**Open Geospatial Consortium**

Submission Date: <2024-02-27>

Approval Date:   <yyyy-mm-dd>

Publication Date:   <yyyy-mm-dd>

External identifier of this OGC® document: <[http://www.opengis.net/doc/[{doc-type/}]{standard}/{m.n}](http://www.opengis.net/def/%5b%7bdoc-type/%7d%5d%7bstandard%7d/%7bm.n%7d)>

Internal reference number of this OGC® document: 24-006r1

Version: 1.0.0

Editor:   Peng Yue, Ruixiang Liu, Boyi Shangguan

OGC Training Data Markup Language for Artificial Intelligence (TrainingDML-AI) Part 2: JSON Encoding Standard

**Copyright notice**

Copyright © 2024 Open Geospatial Consortium  
To obtain additional rights of use, visit <https://www.ogc.org/ogc/Document>.

**License Agreement**

Permission for use/distribution of this document and any associated materials is subject to the terms of this License Agreement: <https://www.ogc.org/license>

**Warning**

This document is not an OGC Standard. This document is distributed for review and comment. This document is subject to change without notice and may not be referred to as an OGC Standard.

Document type:    OGC® Standard

Document subtype:    if applicable

Document stage:    Draft

Document language:  English

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Contents

[1. Scope 7](#_Toc168951737)

[2. Conformance 8](#_Toc168951738)

[3. Normative References 8](#_Toc168951739)

[4. Terms and Definitions 9](#_Toc168951740)

[4.1 Artificial Intelligence (AI) 9](#_Toc168951741)

[4.2 Machine Learning (ML) 9](#_Toc168951742)

[4.3 Deep Learning (DL) 9](#_Toc168951743)

[4.4 Dataset 10](#_Toc168951744)

[4.5 Training Dataset 10](#_Toc168951745)

[4.6 Label 10](#_Toc168951746)

[4.7 Class 10](#_Toc168951747)

[4.8 Task 10](#_Toc168951748)

[4.9 Provenance 10](#_Toc168951749)

[4.10 Quality 11](#_Toc168951750)

[4.11 Earth Observation 11](#_Toc168951751)

[4.12 Scene Classification 11](#_Toc168951752)

[4.13 Object Detection 11](#_Toc168951753)

[4.14 Semantic Segmentation 11](#_Toc168951754)

[4.15 Change Detection 11](#_Toc168951755)

[4.16 3D Model Reconstruction 11](#_Toc168951756)

[4.17 Generative Model 12](#_Toc168951757)

[4.18 JavaScript Object Notation (JSON) 12](#_Toc168951758)

[4.19 JSON Schema 12](#_Toc168951759)

[4.20 Training Dataset Publisher 12](#_Toc168951760)

[5. Conventions 12](#_Toc168951761)

[5.1 Identifiers 12](#_Toc168951762)

[5.2 Abbreviated Terms 13](#_Toc168951763)

[6. Overview 13](#_Toc168951764)

[6.1 JavaScript Object Notation 14](#_Toc168951765)

[7. Requirements for TrainingDML-AI JSON Encoding 14](#_Toc168951766)

[7.1 Requirements Class: Base 14](#_Toc168951767)

[7.1.1 Requirements Class: JSON Base Type 14](#_Toc168951768)

[7.1.2 Requirements Class: ISO Metadata Type 16](#_Toc168951769)

[7.1.3 Requirements Class: ISO Quality Type 20](#_Toc168951770)

[7.1.4 Requirements Class: Geospatial Type 22](#_Toc168951771)

[7.2 Requirements Class: AI\_TrainingDataset 23](#_Toc168951772)

[7.3 Requirements Class: AI\_TrainingData 30](#_Toc168951773)

[7.4 Requirements Class: AI\_Task 34](#_Toc168951774)

[7.5 Requirements Class: AI\_Label 36](#_Toc168951775)

[7.6 Requirements Class: AI\_Labeling 40](#_Toc168951776)

[7.7 Requirements Class: AI\_DataQuality 44](#_Toc168951777)

[7.8 Requirements Class: AI\_TDChangeset 46](#_Toc168951778)

[Annex A Abstract Test Suite (Normative) 49](#_Toc168951779)

[A.1 Introduction 49](#_Toc168951780)

[A.2 Conformance Class: Base 49](#_Toc168951781)

[A.2.1 Conformance Class: JSON Base Type 49](#_Toc168951782)

[A.2.2 Conformance Class: ISO Metadata Type 50](#_Toc168951783)

[A.2.3 Conformance Class: ISO Quality Type 52](#_Toc168951784)

