisualAsset with a SelectedCondition when the listener is in selected state and selected is set to true, and the listener is in unselected state and selected is set to false respectively.

A.3: Conformance Test Class: Descriptive and Scripting Implementation

http://www.opengis.net/spec/arml/2.0/conf/scripting	
Target Type	Implementation
Dependency	http://www.opengis.net/spec/arml/2.0/conf/core
Tests described in this section shall be used to test conformance of an Implementation capable of handling ARML 2.0 files with descriptive and scripting content. All Conformance Tests in the Conformance Class http://www.opengis.net/spec/arml/2.0/conf/core must validate before the execution of the Conformance Tests in this Conformance Class.	

A.3.1 ARML Injection	
http://www.opengis.net/spec/arml/2.0/conf/scripting/general/arml_injection	
Title	ARML Injection
Abbreviation	general/arml_injection
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/scripting/general/arml_injection
Test Purpose	To validate that the implementation injects the arml object into the ECMAScript

	context.
Test Method	Verify that the implementation injects the arml object into the ECMAScript runtime at startup.

A.3.2 Object Access	
http://www.opengis.net/spec/arml/2.0/conf/scripting/general/object_access	
Title	Object Access
Abbreviation	general/object_access
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/scripting/general/object_access
Test Purpose	To validate that the implementation can access any ID-referenced object through the ECMAScript binding.
Test Method	Verify that, after an ARElement was added to the scene, its ECMAScript representation is accessible in the Implementation through *arml.getARElementById* with the id of the ARElement as parameter.

A.3.3 Declarative and Scripting Synchronization	
http://www.opengis.net/spec/arml/2.0/conf/scripting/general/synchronization	
Title	Declarative and Scripting Synchronization
Abbreviation	general/synchronization
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/scripting/general/synchronization
Test Purpose	To validate that the implementation synchronizes the descriptive and the scripting representation of the scene.
Test Method	Verify that the Implementation makes the values of the properties declared in the declarative specification accessible through ECMAScript.

A.3.4 Misuse of ECMAScript specification	
http://www.opengis.net/spec/arml/2.0/conf/scripting/general/misuse	
Title	Misuse of ECMAScript specification
Abbreviation	general/misuse
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/scripting/general/misuse
Test Purpose	To validate that the implementation raises an error in case the defined constructors, methods and properties are not used in the intended way.
Test Method	Verify that the Implementation raises an ECMAScript Error when constructors, methods and properties are misused.

A.3.5 ARML Injection into Label	
http://www.opengis.net/spec/arml/2.0/conf/scripting/Label/injection	
Title	ARML Injection into Label
Abbreviation	Label/injection
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/scripting/Label/injection

Test Purpose	To validate that each Label can access to the arml object.
Test Method	Verify that the Implementation injects the arml root object into each Label before it is fully constructed.

A.3.6 Invalid animationId	
http://www.opengis.net/spec/arml/2.0/conf/scripting/Model/animationId	
Title	Invalid animationId
Abbreviation	Model/animationId
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/scripting/Model/animationId
Test Purpose	To validate that the implementation raises an error in case a specified animationId is not accessible.
Test Method	Verify that the Implementation raises an ECMAScript Error when the animationId cannot be resolved.

A.3.7 Animation loop count definition for Model Animations	
http://www.opengis.net/spec/arml/2.0/conf/scripting/Model/animationLoopCount	
Title	Animation loop count definition for Model Animations
Abbreviation	Model/animationLoopCount
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/scripting/Model/animationLoopCount
Test Purpose	To validate that the implementation sets the correct default value for a missing loopCount, and implements the correct looping behavior in Model animations.
Test Method	Verify that the Implementation sets the loopCount to 1 when not set, and verify that the animation will loop infinitely often when the value is set to -1.

