OGC® DOCUMENT: 24-006R4

External identifier of this OGC® document: http://www.opengis.net/doc/is/tdml-part2/1.0

Open Geospatial Consortium

OGC TRAINING DATA MARKUP LANGUAGE FOR ARTIFICIAL INTELLIGENCE (TRAININGDML-AI) PART 2: JSON ENCODING STANDARD

STANDARD
Encoding

APPROVED

Version: 1.0.1
Submission Date: 2026-03-01
Approval Date: 2026-04-28
Publication Date: 2026-08-20
Editor: Peng Yue, Ruixiang Liu, Boyi Shangguan

Note

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.

CONTENTS

I ABSTRACT

The OGC Training Data Markup Language for Artificial Intelligence (TrainingDML-AI) Part 2: JSON Encoding Standard defines requirements for encoding AI training datasets as JavaScript Object Notation (JSON). JSON is widely used for encoding data in Web-based applications. It consists of sets of objects described by name/value pairs. TrainingDML-AI Part 2 is based on the OGC Training Data Markup Language for Artificial Intelligence (TrainingDML-AI) Part 1: Conceptual Model Standard.

II KEYWORDS

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

ogcdoc, OGC document, artificial intelligence, machine learning, deep learning, earth observation, remote sensing, training data, training sample, encoding, JSON

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.

IV SECURITY CONSIDERATIONS

No security considerations have been made for this Standard.

V SUBMITTING ORGANIZATIONS

The following organizations submitted this Document to the Open Geospatial Consortium (OGC):

- Wuhan University
- Luoyao Information Technology Co., Ltd
- Pixalytics Ltd
- WiSC Enterprises
- George Mason University
- Laboratoire d'Informatique de Grenoble
- Wuhan University of Technology
- Hubei University
- Chongqing Changan Zhitu Technology Co., Ltd

VI SUBMITTERS

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

NAME	AFFILIATION
Peng Yue	Wuhan University
Ruixiang Liu	Wuhan University
Haoru Wu	Wuhan University
Chenxiao Zhang	Wuhan University
Boyi Shangguan	Luoyao Information Technology Co., Ltd
Samantha Lavender	Pixalytics Ltd
Jim Antonisse	WiSC Enterprises

NAME	AFFILIATION
Liping Di	George Mason University
Eugene Yu	George Mason University
Danielle Ziébelin	Laboratoire d'Informatique de Grenoble
Liangcun Jiang	Wuhan University of Technology
Lei Hu	Hubei University
Mingda Zhang	Hubei University
Kai Yan	Chongqing Changan Zhitu Technology Co., Ltd

VII ACKNOWLEDGEMENTS

Thanks to the members of the TrainingDML-AI Standards Working Group of the OGC as well as all contributors of change requests and comments. In particular: Scott Simmons, Carl Reed, Sam Meek, Kaixuan Wang, Zhipeng Cao, Shuaiqi Liu, Ming Zhao, Hanwen Xu, Haipeng Deng, Baoxin Teng.

1 SCOPE

1 SCOPE

This OGC TrainingDML — AI Part 2: JSON Encoding Standard defines a JSON encoding for the exchange of training datasets. The TrainingDML — AI Part 2 Standard provides a JSON-based encoding for the exchange of information describing training datasets, both within and between different organizations.

The document model is derived from the conceptual models defined in the OGC Training Data Markup Language for Artificial Intelligence (TrainingDML-AI) Part 1: Conceptual Model Standard.

2 CONFORMANCE

2 CONFORMANCE

This document defines a JSON encoding for AI training datasets. The standardization target for this Standard is:

- TrainingDML-AI JSON Encoding Schema

Conformance with this Standard shall be checked using all the relevant tests specified in Annex A of this document. The framework, concepts, and methodology for testing, and the criteria to be achieved to claim conformance are specified in the OGC Compliance Testing Policies and Procedures and the OGC Compliance Testing website.

All requirements-classes and conformance-classes described in this document are owned by the standard identified.

This document identifies eleven (11) conformance classes. Each conformance class is defined by one requirements class. The tests in Annex A are organized by requirements class. So an implementation of each conformance class must pass all tests specified in Annex A for the respective requirements class.

Of these eleven conformance classes, only the AI_TrainingDataset conformance class is mandatory. All other conformance classes are optional. In the case where a conformance class has a dependency on another conformance class, that conformance class should also be implemented.

If AI_TrainingDataset conformance class is implemented, all other conformance classes will eventually be implemented based on the dependencies between the conformance classes. Therefore, all conformance classes can also be considered mandatory.

3 NORMATIVE REFERENCES

3 NORMATIVE REFERENCES

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IETF: RFC 7159, The JavaScript Object Notation (JSON) Data Interchange Format, 2014

IETF: RFC 7946, The GeoJSON Format, 2016

IETF: RFC 3986, Uniform Resource Identifiers (URI): Generic Syntax, 2005

IETF: RFC 3339, Date and Time on the Internet: Timestamps, 2002

IETF: RFC 2046, Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types, 1996

ISO/IEC 21778:2017 Information technology — The JSON data interchange syntax

ISO 19101-1:2014 Geographic information — Reference model — Part 1: Fundamentals

ISO 19103:2024 Geographic information — Conceptual schema language

ISO 19115-1:2014 Geographic information — Metadata — Part 1: Fundamentals

ISO 19156:2023 Geographic information — Observations, measurements and samples

ISO 19157-1:2023 Geographic information — Data quality — Part 1: General requirements

4 TERMS AND DEFINITIONS

4 TERMS AND DEFINITIONS

This document uses the terms defined in OGC Policy Directive 49, which is based on the ISO/IEC Directives, Part 2, Rules for the structure and drafting of International Standards. In particular, the word “shall” (not “must”) is the verb form used to indicate a requirement to be strictly followed to conform to this document and OGC documents do not use the equivalent phrases in the ISO/IEC Directives, Part 2.

This document also uses terms defined in the OGC Standard for Modular specifications (OGC 08-131r3), also known as the ‘ModSpec’. The definitions of terms such as standard, specification, requirement, and conformance test are provided in the ModSpec.

For the purposes of this document, the following additional terms and definitions apply.

This document used the terms defined in OGC Policy Directive 49, which is based on the ISO/IEC Directives, Part 2, Rules for the structure and drafting of International Standards. In particular, the word “shall” (not “must”) is the verb form used to indicate a requirement to be strictly followed to conform to this Standard and OGC documents do not use the equivalent phrases in the ISO/IEC Directives, Part 2.

For the purposes of this document, the following additional terms and definitions apply.

4.1. **Artificial Intelligence (AI)**

refers to a set of methods and technologies that can empower machines or software to learn and perform tasks like humans.

SOURCE: OGC Training Data Markup Language for Artificial Intelligence (TrainingDML-AI) Part 1: Conceptual Model Standard

4.2. **Machine Learning (ML)**

is an important branch of artificial intelligence that gives computers the ability to improve their performance without explicitly being programmed to do so. ML processes create models from training data by using a set of learning algorithms, and then can use these models to make predictions. Depending on whether the training data include labels, the learning algorithms can be divided into supervised and unsupervised learning.

SOURCE: OGC Training Data Markup Language for Artificial Intelligence (TrainingDML-AI) Part 1: Conceptual Model Standard

4.3. Deep Learning (DL)

is a subset of machine learning, which is essentially a neural network with three or more layers. The number of layers is referred to as depth. While a neural network with a single layer can still make approximate predictions, additional hidden layers can help to optimize and refine for accuracy.

SOURCE: https://www.ibm.com/topics/deep-learning

4.4. Dataset

identifiable collection of data

[SOURCE: ISO 19115-1:2014, 4.3]

Note 1 to entry: A dataset can be a smaller grouping of data which, though limited by some constraint such as spatial extent or feature type, is located physically within a larger dataset. Theoretically, a dataset can be as small as a single feature or feature attribute contained within a larger dataset. A hardcopy map or chart can be considered a dataset.

4.5. Training Dataset

a collection of samples, often labelled with known terms or expected values for supervised learning. A training dataset can be divided into training, validation, and test sets. Training samples are different from samples in OGC Observations & Measurements (O&M). They are often collected in purposive ways that deviate from purely probability sampling, with known or expected results labeled as values of a dependent variable for generating a trained predictive model.

SOURCE: OGC Training Data Markup Language for Artificial Intelligence (TrainingDML-AI) Part 1: Conceptual Model Standard

4.6. Label

refers to known or expected results annotated as values of a dependent variable in training samples. A training sample label is different from those on a geographical map, which are known as map labels or annotations.

SOURCE: OGC Training Data Markup Language for Artificial Intelligence (TrainingDML-AI) Part 1: Conceptual Model Standard

Note 1 to entry: A training sample label is different from those on a geographical map, which are known as map labels or annotations.

4.7. Class

<classification> result of a classification process as part of a classification system which subdivides concepts within a given topic area.

[SOURCE: ISO 19144-2:2023, 3.1.6]

4.8. Task

the specific goal that an AI application want to achieve.

SOURCE: OGC Training Data Markup Language for Artificial Intelligence (TrainingDML-AI) Part 1: Conceptual Model Standard

4.9. Provenance

information about entities, activities, and people involved in producing a piece of data or thing, which can be used to form assessments about its quality, reliability or trustworthiness. In this standard provenance is a record of how training data were prepared.

SOURCE: W3C (https://www.w3.org/TR/prov-overview/)

4.10. Quality

degree to which a set of inherent characteristics of an object fulfils requirements [ISO 9000:2015, 3.6.2, modified — Notes 1 and 2 to entry have been deleted]. Quality of training data (such as data imbalance and mislabeling) can impact the performance of AI/ML models.

SOURCE: OGC Training Data Markup Language for Artificial Intelligence (TrainingDML-AI) Part 1: Conceptual Model Standard

4.11. Earth Observation

data and information collected about our planet, whether atmospheric, oceanic or terrestrial. This includes space-based or remotely-sensed data, as well as ground-based or in situ data.

SOURCE: GEO (https://earthobservations.org/geo_wwd.php)

4.12. Scene Classification

task of identifying scene categories of images, on the basis of a training set of images whose scene categories are known.

SOURCE: OGC Training Data Markup Language for Artificial Intelligence (TrainingDML-AI) Part 1: Conceptual Model Standard

4.13. Object Detection

task of recognizing objects such as cars from images. The objects are often localized using bounding boxes.

SOURCE: OGC Training Data Markup Language for Artificial Intelligence (TrainingDML-AI) Part 1: Conceptual Model Standard

4.14. Semantic Segmentation

task of assigning class labels to pixels of images or points of point clouds.

SOURCE: OGC Training Data Markup Language for Artificial Intelligence (TrainingDML-AI) Part 1: Conceptual Model Standard

4.15. Change Detection

recognition of changes between images acquired at different times.

SOURCE: OGC Training Data Markup Language for Artificial Intelligence (TrainingDML-AI) Part 1: Conceptual Model Standard

4.16. 3D Model Reconstruction

task that builds 3D objects and scenes from multi-view images.

SOURCE: OGC Training Data Markup Language for Artificial Intelligence (TrainingDML-AI) Part 1: Conceptual Model Standard

4.17. Generative Model

is one of the methods of large model training, which improve model performance through unsupervised pre-training. In the fine-tuning phase, labeled data plays a critical role in optimizing the model for specific vertical domains or tasks. By incorporating labeled data, the model can learn to accurately identify and extract relevant features, leading to better performance on specific downstream tasks. Overall, the combination of generative models and fine-tuning with labeled data can significantly improve the performance of large models in specialized domains or tasks.

SOURCE: OGC Training Data Markup Language for Artificial Intelligence (TrainingDML-AI) Part 1: Conceptual Model Standard

4.18. JavaScript Object Notation (JSON)

is a lightweight, text-based, language-independent syntax for defining data interchange formats. It was derived from the ECMAScript programming language but is programming language independent. JSON defines a small set of structuring rules for the portable representation of structured data.

SOURCE: ISO/IEC 21778:2017

4.19. JSON Schema

is a vocabulary that allows you to annotate and validate JSON documents.

SOURCE: https://json-schema.org/

4.20. **Training Dataset Publisher**

refers to the entity or individual responsible for creating and releasing the JSON-based serialization syntax for geospatial training datasets, as defined in the TrainingDML-AI Part 2: JSON Encoding Standard.

5 CONVENTIONS

5 CONVENTIONS

This section provides details and examples for any conventions used in the document.

5.1. Identifiers

The normative provisions in this Standard are denoted by the URI:

http://www.opengis.net/spec/TrainingDML-AI-2/1.0

All requirements and conformance tests that appear in this document are denoted by partial URIs which are relative to this base.

5.2. Abbreviated Terms

In this document, the following abbreviations and acronyms are used or introduced:

AI	Artificial Intelligence
DL	Deep Learning
EO	Earth Observation
IETF	Internet Engineering Task Force
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
ML	Machine Learning
OGC	Open Geospatial Consortium
RS	Remote Sensing
TD	Training Data
UML	Unified Modelling Language
URL	Uniform Resource Locator

UTC	Coordinated Universal Time
W3C	World Wide Web Consortium
XML	Extensible Markup Language

6 OVERVIEW

6 OVERVIEW

The TrainingDML-AI Part 2: JSON Encoding Standard defines a JSON-based serialization syntax for geospatial training datasets. While other serialization formats are possible, such alternatives are not discussed in this Standard.

JSON does not have a formal class model. JSON objects are just sets of properties. However, the JSON encoding described in this Standard features a “type” property on each JSON object.

A training dataset document conforming to this Standard is a JSON document whose root value is an AI_TrainingDataset object.

6.1. JavaScript Object Notation

JavaScript Object Notation (JSON) is a lightweight, text-based, language-independent data interchange format that defines a small set of formatting rules for the portable representation of structured data. JSON is derived from the object literals of JavaScript, as defined in the ECMAScript Programming Language Standard and can represent four primitive types (strings, numbers, Boolean values, and null) and two structured types (objects and arrays). The ordering of the members or properties of any JSON object is considered irrelevant. Even though JSON is based on a subset of the JavaScript Programming Language it is currently well-supported by nearly all programming languages, including Java, Python, and C#.

The JSON format is currently described by two competing standards, RFC7159 and ISO/IEC 21778:2017. Both standards documents are consistent, but the latter defines mainly the grammatical syntax where the former provides some additional semantic and security points.

When serialized, absent properties are represented by either (a) setting the property value to null, or (b) by omitting the property declaration altogether at the option of the training dataset publisher. These representations are semantically equivalent. If a property has an array value, the absence of any items in that array shall be represented by omitting the property entirely or by setting the value to null. The appropriate interpretation of an omitted or explicitly null value is that no value has been assigned, as opposed to the view that the given value is empty or nil.

7

REQUIREMENTS FOR TRAININGDML-AI JSON ENCODING

7 REQUIREMENTS FOR TRAININGDML-AI JSON ENCODING

7.1. Requirements Class: Base

7.1.1. Requirements Class: JSON Base Type

The JSON Base Type requirements class defines the base requirements for JSON encodings, which includes definitions of common types used in the TrainingDML-AI JSON encoding.

REQUIREMENTS CLASS	
/req/base/jsonbasetype	
Dependency	JSON
Requirement 1	/req/base/jsonbasetype/json
Requirement 2	/req/base/jsonbasetype/datetime
Requirement 3	/req/base/jsonbasetype/namedvalue
Requirement 4	/req/base/jsonbasetype/url

The first requirement is that a TrainingDML-AI JSON document is a valid JSON document.

Requirement 1	/req/base/jsonbasetype/json An instance SHALL be a conformant JSON document, as defined in ISO/IEC 21778:2017

JSON has a limited range of built-in types (http://json.org/). The following requirements provide standard JSON representations of additional types required across all requirements within this specification.

The DateTime is encoded as a text string.

Requirement 2	/req/base/jsonbasetype/datetime Each DateTime value SHALL be encoded as a text string defined in Date and Time on the Internet: Timestamps [RFC 3339 Section 5.6], and SHALL support both the date-time and full-date formats. RFC 3339 is a profile of the ISO 8601 standard for representation of dates and times using the Gregorian calendar. The specification of date and time in any JSON encoding of training set data SHALL be specified in UTC.

Examples:

```
a) "2002-05-30T09:30:10Z"
b) "2002-09-24"
```

The NamedValue is encoded as a JSON object with two properties named “key” and “value”. Typically, the “key” property represents a unique identifier or name for the value being described, while the “value” property contains the actual data associated with that identifier. The “key” element is an open field allowing for arbitrary keys to be used. The “value” element may be any types.

Requirement 3	/req/base/jsonbasetype/namedvalue Each NamedValue value SHALL be encoded as a JSON object with properties “key” and “value”, while the value of property “key” is a text string.

Examples:

```
a) {
     "key": "forest",
     "value": "RGB(0,255,255)"
   }
b) {
     "key": "precision",
     "value": 0.8
   }
```

The URL is encoded as a text string.

Requirement 4	/req/base/jsonbasetype/url Each URL value SHALL be encoded as a text string defined in Uniform Resource Identifier (URI): Generic Syntax [RFC 3986 Section 4.1].

Examples:

```
a) "http://www.opengeospatial.org"
b) "/file.txt"
```

7.1.2. Requirements Class: ISO Metadata Type

The ISO Metadata Type requirements class defines the requirements for JSON encoding of ISO metadata types.

REQUIREMENTS CLASS	
/req/base/isometadatatype	
Dependency	JSON
Dependency	GeoJSON
Requirement 5	/req/base/isometadatatype/band
Requirement 6	/req/base/isometadatatype/extent
Requirement 7	/req/base/isometadatatype/citation
Requirement 8	/req/base/isometadatatype/scope
Requirement 9	/req/base/isometadatatype/referencesystem
Requirement 10	/req/base/isometadatatype/lineage
Requirement 11	/req/base/isometadatatype/resolution

The MD_Band is encoded as a JSON object.

Requirement 5	/req/base/isometadatatype/band Each MD_Band value SHALL be encoded as a JSON object matching the XML Schema type as defined in: https://schemas.isotc211.org/19115/-1/mrc/1.3.0/content.xsd

Examples:

```
a) {
     "name": [
       {
         "code": "red"
       }
     ]
   }
b) {
     "name": [
       {
         "code": "B4"
       }
     ]
   }
c) {
     "boundMax": 690,
     "boundMin": 630,
     "boundUnits": "nm"
```

```
}
```

The EX_Extent is encoded as a GeoJSON bounding box or a JSON Object.

Requirement 6	/req/base/isometadatatype/extent Each EX_Extent value SHALL be encoded using one of the following two approaches: 1. as a GeoJSON bounding box encoding as defined in The GeoJSON Format [RFC 7946 Section 5], which SHALL conform to WGS 84 (EPSG:4326) as defined in [RFC 7946 Section 4]; 2. as a JSON object matching the XML Schema type as defined in: https://schemas.isotc211.org/19115/-1/gex/1.3.0/extent.xsd, in which metadata-level CRS declarations are recommended to be provided.

Examples:

```
a) [
     120.0,
     30.0,
     130.0,
     40.0
   ]
b) [
     120.0,
     30.0,
     10.0,
     130.0,
     40.0,
     20.0
   ]
c) {
     "geographicElement": [
       {
         "westBoundLongitude": -171.76409,
         "eastBoundLongitude": -157.86768,
         "southBoundLatitude": -14.42443,
         "northBoundLatitude": 21.31573
       }
     ],
     "crs": "EPSG:4326"
   }
d) {
     "geographicElement": [
       {
         "polygon": [
           {
             "exterior": {
               "LinearRing": {
                 "posList": [
                   51.556272,
                   -0.2803943,
                   51.5562758,
                   -0.2787397,
                   51.5556539,
                   -0.278736,
                   51.5556501,
                   -0.2803906,
                   51.556272,
                   -0.2803943
                 ]
               }
```

```
            }
          }
        ]
      }
    ],
    "crs": "EPSG:4326"
  }
```

The CI_Citation is encoded as a JSON object.

Requirement 7	/req/base/isometadatatype/citation Each CI_Citation value SHALL be encoded as a JSON object matching the XML Schema type as defined in: https://schemas.isotc211.org/19115/-1/cit/1.3.0/citation.xsd

Example:

```
{
  "title": "Open Geospatial Consortium",
  "alternateTitle": [
    "OGC"
  ],
  "identifier": [
    {
      "code": "https://portal.ogc.org/files/?artifact_id=104605&version=1"
    }
  ]
}
```

The MD_Scope is encoded as a JSON object.

Requirement 8	/req/base/isometadatatype/scope Each MD_Scope value SHALL be encoded as a JSON object matching the XML Schema type as defined in: https://schemas.isotc211.org/19115/-1/mcc/1.3.0/commonClasses.xsd

Examples:

```
a) {
     "level": "dataset",
     "levelDescription": [
       {
         "dataset": "whu_rs19"
       }
     ]
   }
b) {
     "level": "attribute",
     "levelDescription": [
       {
         "attributes": "completeness"
       }
     ]
   }
c) {
     "level": "coverage",
     "extent": [
```

```
      {
        "geographicElement": [
          {
            "westBoundLongitude": -171.76409,
            "eastBoundLongitude": -157.86768,
            "southBoundLatitude": -14.42443,
            "northBoundLatitude": 21.31573
          }
        ]
      }
    ]
  }
d) {
    "level": "feature",
    "extent": [
      {
        "geographicElement": [
          {
            "polygon": [
              {
                "exterior": {
                  "LinearRing": {
                    "posList": [
                      51.556272,
                      -0.2803943,
                      51.5562758,
                      -0.2787397,
                      51.5556539,
                      -0.278736,
                      51.5556501,
                      -0.2803906,
                      51.556272,
                      -0.2803943
                    ]
                  }
                }
              }
            ]
          }
        ]
      }
    ]
  }
```

The MD_ReferenceSystem is encoded as a JSON object.

Requirement 9	/req/base/isometadatatype/referencesystem Each MD_ReferenceSystem value SHALL be encoded as a JSON object matching the XML Schema type as defined in: https://standards.iso.org/iso/19115/-3/mrs/1.0/referenceSystem.xsd

Examples:

```
a) {
    "referenceSystem": {
      "referenceSystemIdentifier": "EPSG:3857"
    }
  }
b) {
    "referenceSystem": {
```

```
      "referenceSystemIdentifier": "EPSG:4326"
    }
  }
```

The LI_Lineage is encoded as a JSON object.

Requirement 10	/req/base/isometadatatype/lineage Each LI_Lineage value SHALL be encoded as a JSON object matching the XML Schema type as defined in: https://schemas.isotc211.org/19115/-1/mrl/1.3.0/lineage.xsd

Examples:

```
{
  "LI_Lineage": {
    "statement": "The dataset was generated from high-resolution satellite
imagery through a standardized preprocessing and manual labeling workflow for AI
model training.",
    "source": [
      {
        "LI_Source": {
          "description": "High-resolution multispectral satellite imagery
acquired in 2023 covering urban areas.",
          "sourceCitation": {
            "CI_Citation": {
              "title": "Urban Satellite Imagery Dataset 2023"
            }
          }
        }
      }
    ],
    "processStep": [
      {
        "LI_ProcessStep": {
          "description": "Radiometric calibration and geometric correction were
applied to the raw satellite images.",
          "rationale": "To ensure spatial alignment and consistent reflectance
values for subsequent labeling.",
          "stepDateTime": {
            "TimeInstant": "2023-06-15T10:00:00Z"
          }
        }
      },
      {
        "LI_ProcessStep": {
          "description": "Trained annotators manually labeled buildings and
roads using a GIS-based annotation platform.",
          "rationale": "To generate high-quality ground truth data for
supervised deep learning models.",
          "stepDateTime": {
            "TimePeriod": {
              "begin": "2023-06-20",
              "end": "2023-07-10"
            }
          }
        }
      }
    ]
  }
```

}

The MD_Resolution is encoded as a JSON object.