A.3.8 Animation callback definition for Model Animations	
http://www.opengis.net/spec/arml/2.0/conf/scripting/Model/animationCallback	
Title	Animation callback definition for Model Animations
Abbreviation	Model/animationCallback
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/scripting/Model/animationCallback
Test Purpose	To validate that the implementation implements the correct animation callback behavior.
Test Method	Verify that the Implementation triggers the callback when the animation finished playing, but will not trigger it when the animation was stopped explicitly by calling stop3DAnimation.

A.3.9 Animation loop count definition for regular Animations	
http://www.opengis.net/spec/arml/2.0/conf/scripting/Animation/loopCount	
Title	Animation loop count definition for regular Animations
Abbreviation	Animation/animationLoopCount
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/scripting/Animation/loopCount
Test Purpose	To validate that the implementation sets the correct default value for a missing loopCount, and implements the correct looping behavior in Animations.

Test Method	Verify that the Implementation sets the loopCount to 1 when not set, and verify that the animation will loop infinitely often when the value is set to -1.

A.3.10 Default delay setting for Animations	
http://www.opengis.net/spec/arml/2.0/conf/scripting/Animation/delay	
Title	Default delay setting for Animations
Abbreviation	Animation/delay
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/scripting/Animation/delay
Test Purpose	To validate that the implementation sets the correct default value for a missing delay property in Animations.
Test Method	Verify that the Implementation sets the delay to 0 when not set.

A.3.11 Default animation start value	
http://www.opengis.net/spec/arml/2.0/conf/scripting/Animation/Number/start	
Title	Default animation start value
Abbreviation	Animation/Number/start
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/scripting/Animation/Number/start
Test Purpose	To validate that the implementation sets the correct default value for a missing start property in Animations.
Test Method	Verify that the Implementation sets the Animation's start value to the current value of the property when no explicit start value is given.

A.3.12 Animation end value	
http://www.opengis.net/spec/arml/2.0/conf/scripting/Animation/Number/end	
Title	Animation end value
Abbreviation	Animation/Number/end
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/scripting/Animation/Number/end
Test Purpose	To validate that the implementation sets the correct end value in Animations.
Test Method	Verify that, after the completion of the Animation, the Implementation sets the property's value to the specified end value.

A.3.13 Animation end definition	
http://www.opengis.net/spec/arml/2.0/conf/scripting/Animation/Group/endDefinition	
Title	Animation end definition
Abbreviation	Animation/Group/endDefinition
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/scripting/Animation/Group/endDefinition
Test Purpose	To validate that the implementation implements the correct behavior when an Animation is finished.
Test Method	Verify that the implementation considers an Animation group finished in the following cases: - The longest Animation has finished in a parallel group. - The last Animation has finished in a sequential group.

A.3.14 Event Handling	
http://www.opengis.net/spec/arml/2.0/conf/scripting/EventHandling/firing	
Title	Event Handling
Abbreviation	EventHandling/firing
Type	Basic
Requirement	http://www.opengis.net/spec/arml/2.0/req/scripting/EventHandling/firing
Test Purpose	To validate that the implementation implements the correct event handling mechanisms.
Test Method	Verify that the Implementation triggers the EventHandler when an event occurred for which the EventListener interface was registered.

Annex B: Examples

The following section provides some examples of ARML snippets in common use cases. All use cases assume the following:

- A valid COLLADA 3D Model (including correctly referenced texture images) exists at the following location: http://www.myserver.com/myModel.dae; the Model's coordinate system is left-handed (x points left, y points up, z points to the front)
- A 512x512 (arbitrary) image exists at the following location: http://www.myserver.com/myImage.jpg
- A 512px wide and 1024px high artificial marker exists at the following location: http://www.myserver.com/myMarker.jpg. When printed, the marker is 20cm wide and 40cm high.

B1: Typical geospatial AR Browser

A typical geospatial AR Browser shows placemarks, referenced by latitude and longitude values, as icons on the camera screen. When the user clicks on a placemark, a static info window is shown at the bottom of the screen, displaying some textual information.