Requirement 11	/req/base/isometadatatype/resolution Each MD_Resolution value SHALL be encoded as a JSON object matching the XML Schema type as defined in: https://standards.iso.org/iso/19115/-3/mri/1.0/identification.xsd

Examples:

```
a) {
     "distance": {
       "value": 0.5,
       "uom": "m"
     }
   }
b) {
     "equivalentScale": {
       "MD_RepresentativeFraction": {
         "denominator": 10000
       }
     }
   }
c) {
     "vertical": {
       "value": 1,
       "uom": "m"
     }
   }
d) {
     "angularDistance": {
       "value": 0.01,
       "uom": "degree"
     }
   }
e) {
     "levelOfDetail": "High resolution imagery suitable for urban analysis"
   }
```

7.1.3. Requirements Class: ISO Quality Type

The ISO Quality Type requirements class defines the requirements for JSON encoding of ISO quality types.

REQUIREMENTS CLASS	
/req/base/isoqualitytype	
Dependency	JSON
Requirement 12	/req/base/isoqualitytype/element

The QualityElement object is encoded as a JSON object with properties shown in Table 1.

Requirement 12	/req/base/isoqualitytype/element Each QualityElement value SHALL be encoded as a JSON object with properties shown in Table 1.

Table 1 — QualityElement properties

JSON PROPERTY	DEFINITION	DATA TYPE AND VALUES	OBLIGATION
type	The type of the quality element object.	CharacterString [1..1]	Mandatory
measure	Reference to measure used.	MeasureReference [1..1]	Mandatory
evaluationMethod	Evaluation information.	EvaluationMethod [1..*]	Mandatory
result	Value obtained from applying a data quality measure.	QualityResult [1..*]	Mandatory

Example:

```
{
  "type": "FormatConsistency",
  "measure": {
    "measureDescription": "Percentage of training samples with inconsistent
image format"
  },
  "evaluationMethod": [
    {
      "evaluationMethodDescription": "Full test method to calculate the
percentage of training samples with an inconsistent format"
    }
  ],
  "result": [
    {
      "quantitativeResult": {
        "value": [
          0
        ],
        "valueUnit": "%"
      }
    }
  ]
}
```

7.1.4. Requirements Class: Geospatial Type

The Geospatial Type requirements class defines the requirements for JSON encoding of geospatial types.

REQUIREMENTS CLASS	
/req/base/geospatialtype	
Dependency	JSON
Dependency	GeoJSON
Requirement 13	/req/base/geospatialtype/feature

The encoding of one or more features follows the GeoJSON RFC rules for encoding a Feature object, with members “type”, “geometry” and “properties”. A Feature object represents a spatially bounded thing. Every Feature object is a GeoJSON object no matter where it occurs in a GeoJSON text. [RFC 7946]

Requirement 13	/req/base/geospatialtype/feature Each Feature value SHALL be encoded using the GeoJSON feature encoding defined in the GeoJSON Format [RFC 7946 Section 3.2].

Examples of Feature encodings are:

```
a) {
     "type": "Feature",
     "geometry": {
       "type": "Point",
       "coordinates": [
         120.0,
         30.0
       ]
     },
     "properties": {
       "class": "station"
     }
   }
b) {
     "type": "Feature",
     "geometry": {
       "type": "LineString",
       "coordinates": [
         [
           120.0,
           30.0
         ],
         [
           130.0,
           40.0
         ]
       ]
     },
     "properties": {
       "class": "road"
     }
   }
```

```
c) {
     "type": "Feature",
     "geometry": {
       "type": "Polygon",
       "coordinates": [
         [
           [
             120.0,
             30.0
           ],
           [
             130.0,
             30.0
           ],
           [
             125.0,
             40.0
           ],
           [
             120.0,
             30.0
           ]
         ]
       ]
     },
     "properties": {
       "class": "building"
     }
   }
```

7.2. Requirements Class: AI_TrainingDataset

The AI_TrainingDataset requirements class defines a JSON encoding for the AI_TrainingDataset module, which is based on the UML model specified in the TrainingDML-AI Part 1: Conceptual Model Standard.

REQUIREMENTS CLASS	
/req/aitrainingdataset	
Dependency	JSON
Dependency	/req/base/jsonbasetype
Dependency	/req/base/isometadatatype
Dependency	/req/aitrainingdata

REQUIREMENTS CLASS	
Dependency	/req/aitask
Dependency	/req/ailabeling
Dependency	/req/aidataquality
Dependency	/req/aitdchangeset
Requirement 14	/req/aitrainingdataset/trainingdataset
Requirement 15	/req/aitrainingdataset/metricsinliterature
Requirement 16	/req/aitrainingdataset/eotrainingdataset

The AI_TrainingDataset object is encoded as a JSON object with properties shown in Table 2.

Requirement 14	/req/aitrainingdataset/trainingdataset Each AI_TrainingDataset object SHALL implement the Mandatory properties shown in Table 2.

Table 2 — AI_TrainingDataset properties

JSON PROPERTY	DEFINITION	DATA TYPE AND VALUES	OBLIGATION
type	Type of the training dataset.	"AI_AbstractTraining Dataset"	Mandatory
id	Identification of the AI training dataset.	CharacterString [1..1]	Mandatory
doi	Digital object identifier of the AI training dataset.	CharacterString [0..1]	Optional
scope	Description of the scope of the training dataset.	MD_Scope [0..1]	Optional
name	Name of the AI training dataset.	CharacterString [1..1]	Mandatory
description	Description of the AI training dataset.	CharacterString [1..1]	Mandatory
version	Version number of the AI training dataset.	CharacterString [0..1]	Optional

JSON PROPERTY	DEFINITION	DATA TYPE AND VALUES	OBLIGATION
amountOfTrainingData	Total number of training samples in the AI training dataset.	Int [0..1]	Optional
createdTime	Time when the AI training dataset was created.	DateTime [0..1]	Optional
updatedTime	Time when the AI training dataset was updated.	DateTime [0..1]	Optional
license	License description of the AI training dataset.	CharacterString [1..1]	Mandatory
providers	People or organizations who provide the AI training dataset.	CharacterString [0..*]	Optional
keywords	Keywords of the AI training dataset.	CharacterString [0..*]	Optional
metricsInLIT	Results of performance metrics achieved by AI/ML algorithms in the peer-reviewed literature.	AI_MetricsInLiterature [0..*]	Optional
statisticsInfo	Statistical results for training samples in each class.	NamedValue [0..*]	Optional
dataSources	Citation for the data sources.	CI_Citation [0..*]	Optional
numberOfClasses	Total number of classes in the AI training dataset.	Int [0..1]	Optional
classificationScheme	Classification scheme for classes used in the AI training dataset.	CharacterString [0..1]	Optional
classes	Classes used in the AI training dataset.	NamedValue [0..*]	Optional
extent	Spatial extent of the training dataset.	EX_Extent [0..1]	Optional
referenceSystem	Reference system of the training dataset.	MD_ReferenceSystem [0..1]	Optional
lineage	Information about the provenance, source(s), and/or the production process (es) applied to the training dataset.	LI_Lineage [0..*]	Optional

JSON PROPERTY	DEFINITION	DATA TYPE AND VALUES	OBLIGATION
tasks	Task description of the training dataset.	AI_Task [1..*]	Mandatory
labeling	Provenance information of how the training dataset is labeled.	AI_Labeling [0..*]	Optional
quality	Quality information of the training dataset.	DataQuality [0..*]	Optional
changesets	Changeset between two versions of the training dataset.	AI_TDChangeset [0..*]	Optional
data	Training data in the training dataset.	AI_AbstractTrainingData [1..*]	Mandatory

Example:


```
{
  "type": "AI_AbstractTrainingDataset",
  "id": "whu_rs19",
  "name": "WHU-RS19",
  "description": "Wuhan University-Remote Sensing 19 Categories (WHU-RS19) has
19 classes of remote sensing images scenes obtained from Google Earth",
  "license": "CC BY-SA 4.0",
  "amountOfTrainingData": 1013,
  "createdTime": "2010-01-01",
  "providers": [
    "Wuhan University"
  ],
  "keywords": [
    "Remote Sensing",
    "Scene Classification"
  ],
  "numberOfClasses": 19,
  "classes": [
    {
      "key": "Airport",
      "value": null
    },
    {
      "key": "Beach",
      "value": null
    },
    {
      "key": "Bridge",
      "value": null
    },
    {
      "key": "Commercial",
      "value": null
    },
    {
      "key": "Desert",
      "value": null
    },
    {
```

```
        "key": "Farmland",
        "value": null
      },
      {
        "key": "footballField",
        "value": null
      },
      {
        "key": "Forest",
        "value": null
      },
      {
        "key": "Industrial",
        "value": null
      },
      {
        "key": "Meadow",
        "value": null
      },
      {
        "key": "Mountain",
        "value": null
      },
      {
        "key": "Park",
        "value": null
      },
      {
        "key": "Parking",
        "value": null
      },
      {
        "key": "Pond",
        "value": null
      },
      {
        "key": "Port",
        "value": null
      },
      {
        "key": "railwayStation",
        "value": null
      },
      {
        "key": "Residential",
        "value": null
      },
      {
        "key": "River",
        "value": null
      },
      {
        "key": "Viaduct",
        "value": null
      }
    ],
    "extent": [
      -180,
      -90,
      180,
      90
    ],
    "referenceSystem": {
```

```
    "referenceSystemIdentifier": "EPSG:4326"
  },
  "tasks": [
    {
      "type": "AI_EOTask",
      "id": "whu_rs19-task",
      "description": "Structural high-resolution satellite image indexing",
      "taskType": "Scene Classification"
    }
  ],
  "data": [
    {
      "type": "AI_EOTrainingData",
      "id": "airport_01",
      "dataSources": [
        {
          "title": "googleEarth"
        }
      ],
      "dataURL": [
        "image/Airport/airport_01.jpg"
      ],
      "labels": [
        {
          "type": "AI_SceneLabel",
          "class": "Airport"
        }
      ]
    },
    …
  ]
}
```


If the optional element AI_MetricsInLiterature is specified, this element is encoded as JSON object with properties as shown in Table 3.

Requirement 15	/req/aitrainingdataset/metricsinliterature Each AI_MetricsInLiterature value SHALL implement the Mandatory properties shown in Table 3.

Table 3 — AI_MetricsInLiterature properties

JSON PROPERTY	DEFINITION	DATA TYPE AND VALUES	OBLIGATION
doi	Digital object identifier of the peer-reviewed literature.	CharacterString [1..1]	Mandatory
algorithm	AI/ML algorithms used in the peer-reviewed literature.	CharacterString [0..1]	Optional
metrics	Metrics and results of AI/ML algorithms in the peer-reviewed literature.	NamedValue [1..*]	Mandatory

Example:


```
{
  "doi": "10.1109/TGRS.2019.2917161",
  "algorithm": "FACNN",
  "metrics": [
    {
      "key": "Overall Accuracy",
      "value": 0.9881
    }
  ]
}
```


The AI_EOTrainingDataset object is encoded as a JSON object with properties shown in Table 2 and Table 4.

Requirement 16	/req/aitrainingdataset/eotrainingdataset Each AI_EOTrainingDataset object SHALL implement the Mandatory properties both shown in Table 2 and Table 4.

Table 4 — AI_EOTrainingDataset properties

JSON PROPERTY	DEFINITION	DATA TYPE AND VALUES	OBLIGATION
type	Type of the training dataset.	"AI_EOTrainingDataset"	Mandatory
bands	Description of the image bands used in the EO training dataset.	MD_Band [0..*]	Optional
imageSize	Size of the images used in the EO training dataset.	CharacterString [0..1]	Optional

Example:


```
{
  "type": "AI_EOTrainingDataset",
  "id": "whu_rs19",
  "name": "WHU-RS19",
  "description": "Wuhan University-Remote Sensing 19 Categories (WHU-RS19) has
19 classes of remote sensing images scenes obtained from Google Earth",
  "license": "CC BY-SA 4.0",
  "amountOfTrainingData": 1013,
  "createdTime": "2010-01-01",
  "providers": [
    "Wuhan University"
  ],
  "keywords": [
    "Remote Sensing",
    "Scene Classification"
  ],
  "numberOfClasses": 19,
  "extent": [
    -180,
    -90,
    180,
```

```
    90
  ],
  "referenceSystem": {
    "referenceSystemIdentifier": "EPSG:4326"
  },
  "bands": [
    {
      "name": [
        {
          "code": "red"
        }
      ]
    },
    {
      "name": [
        {
          "code": "green"
        }
      ]
    },
    {
      "name": [
        {
          "code": "blue"
        }
      ]
    }
  ],
  "imageSize": "6000x7600",
  "classes": [
    {
      "key": "Airport",
      "value": null
    },
    {
      "key": "Beach",
      "value": null
    },
    {
      "key": "Bridge",
      "value": null
    },
    {
      "key": "Commercial",
      "value": null
    },
    {
      "key": "Desert",
      "value": null
    },
    {
      "key": "Farmland",
      "value": null
    },
    {
      "key": "footballField",
      "value": null
    },
    {
      "key": "Forest",
      "value": null
    },
    {
```

```
      "key": "Industrial",
      "value": null
    },
    {
      "key": "Meadow",
      "value": null
    },
    {
      "key": "Mountain",
      "value": null
    },
    {
      "key": "Park",
      "value": null
    },
    {
      "key": "Parking",
      "value": null
    },
    {
      "key": "Pond",
      "value": null
    },
    {
      "key": "Port",
      "value": null
    },
    {
      "key": "railwayStation",
      "value": null
    },
    {
      "key": "Residential",
      "value": null
    },
    {
      "key": "River",
      "value": null
    },
    {
      "key": "Viaduct",
      "value": null
    }
  ],
  "tasks": [
    {
      "type": "AI_EOTask",
      "id": "whu_rs19-task",
      "description": "Structural high-resolution satellite image indexing",
      "taskType": "Scene Classification"
    }
  ],
  "data": [
    {
      "type": "AI_EOTrainingData",
      "id": "airport_01",
      "dataSources": [
        {
          "title": "googleEarth"
        }
      ],
      "dataURL": [
        "image/Airport/airport_01.jpg"
```

```
      ],
      "labels": [
        {
          "type": "AI_SceneLabel",
          "class": "Airport"
        }
      ]
    },
    …
  ]
}
```

7.3. Requirements Class: AI_TrainingData

The AI_TrainingData requirements class defines a JSON encoding for the AI_TrainingData module, which is based on the UML model specified in the TrainingDML-AI Part 1: Conceptual Model Standard.

REQUIREMENTS CLASS	
/req/aitrainingdata	
Dependency	JSON
Dependency	/req/base/jsonbasetype
Dependency	/req/base/isometadatatype
Dependency	/req/ailabel
Dependency	/req/ailabeling
Dependency	/req/aidataquality
Requirement 17	/req/aitrainingdata/trainingdata
Requirement 18	/req/aitrainingdata/trainingtypecode
Requirement 19	/req/aitrainingdata/eotrainingdata

The AI_TrainingData object is encoded as a JSON object with properties shown in Table 5.

Requirement 17	/req/aitrainingdata/trainingdata Each AI_TrainingData object SHALL implement the Mandatory properties shown in Table 5.

Table 5 — AI_TrainingData properties

JSON PROPERTY	DEFINITION	DATA TYPE AND VALUES	OBLIGATION
type	Type of the training sample.	"AI_AbstractTrainingData"	Mandatory
id	Identification of an individual AI training sample.	CharacterString [1..1]	Mandatory
datasetId	Identification of the training dataset that the training sample belongs to.	CharacterString [0..1]	Optional
trainingType	Training type of the individual AI training sample.	AI_TrainingTypeCode [0..1]	Optional
numberOfLabels	Total number of labels in the individual AI training sample.	Int [0..1]	Optional
dataSources	Citation of inputs to prepare a training sample.	CI_Citation [0..*]	Optional
extent	Spatial extent of the individual training sample.	EX_Extent [0..1]	Optional
dataTime	Date and time at which the geospatial data were obtained.	DateTime [0..*]	Optional
lineage	Information about the provenance, source(s), and/or the production process (es) applied to the training sample.	LI_Lineage [0..*]	Optional
labels	Labels in the training data.	AI_Label [0..*]	Optional
labeling	Provenance information of how the training data is labeled.	AI_Labeling [0..*]	Optional
quality	Quality information of the training data.	DataQuality [0..*]	Optional

Example:

```
{
  "type": "AI_AbstractTrainingData",
```

```
  "id": "airport_01",
  "dataSources": [
    {
      "title": "googleEarth"
    }
  ],
  "extent": {
    "geographicElement": [
      {
        "westBoundLongitude": -171.76409,
        "eastBoundLongitude": -171.56578,
        "southBoundLatitude": -14.42443,
        "northBoundLatitude": -14.32568
      }
    ]
  },
  "dataTime": [
    "2002-05-30T09:30:10Z"
  ],
  "labels": [
    {
      "type": "AI_SceneLabel",
      "class": "Airport"
    }
  ]
}
```

The AI_TrainingTypeCode is encoded as a text string whose value is one of “training”, “validation”, “test” or “retraining”.

Requirement 18	/req/aitrainingdata/trainingtypecode Each AI_TrainingTypeCode value SHALL be a text string whose value is one of “training”, “validation”, “test” or “retraining”.

Examples:

```
a) "training"
b) "validation"
c) "test"
d) "retraining"
```

The AI_EOTrainingData object is encoded as a JSON object with properties both shown in Table 5 and Table 6.

Requirement 19	/req/aitrainingdata/eotrainingdata Each AI_EOTrainingData object SHALL implement the Mandatory properties as defined in Table 5 and Table 6.

Table 6 — AI_EOTrainingData properties

JSON PROPERTY	DEFINITION	DATA TYPE AND VALUES	OBLIGATION
spatialResolution	Spatial resolution of the individual EO training sample.	MD_Resolution [0..1]	Optional
dataURL	URLs of the EO data, including both relative and absolute paths, which can encompass local paths, network addresses, and more.	URI [1..*]	Mandatory

Example:

```
{
  "type": "AI_EOTrainingData",
  "id": "airport_01",
  "dataSources": [
    {
      "title": "googleEarth"
    }
  ],
  "extent": {
    "geographicElement": [
      {
        "westBoundLongitude": -171.76409,
        "eastBoundLongitude": -171.56578,
        "southBoundLatitude": -14.42443,
        "northBoundLatitude": -14.32568
      }
    ]
  },
  "dataTime": [
    "2002-05-30T09:30:10Z"
  ],
  "dataURL": [
    "image/Airport/airport_01.jpg"
  ],
  "spatialResolution": {
    "distance": {
      "value": "0.5",
      "uom": "m"
    }
  },
  "labels": [
    {
      "type": "AI_SceneLabel",
      "class": "Airport"
    }
  ]
}
```

7.4. Requirements Class: AI_Task

The AI_Task requirements class defines a JSON encoding for the AI_Task module, which is based on the UML model specified in the TrainingDML-AI Part 1: Conceptual Model Standard.

REQUIREMENTS CLASS	
/req/aitask	
Dependency	JSON
Dependency	/req/base/jsonbasetype
Requirement 20	/req/aitask/task
Requirement 21	/req/aitask/eotask

The AI_Task object is encoded as a JSON object with properties as shown in Table 7.

Requirement 20	/req/aitask/task Each AI_Task object SHALL implement the Mandatory properties shown in Table 7.

Table 7 — AI_Task properties

JSON PROPERTY	DEFINITION	DATA TYPE AND VALUES	OBLIGATION
type	Type of the task object.	"AI_AbstractTask"	Mandatory
id	Identification of the task.	CharacterString [1..1]	Mandatory
datasetId	Identification of the training dataset the training sample belongs to.	CharacterString [0..1]	Optional
description	Description of the AI task.	CharacterString [0..1]	Optional

Example:

```
{
  "type": "AI_AbstractTask",
  "id": "image-indexing-task",
  "description": "Structural high-resolution satellite image indexing"
}
```

The AI_EOTask object is encoded as a JSON object with properties both shown in Table 7 and Table 8.

	/req/aitask/eotask
Requirement 21	Each AI_EOTask object SHALL implement the Mandatory properties shown in Table 7 and Table 8.

Table 8 — AI_EOTask properties

JSON PROPERTY	DEFINITION	DATA TYPE AND VALUES	OBLIGATION
type	Type of the task object.	"AI_EOTask"	Mandatory
taskType	Type of the EO task.	CharacterString [1..1]	Mandatory

Example:

```
{
  "type": "AI_EOTask",
  "id": "image-indexing-task",
  "description": "Structural high-resolution satellite image indexing",
  "taskType": "Scene Classification"
}
```

7.5. Requirements Class: AI_Label

The AI_Label requirements class defines a JSON encoding for the AI_Label module, which is based on the UML model specified in the TrainingDML-AI Part 1: Conceptual Model Standard.

REQUIREMENTS CLASS	
/req/ailabel	
Dependency	JSON
Dependency	/req/base/jsonbasetype
Dependency	/req/base/geospatialtype
Requirement 22	/req/ailabel/label

REQUIREMENTS CLASS	
Requirement 23	/req/ailabel/scenelabel
Requirement 24	/req/ailabel/objectlabel
Requirement 25	/req/ailabel/pixellabel
Requirement 26	/req/ailabel/imageformatcode

The AI_Label object is encoded as a JSON object with properties as shown in Table 9.

Requirement 22	/req/ailabel/label Each AI_Label object SHALL implement the Mandatory properties shown in Table 9.

Table 9 — AI_Label properties

JSON PROPERTY	DEFINITION	DATA TYPE AND VALUES	OBLIGATION
type	Type of the label object.	"AI_AbstractLabel"	Mandatory
isNegative	Whether the training sample related to the label is a positive or negative sample.	Bool [0..1] Default: false	Optional
confidence	Confidence score of the labeler.	Float [0..1] Default: 1.0 Range: [0, 1]	Optional

Example:

```
{
  "type": "AI_AbstractLabel",
  "isNegative": false,
  "confidence": 1.0
}
```

The AI_SceneLabel object is encoded as a JSON object with properties as shown in Table 10.

Requirement 23	/req/ailabel/scenelabel Each AI_SceneLabel object SHALL implement the properties shown in Table 10.

Table 10 — AI_SceneLabel properties

JSON PROPERTY	DEFINITION	DATA TYPE AND VALUES	OBLIGATION
type	Type of the label object at the scene level.	"AI_SceneLabel"	Mandatory
class	Class that records the semantic of the scene of the training sample.	CharacterString [1..1]	Mandatory

Example:

```
{
  "type": "AI_SceneLabel",
  "class": "Airport"
}
```

The AI_ObjectLabel object is encoded as a JSON object with properties shown in Table 11.

Requirement 24	/req/ailabel/objectlabel Each AI_ObjectLabel object SHALL implement the Mandatory properties shown in Table 11.