Remark: Descriptions of the Placemarks are taken from the Wikipedia pages of the Golden Gate Bridge and Coit Tower.

```
<arml>
  <ARElements>

   <!-- define the placemark marker; we use custom scaling mode to allow
markers to be visible from further away markers will appear 20 meters wide
as a maximum in the composed scene. The Image will be used by each
Placemark in the scene. -->

   <Image id="placemarkMarker">
     <ScalingMode type="custom">
       <minScalingDistance>10</minScalingDistance>
       <maxScalingDistance>1000</maxScalingDistance>
       <scalingFactor>0.4</scalingFactor>
     </ScalingMode>
     <width>20</width>
     <href xlink:href="http://www.myserver.com/myImage.jpg" />
   </Image>

   <!-- define the info window. The info window is located at the bottom of
the screen and displays the name and description of the Feature it is
attached to. It will only be visible when the particular Feature
(placemark) was selected, and will disappear as soon as the Feature is
unselected. The Anchor will be used by each Placemark in the scene. -->

   <ScreenAnchor id="infoWindow">
     <style>left: 0; width: 100%; bottom: 0; height: 25%</style>
     <assets>
       <Label>
         <conditions>
           <SelectedCondition>
```

```
            <listener>feature</listener>
            <selected>true</selected>
          </SelectedCondition>
        </conditions>
        <src><b>$[name]</b><br />$[description]</src>
      </Label>
    </assets>
  </ScreenAnchor>

  <!-- Golden Gate Placemark -->

  <Feature id="goldenGateBridge">
   <name>Golden Gate Bridge</name>
   <description>The Golden Gate Bridge is a suspension bridge spanning the
Golden Gate, the opening of the San Francisco Bay into the Pacific
Ocean.</description>
    <anchors>
     <anchorRef xlink:href="#infoWindow" />
      <Geometry>
       <assets><assetRef xlink:href="#placemarkMarker" /></assets>
        <gml:Point gml:id="myPoint">
          <gml:pos>37.818599 -122.478511</gml:pos>
        </gml:Point>
      </Geometry>
    </anchors>
  </Feature>

  <!-- Coit Tower Placemark -->

  <Feature id="coitTower">
   <name>Coit Tower</name>
   <description>Coit Tower, also known as the Lillian Coit Memorial Tower,
is a 210-foot (64 m) tower in the Telegraph Hill neighborhood of San
Francisco, California.</description>
    <anchors>
     <anchorRef xlink:href="#infoWindow" />
     <Geometry>
      <assets><assetRef xlink:href="#placemarkMarker" /></assets>
        <gml:Point gml:id="myPoint">
          <gml:pos>37.802494 -122.405727</gml:pos>
        </gml:Point>
     </Geometry>
   </anchors>
  </Feature>

 </ARElements>
</arml>
```