Table 11 — AI_ObjectLabel properties

JSON PROPERTY	DEFINITION	DATA TYPE AND VALUES	OBLIGATION
type	Type of the label object at the object level.	"AI_ObjectLabel"	Mandatory
object	Feature that represents the position and attributes of the object.	Feature [1..1]	Mandatory
bboxType	Type of the bbox.	CharacterString [0..1]	Optional
class	Class that records the semantic of the object type.	CharacterString [1..1]	Mandatory
dateTime	Created time of the object label.	DateTime [0..1]	Optional

Example:

```
{
  "type": "AI_ObjectLabel",
  "class": "Truck",
  "object": {
    "type": "Feature",
    "properties": {
      "truncated": 0.0,
      "occluded": 0,
      "alpha": -1.57
    },
```

```
    "geometry": {
      "type": "Polygon",
      "coordinates": [
        [
          [
            2257.0,
            332.0
          ],
          [
            2271.0,
            332.0
          ],
          [
            2271.0,
            350.0
          ],
          [
            2257.0,
            350.0
          ],
          [
            2257.0,
            332.0
          ],
          [
            2257.0,
            332.0
          ]
        ]
      ]
    }
  },
  "bboxType": "Horizontal BBox"
}
```

The AI_PixelLabel object is encoded as a JSON object with properties as shown in Table 12.

Requirement 25	/req/ailabel/pixellabel Each AI_PixelLabel object shall implement the Mandatory properties shown in Table 12.

Table 12 — AI_PixelLabel properties

JSON PROPERTY	DEFINITION	DATA TYPE AND VALUES	OBLIGATION
type	Type of the label object at the pixel level.	"AI_PixelLabel"	Mandatory
imageURL	URL of the images representing the label information.	URI [1..*]	Mandatory
imageFormat	Image data format.	AI_ImageFormatCode [1..*]	Mandatory

Example:

```
{
  "type": "AI_PixelLabel",
  "imageURL": [
    "/label_5classes/GF2_PMS1__L1A0000647767-MSS1_label.tif"
  ],
```

```
  "imageFormat": [
    "image/tiff; application=geotiff"
  ]
}
```

The AI_ImageFormatCode is encoded as a text string whose value is defined in Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types RFC 2046.

Requirement 26	/req/ailabel/imageformatcode Each AI_ImageFormatCode value SHALL be encoded as a text string defined in Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types [RFC 2046].

Examples:

```
a) "image/tiff; application=geotiff"
b) "application/x-netcdf"
c) "image/png"
d) "image/jp2"
```

7.6. Requirements Class: AI_Labeling

The AI_Labeling requirements class defines a JSON encoding for the AI_Labeling module, which is based on the UML model specified in the TrainingDML-AI Part 1: Conceptual Model Standard.

REQUIREMENTS CLASS	
/req/ailabeling	
Dependency	JSON
Dependency	/req/base/jsonbasetype
Dependency	/req/base/isometadatatype
Requirement 27	/req/ailabeling/labeling
Requirement 28	/req/ailabeling/labeler
Requirement 29	/req/ailabeling/labelingprocedure
Requirement 30	/req/ailabeling/labelingmethodcode

The AI_Labeling object is encoded as a JSON object with properties shown in Table 13.

Requirement 27	/req/ailabeling/labeling Each AI_Labeling object SHALL implement the Mandatory properties shown in Table 13.

Table 13 — AI_Labeling properties

JSON PROPERTY	DEFINITION	DATA TYPE AND VALUES	OBLIGATION
type	Type of the labeling object.	“AI_Labeling”	Mandatory
id	Identifier of the labeling.	CharacterString [1..1]	Mandatory
scope	Description of the scope of the labeling.	MD_Scope [1..1]	Mandatory
labelingTime	Date and time at which the labeling procedure was carried out.	DateTime [0..1]	Optional
labelers	Labelers of the labeling activity.	AI_Labeler [0..*]	Optional
procedure	Procedure used in the labeling activity.	AI_LabelingProcedure [0..1]	Optional

Example:

```
{
  "type": "AI_Labeling",
  "id": "0",
  "scope": {
    "level": "dataset",
    "levelDescription": [
      {
        "dataset": "whu_rs19"
      }
    ]
  },
  "labelingTime": "2015-04-20T07:04:15Z",
  "labelers": [
    {
      ..
    }
  ],
  "procedure": {
    ..
  }
}
```

The AI_Labeler object is encoded as a JSON object with properties as shown in Table 14.

Requirement 28	/req/ailabeling/labeler Each AI_Labeler object SHALL implement the Mandatory properties shown in Table 14.

Table 14 — AI_Labeler properties

JSON PROPERTY	DEFINITION	DATA TYPE AND VALUES	OBLIGATION
type	Type of the labeler object.	“AI_Labeler”	Mandatory
id	Identifier of the labeler.	CharacterString [1..1]	Mandatory
name	Name of the labeler.	CharacterString [1..1]	Mandatory

Example:

```
{
  "type": "AI_Labeler",
  "id": "0",
  "name": "Tom"
}
```

An AI_LabelingProcedure object is encoded as a JSON object with properties as shown in Table 15.

Requirement 29	/req/ailabeling/labelingprocedure Each AI_LabelingProcedure object SHALL implement the Mandatory properties shown in Table 15.

Table 15 — AI_LabelingProcedure properties

JSON PROPERTY	DEFINITION	DATA TYPE AND VALUES	OBLIGATION
type	Type of the labeling procedure object.	“AI_LabelingProcedure”	Mandatory
id	Identifier of the labeling procedure.	CharacterString [1..1]	Mandatory
methods	Methods used in the labeling procedure.	AI_LabelingMethodCode [1..*]	Mandatory
tools	Tools or software used in the labeling procedure.	CharacterString [0..*]	Optional

Example:

```
{
  "type": "AI_LabelingProcedure",
  "id": "0",
  "methods": [
    "manual"
  ],
  "tools": [
    "ArcGIS"
```

```
  ]
}
```

The AI_LabelingMethodCode is encoded as a text string whose value is one of “manual”, “semi-automatic”, “automatic” or “unknown”.

Requirement 30	/req/ailabeling/labelingmethodcode Each AI_LabelingMethodCode value SHALL be a text string whose value is one of “manual”, “semi-automatic”, “automatic” or “unknown”.

Examples:

```
a) "manual"
b) "semi-automatic"
c) "automatic"
d) "unknown"
```

7.7. Requirements Class: AI_DataQuality

The AI_DataQuality requirements class defines a JSON encoding for the AI_DataQuality module, which is based on the UML model specified in the TrainingDML-AI Part 1: Conceptual Model Standard.

REQUIREMENTS CLASS	
/req/aidataquality	
Dependency	JSON
Dependency	/req/base/jsonbasetype
Dependency	/req/base/isometadatatype
Dependency	/req/base/isoqualitytype
Requirement 31	/req/aidataquality/classbalancedegree

The AI_ClassBalanceDegree object is encoded as a JSON object with properties as shown in Table 16.

Requirement 31	/req/aidataquality/classbalancedegree

	Each AI_ClassBalanceDegree object SHALL implement the Mandatory properties as shown in Table 16.

Table 16 — AI_ ClassBalanceDegree properties

JSON PROPERTY	DEFINITION	DATA TYPE AND VALUES	OBLIGATION
type	Type of the class balance degree object.	"AI_ClassBalanceDegree"	Mandatory
measure	Reference to measure used.	MeasureReference [1..1]	Mandatory
evaluationMethod	Evaluation information.	EvaluationMethod [1..*]	Mandatory
result	Value obtained from applying a data quality measure.	QualityResult [1..*]	Mandatory

Example:

```
{
  "type": "AI_ClassBalanceDegree",
  "measure": {
    "measureDescription": "Balance degree of label classes"
  },
  "evaluationMethod": [
    {
      "evaluationMethodDescription": "Counting the number of training samples
belonging to each class and calculating the balance degree"
    }
  ],
  "result": [
    {
      "quantitativeResult": {
        "value": [
          93.5
        ],
        "valueUnit": "%"
      }
    }
  ]
}
```

7.8. Requirements Class: AI_TDChangeset

The AI_TDChangeset requirements class defines a JSON encoding for the AI_TDChangeset module, which is based on the UML model specified in the TrainingDML-AI Part 1: Conceptual Model Standard.

REQUIREMENTS CLASS	
/req/aitdchangeset	
Dependency	JSON
Dependency	/req/base/jsonbasetype
Dependency	/req/aitrainingdata
Requirement 32	/req/aitdchangeset/tdchangeset

The AI_TDChangeset object is encoded as a JSON object with properties shown in Table 17.

Requirement 32	/req/aitdchangeset/tdchangeset Each AI_TDChangeset object SHALL implement the Mandatory properties as shown in Table 17.

Table 17 — AI_ TDChangeset properties

JSON PROPERTY	DEFINITION	DATA TYPE AND VALUES	OBLIGATION
type	Type of the TD changeset object.	"AI_TDChangeset"	Mandatory
id	Identifier of the changeset.	CharacterString [1..1]	Mandatory
datasetId	Identifier of the training dataset the changeset belongs to.	CharacterString [0..1]	Optional
version	Version of the training dataset that the changeset belongs to.	CharacterString [0..1]	Optional
changeCount	Total number of changed training samples.	Int [1..1]	Mandatory
createdTime	The time that the changeset was created.	DateTime [0..1]	Optional
add	Added training samples.	AI_AbstractTrainingData [0..*]	Optional
modify	Modified training samples.	AI_AbstractTrainingData [0..*]	Optional
delete	Deleted training samples.	AI_AbstractTrainingData [0..*]	Optional

Example:


```
{
  "type": "AI_TDChangeset",
  "id": "changeset-dota_v1.5",
  "datasetId": "dota_v1.5",
  "createdTime": "2019-01-01",
  "changeCount": 9,
  "modify": [
    {
      "type": "AI_EOTrainingData",
      "id": "P1228",
      "dataSources": [
        {
          "title": "GF"
        }
      ],
      "dataURL": [
        "train/images/P1228.png"
      ],
      "numberOfLabels": 50,
      "trainingType": "training",
      "labels": [
        {
          "type": "AI_ObjectLabel",
          "class": "ship",
          "object": {
            "type": "Feature",
            "properties": {},
            "geometry": {
              "type": "Polygon",
              "coordinates": [
                [
                  [
                    2306.0,
                    729.0
                  ],
                  [
                    2330.0,
                    729.0
                  ],
                  [
                    2330.0,
                    744.0
                  ],
                  [
                    2306.0,
                    744.0
                  ],
                  [
                    2306.0,
                    729.0
                  ]
                ]
              ]
            }
          },
          "bboxType": "Horizontal BBox"
        },
        …
      ]
    }
  ]
```

```
}
```

A

ANNEX A (NORMATIVE) ABSTRACT TEST SUITE

ANNEX A (NORMATIVE) ABSTRACT TEST SUITE

A.1. Introduction

Conformance is tested using the JSON Schema document which formalize the requirements described above.

A.2. Conformance Class: Base

The Base conformance class tests that occurrences of the basic types are encoded according to the requirements.

A.2.1. Conformance Class: JSON Base Type

CONFORMANCE CLASS A.1: JSON BASE TYPE	
IDENTIFIER	`http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/base/jsonbasetype`
REQUIREMENTS CLASS	/req/base/jsonbasetype
CONFORMANCE TESTS	Abstract test A.1: `/conf/base/jsonbasetype/json` Abstract test A.2: `/conf/base/jsonbasetype/datetime` Abstract test A.3: `/conf/base/jsonbasetype/namedvalue` Abstract test A.4: `/conf/base/jsonbasetype/url`

ABSTRACT TEST A.1	
IDENTIFIER	`/conf/base/jsonbasetype/json`

ABSTRACT TEST A.1

REQUIREMENT	/req/base/jsonbasetype/json
TEST PURPOSE	Verify that the document is well-formed JSON.
TEST METHOD	Load the document in a JSON validator. Pass if no errors reported. Fail otherwise.

ABSTRACT TEST A.2

IDENTIFIER	`/conf/base/jsonbasetype/datetime`
REQUIREMENT	/req/base/jsonbasetype/datetime
TEST PURPOSE	Verify that JSON instance documents claiming conformance to this specification contain valid Date Time values according to Date and Time on the Internet: Timestamps [RFC 3339 Section 5.6].
TEST METHOD	Inspect the instance document to verify the above requirement.

ABSTRACT TEST A.3

IDENTIFIER	`/conf/base/jsonbasetype/namedvalue`
REQUIREMENT	/req/base/jsonbasetype/namedvalue
TEST PURPOSE	Verify that JSON instance documents claiming conformance to this specification validate against the JSON schema specified in http://schemas.opengis.net/trainingdml-ai/part2/1.0/namedValue.json.
TEST METHOD	Validate the instance document against the namedValue.json schema to verify the above requirement. The process may be using an appropriate software tool for validation or be a manual process that checks all definitions from the JSON schema specification.

ABSTRACT TEST A.4

IDENTIFIER	`/conf/base/jsonbasetype/url`
REQUIREMENT	/req/base/jsonbasetype/url
TEST PURPOSE	Verify that JSON instance documents claiming conformance to this specification contain valid URL values according to Uniform Resource Identifier (URI): Generic Syntax [RFC 3986 Section 4.1]. A URL value can be absolute or relative and may have an optional fragment identifier.

ABSTRACT TEST A.4	
TEST METHOD	Inspect the instance document to verify the above requirement.

A.2.2. Conformance Class: ISO Metadata Type

CONFORMANCE CLASS A.2: ISO METADATA TYPE	
IDENTIFIER	`http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/base/isometadatatype`
REQUIREMENTS CLASS	/req/base/isometadatatype
CONFORMANCE TESTS	Abstract test A.5: `/conf/base/isometadatatype/band` Abstract test A.6: `/conf/base/isometadatatype/extent` Abstract test A.7: `/conf/base/isometadatatype/citation` Abstract test A.8: `/conf/base/isometadatatype/scope` Abstract test A.9: `/conf/base/isometadatatype/referencesystem` Abstract test A.10: `/conf/base/isometadatatype/lineage` Abstract test A.11: `/conf/base/isometadatatype/resolution`

ABSTRACT TEST A.5	
IDENTIFIER	`/conf/base/isometadatatype/band`
REQUIREMENT	/req/base/isometadatatype/band
TEST PURPOSE	Verify that instance documents using the MD_Band JSON objects validate against the JSON schema specified in http://schemas.opengis.net/trainingdml-ai/part2/1.0/md_band.json.
TEST METHOD	Validate the instance document against the md_band.json schema to verify the above requirement. The process may be using an appropriate software tool for validation or be a manual process that checks all definitions from the JSON schema specification.

ABSTRACT TEST A.6	
IDENTIFIER	`/conf/base/isometadatatype/extent`
REQUIREMENT	/req/base/isometadatatype/extent
TEST PURPOSE	Verify that instance documents using the EX_Extent JSON objects validate against the JSON schema specified in http://schemas.opengis.net/trainingdml-ai/part2/1.0/ex_extent.json.

ABSTRACT TEST A.6

TEST METHOD	Validate the instance document against the ex_extent.json schema to verify the above requirement. The process may be using an appropriate software tool for validation or be a manual process that checks all definitions from the JSON schema specification.

ABSTRACT TEST A.7

IDENTIFIER	`/conf/base/isometadatatype/citation`
REQUIREMENT	/req/base/isometadatatype/citation
TEST PURPOSE	Verify that instance documents using the CI_Citation JSON objects validate against the JSON schema specified in http://schemas.opengis.net/trainingdml-ai/part2/1.0/ci_citation.json.
TEST METHOD	Validate the instance document against the ci_citation.json schema to verify the above requirement. The process may be using an appropriate software tool for validation or be a manual process that checks all definitions from the JSON schema specification.

ABSTRACT TEST A.8

IDENTIFIER	`/conf/base/isometadatatype/scope`
REQUIREMENT	/req/base/isometadatatype/scope
TEST PURPOSE	Verify that instance documents using the MD_Scope JSON objects validate against the JSON schema specified in http://schemas.opengis.net/trainingdml-ai/part2/1.0/md_scope.json.
TEST METHOD	Validate the instance document against the md_scope.json schema to verify the above requirement. The process may be using an appropriate software tool for validation or be a manual process that checks all definitions from the JSON schema specification.

ABSTRACT TEST A.9

IDENTIFIER	`/conf/base/isometadatatype/referencesystem`
REQUIREMENT	/req/base/isometadatatype/referencesystem
TEST PURPOSE	Verify that instance documents using the MD_ReferenceSystem JSON objects validate against the JSON schema specified in http://schemas.opengis.net/trainingdml-ai/part2/1.0/md_referenceSystem.json.
TEST METHOD	Validate the instance document against the md_referenceSystem.json schema to verify the above requirement. The process may be using an appropriate software tool for validation or be a manual process that checks all definitions from the JSON schema specification.

ABSTRACT TEST A.10	
IDENTIFIER	`/conf/base/isometadatatype/lineage`
REQUIREMENT	/req/base/isometadatatype/lineage
TEST PURPOSE	Verify that instance documents using the LI_Lineage JSON objects validate against the JSON schema specified in http://schemas.opengis.net/trainingdml-ai/part2/1.0/li_lineage.json.
TEST METHOD	Validate the instance document against the li_lineage.json schema to verify the above requirement. The process may be using an appropriate software tool for validation or be a manual process that checks all definitions from the JSON schema specification.

ABSTRACT TEST A.11	
IDENTIFIER	`/conf/base/isometadatatype/resolution`
REQUIREMENT	/req/base/isometadatatype/resolution
TEST PURPOSE	Verify that instance documents using the MD_Resolution JSON objects validate against the JSON schema specified in http://schemas.opengis.net/trainingdml-ai/part2/1.0/md_resolution.json.
TEST METHOD	Validate the instance document against the md_resolution.json schema to verify the above requirement. The process may be using an appropriate software tool for validation or be a manual process that checks all definitions from the JSON schema specification.

A.2.3. Conformance Class: ISO Quality Type

CONFORMANCE CLASS A.3: ISO QUALITY TYPE	
IDENTIFIER	`http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/base/isoqualitytype`
REQUIREMENTS CLASS	/req/base/isoqualitytype
CONFORMANCE TEST	Abstract test A.12: `/conf/base/isoqualitytype/element`

ABSTRACT TEST A.12	
IDENTIFIER	`/conf/base/isoqualitytype/element`

ABSTRACT TEST A.12	
REQUIREMENT	/req/base/isoqualitytype/element
TEST PURPOSE	Verify that instance documents using the QualityElement JSON objects validate against the JSON schema specified in http://schemas.opengis.net/trainingdml-ai/part2/1.0/qualityElement.json.
TEST METHOD	Validate the instance document against the qualityElement.json schema to verify the above requirement. The process may be using an appropriate software tool for validation or be a manual process that checks all definitions from the JSON schema specification.

A.2.4. Conformance Class: Geospatial Type

CONFORMANCE CLASS A.4: GEOSPATIAL TYPE	
IDENTIFIER	`http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/base/geospatialtype`
REQUIREMENTS CLASS	/req/base/geospatialtype
CONFORMANCE TEST	Abstract test A.13: `/conf/base/geospatialtype/feature`

ABSTRACT TEST A.13	
IDENTIFIER	`/conf/base/geospatialtype/feature`
REQUIREMENT	/req/base/geospatialtype/feature
TEST PURPOSE	Verify that instance documents using the Feature JSON objects validate against the JSON schema specified in https://geojson.org/schema/Feature.json.
TEST METHOD	Validate the instance document against the feature.json schema to verify the above requirement. The process may be using an appropriate software tool for validation or be a manual process that checks all definitions from the JSON schema specification.

A.3. Conformance Class: AI_TrainingDataset

The AI_TrainingDataset conformance class tests that the training dataset object is encoded according to the requirements.

CONFORMANCE CLASS A.5: AI_TRAININGDATASET

IDENTIFIER	`http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/aitrainingdataset`
REQUIREMENTS CLASS	/req/aitrainingdataset
PREREQUISITES	Conformance class A.1: `http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/base/jsonbasetype` Conformance class A.2: `http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/base/isometadatatype` Conformance class A.6: `http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/aitrainingdata` Conformance class A.7: `http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/aitask` Conformance class A.9: `http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/ailabeling` Conformance class A.10: `http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/aidataquality` Conformance class A.11: `http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/aitdchangeset`
CONFORMANCE TESTS	Abstract test A.14: `/conf/aitrainingdataset/trainingdataset` Abstract test A.15: `/conf/aitrainingdataset/metricsinliterature` Abstract test A.16: `/conf/aitrainingdataset/eotrainingdataset`

ABSTRACT TEST A.14

IDENTIFIER	`/conf/aitrainingdataset/trainingdataset`
REQUIREMENT	/req/aitrainingdataset/trainingdataset
TEST PURPOSE	Verify that instance documents using the AI_TrainingDataset JSON objects listed in Table 2 validate against the JSON schema specified in http://schemas.opengis.net/trainingdml-ai/part2/1.0/ai_trainingDataset.json.
TEST METHOD	Validate the instance document against the ai_trainingDataset.json schema to verify the above requirement. The process may be using an appropriate software tool for validation or be a manual process that checks all definitions from the JSON schema specification.

ABSTRACT TEST A.15

IDENTIFIER	`/conf/aitrainingdataset/metricsinliterature`
REQUIREMENT	/req/aitrainingdataset/metricsinliterature

ABSTRACT TEST A.15	
TEST PURPOSE	Verify that instance documents using the AI_MetricsInLiterature JSON objects listed in Table 3 validate against the JSON schema specified in http://schemas.opengis.net/trainingdml-ai/part2/1.0/ai_metricsInLiterature.json.
TEST METHOD	Validate the instance document against the ai_metricsInLiterature.json schema to verify the above requirement. The process may be using an appropriate software tool for validation or be a manual process that checks all definitions from the JSON schema specification.

ABSTRACT TEST A.16	
IDENTIFIER	`/conf/aitrainingdataset/eotrainingdataset`
REQUIREMENT	/req/aitrainingdataset/eotrainingdataset
TEST PURPOSE	Verify that instance documents using the AI_EOTrainingDataset JSON objects listed in Table 2 and Table 4 validate against the JSON schema specified in http://schemas.opengis.net/trainingdml-ai/part2/1.0/ai_eoTrainingDataset.json.
TEST METHOD	Validate the instance document against the ai_eoTrainingDataset.json schema to verify the above requirement. The process may be using an appropriate software tool for validation or be a manual process that checks all definitions from the JSON schema specification.

A.4. Conformance Class: AI_TrainingData

The AI_TrainingData conformance class tests that the training data objects are encoded according to the requirements.

CONFORMANCE CLASS A.6: AI_TRAININGDATA	
IDENTIFIER	`http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/aitrainingdata`
REQUIREMENTS CLASS	/req/aitrainingdata
PREREQUISITES	Conformance class A.1: `http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/base/jsonbasetype` Conformance class A.2: `http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/base/isometadatatype` Conformance class A.8: `http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/ailabel`

CONFORMANCE CLASS A.6: AI_TRAININGDATA	
	Conformance class A.9: `http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/ailabeling` Conformance class A.10: `http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/aidataquality`
CONFORMANCE TESTS	Abstract test A.17: `/conf/aitrainingdata/trainingdata` Abstract test A.18: `/conf/aitrainingdata/trainingtypecode` Abstract test A.19: `/conf/aitrainingdata/eotrainingdata`

ABSTRACT TEST A.17	
IDENTIFIER	`/conf/aitrainingdata/trainingdata`
REQUIREMENT	/req/aitrainingdata/trainingdata
TEST PURPOSE	Verify that instance documents using the AI_TrainingData JSON objects listed in Table 5 validate against the JSON schema specified in http://schemas.opengis.net/trainingdml-ai/part2/1.0/ai_trainingData.json.
TEST METHOD	Validate the instance document against the ai_trainingData.json schema to verify the above requirement. The process may be using an appropriate software tool for validation or be a manual process that checks all definitions from the JSON schema specification.