B2: Different Representations based on Distance

The Golden Gate Bridge example from above will be reused, but this time, the Golden Gate Bridge should appear as a (scaled) icon when viewed from more than 5 kilometers away, as a red colored line when viewed from between 1 and 5 kilometers away, and as a 3D model showing the bridge just after its completion when viewed from less than 1 kilometer away.

```
<arml>
  <ARElements>
    <Image id="placemarkMarker">
      <conditions>
        <DistanceCondition>
          <min>5000</min>
        </DistanceCondition>
      </conditions>
      <ScalingMode type="custom">
        <minScalingDistance>10</minScalingDistance>
        <maxScalingDistance>1000</maxScalingDistance>
        <scalingFactor>0.4</scalingFactor>
      </ScalingMode>
      <width>20</width>
      <href xlink:href="http://www.myserver.com/myImage.jpg" />
    </Image>

    <Fill id="myRedFill">
      <!-- only visible when 1km <= distance <= 5km -->
      <conditions>
        <DistanceCondition>
          <max>5000</max>
          <min>1000</min>
        </DistanceCondition>
      </conditions>

      <!-- the Golden Gate Bridge is 27.4 meters wide, thus the height of
the Fill (which represents the width of the Bridge) is set to 27.4 meters -
->
      <height>27.4</height>
      <!--  red color -->
      <style>color:#FF0000;</style>
    </Fill>

    <Model id="3dModel">
      <!-- only visible when distance <= 1km -->
      <conditions>
        <DistanceCondition>
          <max>1000</max>
        </DistanceCondition>
      </conditions>
      <href xlink:href="http://www.myserver.com/myModel.dae" />
    </Model>

    <!-- Golden Gate Placemark -->

    <Feature id="goldenGateBridge">
      <name>Golden Gate Bridge</name>
      <anchors>
        <Geometry>
          <assets>
            <!-- the model and the icon are mapped onto the same point, but
shown at different distances (see the VisualAssets declaration on top for
details) -->
            <assetRef xlink:href="#placemarkMarker" />
```

```
            <assetRef xlink:href="#3dModel" />
          </assets>
          <gml:Point gml:id="myPoint">
            <gml:pos>37.818599 -122.478511</gml:pos>
          </gml:Point>
        </Geometry>

        <Geometry>
          <!-- the line-representation must be mapped as a LineString
Geometry -->
          <assets><assetRef xlink:href="#filledLine" /></assets>
          <gml:LineString gml:id="myLineString">
            <gml:posList>
              37.827752 -122.479541 37.811005 -122.477739
            </gml:posList>
          </gml:LineString>
        </Geometry>
      </anchors>
    </Feature>

  </ARElements>
<arml>
```

B3: 3D Model on a Trackable

The 3D Model should appear on top of the referenced marker to play a game etc.

```
<arml>
  <ARElements>
    <!-- register the Tracker to track a generic image -->
    <Tracker id="defaultImageTracker">
      <uri
xlink:href="http://www.opengis.net/arml/tracker/genericImageTracker" />
    </Tracker>

    <!-- define the artificial marker the Model will be placed on top of --
>
    <Trackable>
      <assets>
        <!-- define the 3D Model that should be visible on top of the
marker -->
        <Model>
          <href xlink:href="http://www.myserver.com/myModel.dae" />
        </Model>
      </assets>
      <config>
        <tracker xlink:href="#defaultImageTracker" />
        <src>http://www.myserver.com/myMarker.jpg</src>
      </config>
      <size>0.20</size>
    </Trackable>

  </ARElements>
</arml>
```

B4: Color the Outline of the artificial marker

Use case: When the marker is detected in the camera screen, a red line, 1 centimeter wide, should be drawn around the marker (the marker outline).

```
<arml>

  <ARElements>

    <!-- define the VisualAsset for the outline - the LineString will be
filled with red color -->
    <Fill id="myRedFill">
      <!-- height set to 0.01 causes the LineString to be drawn 1cm thick -
->
      <height>0.01</height>
      <!-- define red color for the fill -->
      <style>color:#FF0000;</style>
    </Fill>

    <!-- define the Tracker and the Marker (see previous example) -->
    <Tracker id="defaultImageTracker">
      <uri
xlink:href="http://www.opengis.net/arml/tracker/genericImageTracker" />
    </Tracker>
    <Trackable id="myTrackable">
      <config>
        <tracker xlink:href="#defaultImageTracker" />
        <src>http://www.myserver.com/myMarker.jpg</src>
      </config>
      <size>0.20</size>
    </Trackable>

    <!-- defines the location of the outline of the marker as a LineString
which has to be defined relative to the Trackable's center point -->
    <RelativeTo id="markerOutline">
      <assets>
        <!-- use the Fill-VisualAsset defined above to draw the LineString
-->
        <assetRef xlink:href="#myRedFill" />
      </assets>
      <!-- reference the Trackable the RelativeTo-geometry will be using --
>
      <ref xlink:href="#myTrackable" />
      <!-- define the Outline as LineString, from the top right corner of
the marker, moving clockwise. The top right point is 10 centimeters to the
right, 20 centimeters to the top and 0 centimeters above the Trackable's
center (0.01, 0.02 and 0 meters). -->
      <gml:LineString gml:id="myLineString">
        <gml:posList>0.01 0.02 0 0.01 -0.02 0 -0.01 -0.02 0 -0.01 0.02 0
0.01 0.02 0</gml:posList>
      </gml:LineString>
    </RelativeTo>

  </ARElements>
</arml>
```