ABSTRACT TEST A.18	
IDENTIFIER	`/conf/aitrainingdata/trainingtypecode`
REQUIREMENT	/req/aitrainingdata/trainingtypecode
TEST PURPOSE	Verify that instance documents using the AI_TrainingTypeCode JSON objects validate against the JSON schema specified in http://schemas.opengis.net/trainingdml-ai/part2/1.0/ai_trainingTypeCode.json.
TEST METHOD	Validate the instance document against the ai_trainingTypeCode.json schema to verify the above requirement. The process may be using an appropriate software tool for validation or be a manual process that checks all definitions from the JSON schema specification.

ABSTRACT TEST A.19	
IDENTIFIER	`/conf/aitrainingdata/eotrainingdata`
REQUIREMENT	/req/aitrainingdata/eotrainingdata

ABSTRACT TEST A.19	
TEST PURPOSE	Verify that instance documents using the AI_EOTrainingData JSON objects listed in Table 5 and Table 6 validate against the JSON schema specified in http://schemas.opengis.net/trainingdml-ai/part2/1.0/ai_eoTrainingData.json.
TEST METHOD	Validate the instance document against the ai_eoTrainingData.json schema to verify the above requirement. The process may be using an appropriate software tool for validation or be a manual process that checks all definitions from the JSON schema specification.

A.5. Conformance Class: AI_Task

The AI_Task conformance class tests that the task objects are encoded according to the requirements.

CONFORMANCE CLASS A.7: AI_TASK	
IDENTIFIER	`http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/aitask`
REQUIREMENTS CLASS	/req/aitask
PREREQUISITE	Conformance class A.1: `http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/base/jsonbasetype`
CONFORMANCE TESTS	Abstract test A.20: `/conf/aitask/task` Abstract test A.21: `/conf/aitask/eotask`

ABSTRACT TEST A.20	
IDENTIFIER	`/conf/aitask/task`
REQUIREMENT	/req/aitask/task
TEST PURPOSE	Verify that instance documents using the AI_Task JSON objects listed in Table 7 validate against the JSON schema specified in http://schemas.opengis.net/trainingdml-ai/part2/1.0/ai_task.json.
TEST METHOD	Validate the instance document against the ai_task.json schema to verify the above requirement. The process may be using an appropriate software tool for validation or be a manual process that checks all definitions from the JSON schema specification.

ABSTRACT TEST A.21	
IDENTIFIER	`/conf/aitask/eotask`
REQUIREMENT	/req/aitask/eotask
TEST PURPOSE	Verify that instance documents using the AI_EOTask JSON objects listed in Table 7 and Table 8 validate against the JSON schema specified in http://schemas.opengis.net/trainingdml-ai/part2/1.0/ai_eoTask.json.
TEST METHOD	Validate the instance document against the ai_eoTask.json schema to verify the above requirement. The process may be using an appropriate software tool for validation or be a manual process that checks all definitions from the JSON schema specification.

A.6. Conformance Class: AI_Label

The AI_Label conformance class tests that the label objects are encoded according to the requirements.

CONFORMANCE CLASS A.8: AI_LABEL	
IDENTIFIER	`http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/ailabel`
REQUIREMENTS CLASS	/req/ailabel
PREREQUISITES	Conformance class A.1: `http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/base/jsonbasetype` Conformance class A.4: `http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/base/geospatialtype`
CONFORMANCE TESTS	Abstract test A.22: `/conf/ailabel/label` Abstract test A.23: `/conf/ailabel/scenelabel` Abstract test A.24: `/conf/ailabel/objectlabel` Abstract test A.25: `/conf/ailabel/pixellabel` Abstract test A.26: `/conf/ailabel/imageformatcode`

ABSTRACT TEST A.22	
IDENTIFIER	`/conf/ailabel/label`
REQUIREMENT	/req/ailabel/label

ABSTRACT TEST A.22

TEST PURPOSE	Verify that instance documents using the AI_Label JSON objects listed in Table 9 validate against the JSON schema specified in http://schemas.opengis.net/trainingdml-ai/part2/1.0/ai_label.json.
TEST METHOD	Validate the instance document against the ai_label.json schema to verify the above requirement. The process may be using an appropriate software tool for validation or be a manual process that checks all definitions from the JSON schema specification.

ABSTRACT TEST A.23

IDENTIFIER	`/conf/ailabel/scenelabel`
REQUIREMENT	/req/ailabel/scenelabel
TEST PURPOSE	Verify that instance documents using the AI_SceneLabel JSON objects listed in Table 10 validate against the JSON schema specified in http://schemas.opengis.net/trainingdml-ai/part2/1.0/ai_sceneLabel.json.
TEST METHOD	Validate the instance document against the ai_sceneLabel.json schema to verify the above requirement. The process may be using an appropriate software tool for validation or be a manual process that checks all definitions from the JSON schema specification.

ABSTRACT TEST A.24

IDENTIFIER	`/conf/ailabel/objectlabel`
REQUIREMENT	/req/ailabel/objectlabel
TEST PURPOSE	Verify that instance documents using the AI_ObjectLabel JSON objects listed in Table 11 validate against the JSON schema specified in http://schemas.opengis.net/trainingdml-ai/part2/1.0/ai_objectLabel.json.
TEST METHOD	Validate the instance document against the ai_objectLabel.json schema to verify the above requirement. The process may be using an appropriate software tool for validation or be a manual process that checks all definitions from the JSON schema specification.

ABSTRACT TEST A.25

IDENTIFIER	`/conf/ailabel/pixellabel`
REQUIREMENT	/req/ailabel/pixellabel

ABSTRACT TEST A.25	
TEST PURPOSE	Verify that instance documents using the AI_PixelLabel JSON objects listed in Table 12 validate against the JSON schema specified in http://schemas.opengis.net/trainingdml-ai/part2/1.0/ai_pixelLabel.json.
TEST METHOD	Validate the instance document against the ai_pixelLabel.json schema to verify the above requirement. The process may be using an appropriate software tool for validation or be a manual process that checks all definitions from the JSON schema specification.

ABSTRACT TEST A.26	
IDENTIFIER	`/conf/ailabel/imageformatcode`
REQUIREMENT	/req/ailabel/imageformatcode
TEST PURPOSE	Verify that instance documents using the AI_ImageFormatCode JSON objects conform to the requirements specified by the Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types RFC 2046.
TEST METHOD	Inspect the instance document to verify the above requirement.

A.7. Conformance Class: AI_Labeling

The AI_Labeling conformance class tests that the labeling objects are encoded according to the requirements.

CONFORMANCE CLASS A.9: AI_LABELING	
IDENTIFIER	`http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/ailabeling`
REQUIREMENTS CLASS	/req/ailabeling
PREREQUISITES	Conformance class A.1: `http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/base/jsonbasetype` Conformance class A.2: `http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/base/isometadatatype`
CONFORMANCE TESTS	Abstract test A.27: `/conf/ailabeling/labeling` Abstract test A.28: `/conf/ailabeling/labeler` Abstract test A.29: `/conf/ailabeling/labelingprocedure` Abstract test A.30: `/conf/ailabeling/labelingmethodcode`

ABSTRACT TEST A.27

IDENTIFIER	`/conf/ailabeling/labeling`
REQUIREMENT	/req/ailabeling/labeling
TEST PURPOSE	Verify that instance documents using the AI_Labeling JSON objects listed in Table 13 validate against the JSON schema specified in http://schemas.opengis.net/trainingdml-ai/part2/1.0/ai_labeling.json.
TEST METHOD	Validate the instance document against the ai_labeling.json schema to verify the above requirement. The process may be using an appropriate software tool for validation or be a manual process that checks all definitions from the JSON schema specification.

ABSTRACT TEST A.28

IDENTIFIER	`/conf/ailabeling/labeler`
REQUIREMENT	/req/ailabeling/labeler
TEST PURPOSE	Verify that instance documents using the AI_Labeler JSON objects listed in Table 14 validate against the JSON schema specified in http://schemas.opengis.net/trainingdml-ai/part2/1.0/ai_labeler.json.
TEST METHOD	Validate the instance document against the ai_labeler.json schema to verify the above requirement. The process may be using an appropriate software tool for validation or be a manual process that checks all definitions from the JSON schema specification.

ABSTRACT TEST A.29

IDENTIFIER	`/conf/ailabeling/labelingprocedure`
REQUIREMENT	/req/ailabeling/labelingprocedure
TEST PURPOSE	Verify that instance documents using the AI_LabelingProcedure JSON objects listed in Table 15 validate against the JSON schema specified in http://schemas.opengis.net/trainingdml-ai/part2/1.0/ai_labelingProcedure.json.
TEST METHOD	Validate the instance document against the ai_labelingProcedure.json schema to verify the above requirement. The process may be using an appropriate software tool for validation or be a manual process that checks all definitions from the JSON schema specification.

ABSTRACT TEST A.30	
IDENTIFIER	`/conf/ailabeling/labelingmethodcode`
REQUIREMENT	/req/ailabeling/labelingmethodcode
TEST PURPOSE	Verify that instance documents using the AI_LabelingMethodCode JSON objects validate against the JSON schema specified in http://schemas.opengis.net/trainingdml-ai/part2/1.0/ai_labelingMethodCode.json.
TEST METHOD	Validate the instance document against the ai_LabelingMethodCode.json schema to verify the above requirement. The process may be using an appropriate software tool for validation or be a manual process that checks all definitions from the JSON schema specification.

A.8. Conformance Class: AI_DataQuality

The AI_DataQuality conformance class tests that the data quality objects are encoded according to the requirements.

CONFORMANCE CLASS A.10: AI_DATAQUALITY	
IDENTIFIER	`http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/aidataquality`
REQUIREMENTS CLASS	/req/aidataquality
PREREQUISITES	Conformance class A.1: `http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/base/jsonbasetype` Conformance class A.2: `http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/base/isometadatatype` Conformance class A.3: `http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/base/isoqualitytype`
CONFORMANCE TEST	Abstract test A.31: `/conf/aidataquality/classbalancedegree`

ABSTRACT TEST A.31	
IDENTIFIER	`/conf/aidataquality/classbalancedegree`
REQUIREMENT	/req/aidataquality/classbalancedegree

ABSTRACT TEST A.31	
TEST PURPOSE	Verify that instance documents using the AI_ClassBalanceDegree JSON objects listed in Table 16 validate against the JSON schema specified in http://schemas.opengis.net/trainingdml-ai/part2/1.0/ai_classBalanceDegree.json.
TEST METHOD	Validate the instance document against the ai_classBalanceDegree.json schema to verify the above requirement. The process may be using an appropriate software tool for validation or be a manual process that checks all definitions from the JSON schema specification.

A.9. Conformance Class: AI_TDChangeset

The AI_TDChangeset conformance class tests that the TD changeset objects are encoded according to the requirements.

CONFORMANCE CLASS A.11: AI_TDCHANGESET	
IDENTIFIER	`http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/aitdchangeset`
REQUIREMENTS CLASS	/req/aitdchangeset
PREREQUISITES	Conformance class A.1: `http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/base/jsonbasetype` Conformance class A.6: `http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/aitrainingdata`
CONFORMANCE TEST	Abstract test A.32: `/conf/aitdchangeset/tdchangeset`

ABSTRACT TEST A.32	
IDENTIFIER	`/conf/aitdchangeset/tdchangeset`
REQUIREMENT	/req/aitdchangeset/tdchangeset
TEST PURPOSE	Verify that instance documents using the AI_TDChangeset JSON objects listed in Table 17 validate against the JSON schema specified in http://schemas.opengis.net/trainingdml-ai/part2/1.0/ai_tdChangeset.json.
TEST METHOD	Validate the instance document against the ai_tdChangeset.json schema to verify the above requirement. The process may be using an appropriate software tool for validation or be a manual process that checks all definitions from the JSON schema specification.