B5: Color the entire area of a marker

The use case above can be slightly altered to color the entire marker area instead of just the outline, only the LineString-element must be significantly changed, while the Fill-element is implicitly set back to 100% width and height, causing the entire marker area to be filled.

```
<arml>

  <ARElements>

    <!-- define the VisualAsset for the colored area -->
    <Fill id="myRedFill">
      <!-- define red color for the fill -->
      <style>color:#FF0000;</style>
    </Fill>

    <!-- define the Tracker and the Marker (see previous example) -->
    <Tracker id="defaultImageTracker">
      <uri
xlink:href="http://www.opengis.net/arml/tracker/genericImageTracker" />
    </Tracker>
    <Trackable id="myTrackable">
      <config>
        <tracker xlink:href="#defaultImageTracker" />
        <src>http://www.myserver.com/myMarker.jpg</src>
      </config>
      <size>0.20</size>
    </Trackable>

    <!-- defines the location of the area of the marker as a Polygon which
has to be defined relative to the Trackable's center point -->
    <RelativeTo id="markerOutline">
      <assets>
        <!-- use the Fill-VisualAsset defined above to draw the LineString
-->
        <assetRef xlink:href="#myRedFill" />
      </assets>
      <!-- reference the Trackable the RelativeTo-geometry will be using --
>
      <ref xlink:href="#myTrackable" />
      <!-- define the Outline as LineString, from the top right corner of
the marker, moving clockwise. The top right point is 10 centimeters to the
right, 20 centimeters to the top and 0 centimeters above the Trackable's
center (0.01, 0.02 and 0 meters). -->
      <gml:Polygon gml:id="myPolygon">
        <gml:exterior>
          <gml:LinearRing>
            <gml:posList>0.01 0.02 0 0.01 -0.02 0 -0.01 -0.02 0 -0.01 0.02
0 0.01 0.02 0</gml:posList>
          </gml:LinearRing>
        </gml:exterior>
      </gml:Polygon>
    </RelativeTo>
```

```
  </ARElements>
</arml>
```

Annex C: Revision history

Date	Release	Author	Paragraph modified	Description
2012-10-31	1.0.0	Martin Lechner	All	Copy from TWiki to this document for RFC
2012-11-02	1.0.0	Martin Lechner	Sections 1,2,4,5,7	Fixed some broken Links, added historical information on ARML 1.0
2012-12-07	1.0.0	Martin Lechner	Sections 6,7,8,9	Incorporated comments received during the public commenting phase
2014-11-26	1.0.0	Martin Lechner	All	Added requirements and conformance classes into the document, along with some minor editorial updates. Final editorial changes before Adoption Vote.
2015-02-09	1.0.0	Martin Lechner	Section 6	Fixed typo error as reported in the adoption vote
2015-02-12	1.0.0	Carl Reed, Scott Simmons	multiple	OGC staff edits prior to publication
2015-02-20	1.0.0	Martin Lechner	Annex	Moved Section 9 to Annex A, moved Annex A, B and C to B, C and D; Added Test Purpose, Type, Name and Abbreviation to Conformance Tests

Annex D: Bibliography

[AR Glossary] - http://www.perey.com/ARStandards/AR_Glossary_2.2_May_3.pdf
[Wikipedia AR Definition] - http://en.wikipedia.org/wiki/Augmented_reality
[Ronald Azuma AR Definition] - http://www.cs.unc.edu/~azuma/ARpresence.pdf
[EPSG Codes] - http://spatialreference.org/ref/epsg/
[ARML 1.0 Specification] - http://openarml.org