B

ANNEX B (INFORMATIVE) EXAMPLE

ANNEX B (INFORMATIVE) EXAMPLE

B.1. TrainingDataset Encoding Examples

B.1.1. WHU-RS19 Dataset

The WHU-RS19 dataset is widely used in scene classification of remote sensing images. This dataset is collected from Google Earth and has 19 classes including airport, beach, bridge, commercial, desert, farmland, football field, forest, industrial, meadow, mountain, park, parking, pond, port, railway station, residential, river, and viaduct. Each class contains around 50 images, with an image size of 600×600 and a resolution of 0.5 m.

An example of JSON encoding of the WHU-RS19 dataset following the TrainingDML-AI UML model can be found in https://github.com/opengeospatial/TrainingDML-AI_SWG/tree/main/use-cases/examples/1.0/WHU-RS19.json.

B.1.2. DOTA-v1.5 Dataset

The DOTA-v1.5 dataset is a large-scale dataset for object detection in aerial images. The sources for content in the dataset include Google Earth, Gaofen-2, and Jilin-1 imagery provided by China Resources Satellite Data Center. The 16 classes in DOTA-v1.5 are plane, ship, storage tank, baseball diamond, tennis court, basketball court, ground track field, harbor, bridge, large vehicle, small vehicle, helicopter, roundabout, soccer ball field, swimming pool, and container crane. Compared with other aerial image object detection datasets, the dataset has the largest number of classes. The images in the dataset have various image sizes (from 800×800 to 2000×2000) and resolutions (Google Earth/0.1 m-1 m, Gaofen-2/1 m, Jilin-1/0.72 m).

An example of JSON encoding of the DOTA-v1.5 dataset following the TrainingDML-AI UML model can be found in https://github.com/opengeospatial/TrainingDML-AI_SWG/tree/main/use-cases/examples/1.0/DOTA-v1.5.json.

B.1.3. KITTI 2D Object Detection Dataset

The KITTI 2D object detection dataset is a novel open-access dataset and benchmark for road area and ego-lane detection. KITTI 2D consists of 7481 annotated training images of high variability from the KITTI autonomous driving platform by two PointGrey Flea2 color cameras,

capturing a broad spectrum of urban street views and road scenes. The eight (8) classes in the KITTI 2D object detection dataset are car, van, truck, pedestrian, person_sitting, cyclist, tram, and misc. Compared with other street view object detection datasets, this dataset compresses diverse scenarios and captures real-world traffic situations, ranging from freeways over rural areas to inner-city scenes with many static and dynamic objects.

An example of JSON encoding of the KITTI 2D object detection dataset following the TrainingDML-AI UML model can be found in https://github.com/opengeospatial/TrainingDML-AI_SWG/tree/main/use-cases/examples/1.0/KITTI.json.

B.1.4. GID Dataset

The GID dataset is one of state-of-art land cover classification datasets. This dataset has a large spatial coverage covering many provinces in China with a relatively high spatial resolution (2 m). GID has two sets. One is the GID-5C. It has 150 images (image size 7200×6800) that are classified into 5 land cover classes. The other set is GID-15C. The images from GID-5C are sliced into 30,000 patches in GID-15C, which have three types of patch sizes (56×56, 112×112, 224×224) and are classified into 15 land cover classes.

An example of JSON encoding of the GID-5C dataset following the TrainingDML-AI UML model can be found in https://github.com/opengeospatial/TrainingDML-AI_SWG/tree/main/use-cases/examples/1.0/GID-5C.json.

B.1.5. Toronto3D Dataset

The Toronto3D dataset is a large urban outdoor point cloud dataset for segmentation collected by the Mobile Laser Scanning System. The dataset covers about 1 km of scene streets in Toronto, including four areas named L001, L002, L003, and L004, with a total of 78.3 million points. Each point in this dataset has 10 attributes representing the 3D position, RGB color, intensity, GPS time, scan angle rank, and category, respectively. This dataset has eight categories, including road, road mark, natural, building, utility line, pole, car, and fence.

An example of JSON encoding of the Toronto3D dataset following the TrainingDML-AI UML model can be found in https://github.com/opengeospatial/TrainingDML-AI_SWG/tree/main/use-cases/examples/1.0/Toronto_3D.json.

B.1.6. WHU-Building Dataset

The WHU-Building dataset is a change detection dataset collected from the Land Information New Zealand Data Service. The dataset is composed of images (with the resolution 0.2 m) in 2012 and 2016, covering 20.5 km2. It includes 12,796 and 16,077 buildings respectively in 2012 and 2016.

An example of JSON encoding of the WHU-Building dataset following the TrainingDML-AI UML model can be found in https://github.com/opengeospatial/TrainingDML-AI_SWG/tree/main/use-cases/examples/1.0/WHU-building.json.

B.1.7. California Change Detection Dataset

The California Change Detection Dataset is composed of two images and a label image. The first image is a Landsat 8 acquisition covering Sacramento County, Yuba County and Sutter County, California, on 5 January 2017. It has nine channels covering the spectrum from deep blue to short-wave infrared, plus two long-wave infrared channels. The second image was acquired on 18 February 2017 by Sentinel-1A over the same area after the occurrence of a flood. The image is recorded in polarizations VV and VH and augmented with the ratio between the two intensities as a third channel. All these channels are log-transformed.

An example of JSON encoding of the California change detection dataset following the TrainingDML-AI UML model can be found in https://github.com/opengeospatial/TrainingDML-AI_SWG/tree/main/use-cases/examples/1.0/UiT_HCD_California_2017.json.

B.1.8. WHU MVS Dataset

The WHU MVS dataset is a synthetic aerial dataset created for large-scale and high-resolution Earth surface reconstruction. The basic training sample of the dataset is a multi-view unit consisting of five aerial images, and their corresponding depth maps are taken as ground truth. There are a total of 5680 pairs of five-view aerial images in the dataset. All the images are simulated from a 3D surface model, which is produced by Smart3D software using Unmanned Aerial Vehicle (UAV) images and refined by manual editing.

An example of JSON encoding of the WHU MVS dataset following the TrainingDML-AI UML model can be found in https://github.com/opengeospatial/TrainingDML-AI_SWG/tree/main/use-cases/examples/1.0/WHU_MVS.json.

B.1.9. iSAID Dataset

The iSAID dataset is the first benchmark dataset for instance segmentation in aerial images. This large-scale and densely annotated dataset contains 655,451 object instances for 15 categories across 2,806 high-resolution images. The images of iSAID is the same as the DOTA-v1.0 dataset, which are mainly collected from the Google Earth, some are taken by satellite JL-1, the others are taken by satellite GF-2 of the China Centre for Resources Satellite Data and Application. The object categories in iSAID include: plane, ship, storage tank, baseball diamond, tennis court, basketball court, ground track field, harbor, bridge, large vehicle, small vehicle, helicopter, roundabout, soccer ball field and swimming pool.

An example of JSON encoding of the iSAID dataset following the TrainingDML-AI UML model can be found in https://github.com/opengeospatial/TrainingDML-AI_SWG/blob/main/use-cases/examples/1.0/iSAID.json.

B.2. DataQuality Encoding Example

B.2.1. WHU-RS19 Data Quality

An encoded data quality example of the WHU-RS19 datasets following the TrainingDML-AI UML model can be found in https://github.com/opengeospatial/TrainingDML-AI_SWG/tree/main/use-cases/examples/1.0/WHU-RS19-quality.json.

B.3. TDChangeset Encoding Example

B.3.1. DOTA-v1.5 Changeset

DOTA-v1.5 uses the same images as DOTA-v1.0, but the extremely small instances (less than 10 pixels) are also annotated. Moreover, a new category “container crane” is added. It contains 403,318 instances in total. The number of images and dataset splits are the same as DOTA-v1.0. This version was released for the DOAI Challenge 2019 on Object Detection in Aerial Images in conjunction with IEEE CVPR 2019.

An encoded changeset example between the DOTA-v1.0 and DOTA-v1.5 datasets following the TrainingDML-AI UML model can be found in https://github.com/opengeospatial/TrainingDML-AI_SWG/tree/main/use-cases/examples/1.0/DOTA-v1.5-changeset.json.

B.4. Non-EO Imagery TrainingDataset Encoding Examples

B.4.1. ERA5 Dataset

The ERA5 dataset is derived from in-situ observational data (Copernicus product), and we limit its usage scenario to the autoregression problem of time series data. Therefore, its label is the data itself. Similar to unsupervised learning, the autoregression task for time series data does not require additional labeled data. For this dataset, inheritance classes for AI_AbstractLabel are not defined, although this class is required in the existing standard (please note that these test cases are for future versions of the standard). In addition, additional attributes to support the complete representation of dataset information were added.

An example of JSON encoding of the ERA5 dataset following the TrainingDML-AI UML model can be found in https://github.com/opengeospatial/TrainingDML-AI_SWG/blob/main/use-cases/examples/1.0/ERA5_hourly_data.json.

B.4.2. SCIERC Dataset

The SCIERC dataset is derived from textual data, and its labels are the classification of the text. This dataset is a text classification problem, with the goal of information extraction and entity recognition. For this textual dataset, the Abstract class is inherited and AI_TextTrainingDataset, AI_TextTrainingData, AI_TextTask, and AI_EntityLabel respectively are defined. In addition, additional attributes to support the complete representation of dataset information were added.

An example of JSON encoding of the SCIERC dataset following the TrainingDML-AI UML model can be found in https://github.com/opengeospatial/TrainingDML-AI_SWG/blob/main/use-cases/examples/1.0/SCIERC.json.

B.4.3. nuScenes Dataset

The nuScenes dataset is a public large-scale dataset for autonomous driving developed by the team at Motional (formerly nuTonomy). The full dataset includes approximately 1.4M camera images, 390k LIDAR sweeps, 1.4M RADAR sweeps and 1.4M object bounding boxes in 40k keyframes. Although the training data may come from different domains, the 3D annotation boxes captured by numerous sensors in the same keyframe are targeted at the same object and are unique. Based on this, a 3D annotation box is used to organize each 3D object using AI_ObjectLabel. Since each training data and each 3D object require many additional attributes to be fully described, many additional attributes to provide a detailed description of the training dataset, training data, labels, etc. were added.

An example of JSON encoding of the nuScenes dataset following the TrainingDML-AI UML model can be found in https://github.com/opengeospatial/TrainingDML-AI_SWG/blob/main/use-cases/examples/1.0/nuScenes.json.

C

ANNEX C (INFORMATIVE) REVISION HISTORY

ANNEX C (INFORMATIVE) REVISION HISTORY

Table — Revision history

DATE	RELEASE	AUTHOR	PARAGRAPH MODIFIED	DESCRIPTION
2023-07-28	0.1	Peng Yue, Ruixiang Liu, Boyi Shangguan	All	Draft for internal review.
2023-12-15	0.2	Peng Yue, Ruixiang Liu, Jim Antonisse	Most	Revisions based on comments from Jim Antonisse.
2024-02-26	0.3	Peng Yue, Ruixiang Liu, Carl Reed	Most	Merge edits and comments from Carl Reed.
2024-06-09	0.4	Peng Yue, Ruixiang Liu	Chapter 2, 4, Annex A	Revisions after OAB review and public comments.
2026-02-28	1.0.1	Peng Yue, Baoxin Teng	Most	Revisions based on the DIS comments on ISO 19178-1.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] How Our Compliance Program Works, n.d. Open Geospatial Consortium [online]. Available from: https://www.ogc.org/how-our-compliance-program-works/ (accessed 11.3.24).