[A.2.4 Conformance Class: Geospatial Type 53](#_Toc168951785)

[A.3 Conformance Class: AI\_TrainingDataset 53](#_Toc168951786)

[A.4 Conformance Class: AI\_TrainingData 55](#_Toc168951787)

[A.5 Conformance Class: AI\_Task 57](#_Toc168951788)

[A.6 Conformance Class: AI\_Label 58](#_Toc168951789)

[A.7 Conformance Class: AI\_Labeling 60](#_Toc168951790)

[A.8 Conformance Class: AI\_DataQuality 62](#_Toc168951791)

[A.9 Conformance Class: AI\_TDChangeset 63](#_Toc168951792)

[Annex B Example (Informative) 65](#_Toc168951793)

[B.1 TrainingDataset Encoding Examples 65](#_Toc168951794)

[B.1.1 WHU-RS19 Dataset 65](#_Toc168951795)

[B.1.2 DOTA-v1.5 Dataset 65](#_Toc168951796)

[B.1.3 KITTI 2D Object Detection Dataset 65](#_Toc168951797)

[B.1.4 GID Dataset 66](#_Toc168951798)

[B.1.5 Toronto3D Dataset 66](#_Toc168951799)

[B.1.6 WHU-Building Dataset 66](#_Toc168951800)

[B.1.7 California Change Detection Dataset 66](#_Toc168951801)

[B.1.8 WHU MVS Dataset 67](#_Toc168951802)

[B.1.9 iSAID Dataset 67](#_Toc168951803)

[B.2 DataQuality Encoding Example 67](#_Toc168951804)

[B.2.1 WHU-RS19 Data Quality 67](#_Toc168951805)

[B.3 TDChangeset Encoding Example 68](#_Toc168951806)

[B.3.1 DOTA-v1.5 Changeset 68](#_Toc168951807)

[B.4 Non-EO Imagery TrainingDataset Encoding Examples 68](#_Toc168951808)

[B.4.1 ERA5 Dataset 68](#_Toc168951809)

[B.4.2 SCIERC Dataset 68](#_Toc168951810)

[B.4.3 nuScenes Dataset 68](#_Toc168951811)

[Annex C Revision History (Informative) 70](#_Toc168951812)

[Annex D Bibliography 71](#_Toc168951813)

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 JSON. JavaScript Object Notation (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.

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.

Security Considerations

No security considerations have been made for this Standard.

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 Automobile Co., Ltd

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 |
| 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 Automobile Co., Ltd |

Acknowledgments

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, Kaixuan Wang, Zhipeng Cao, Shuaiqi Liu, Ming Zhao, Hanwen Xu, Haipeng Deng, Baoxin Teng.

# 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.

# Conformance

This Standard 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 (normative) 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[[1]](#footnote-1).

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

This standard 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.

# Normative References

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

[OGC: OGC 23-008r3, OGC Training Data Markup Language for Artificial Intelligence (TrainingDML-AI) Part1: Conceptual Model Standard, 2023](https://docs.ogc.org/is/23-008r3/23-008r3.html)

[IETF: RFC 7159, The JavaScript Object Notation (JSON) Data Interchange Format, 2014](http://www.ietf.org/rfc/rfc7159.txt)

[IETF: RFC 7946, The GeoJSON Format, 2016](https://tools.ietf.org/html/rfc7946)

[IETF: RFC 3986, Uniform Resource Identifiers (URI): Generic Syntax, 2005](http://www.ietf.org/rfc/rfc3986.txt)

[IETF: RFC 3339, Date and Time on the Internet: Timestamps, 2002](http://www.ietf.org/rfc/rfc3339.txt)

[IETF: RFC 2046, Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types, 1996](https://www.ietf.org/rfc/rfc2046.txt)

[ISO 19107:2019 Geographic information — Spatial schema](https://www.iso.org/standard/66175.html)

[ISO 19115-1:2014 Geographic information — Metadata — Part 1: Fundamentals](https://www.iso.org/standard/53798.html)

[ISO 19157-1:2023 Geographic information — Data quality — Part 1: General requirements](https://www.iso.org/standard/78900.html)

# Terms and Definitions

This document used the terms defined in [OGC Policy Directive 49](https://portal.ogc.org/public_ogc/directives/directives.php), 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.

## 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

## 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

## 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>

## Dataset

identifiable collection of data

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.

[SOURCE: ISO 19115‑1:2014, 4.3]

## Training Dataset

is a collection of samples, often labeled in terms of 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)](https://portal.ogc.org/files/?artifact_id=41579). 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

## 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

## 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]

## 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

## 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/>)

## 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

## 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>)

## 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

## 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

## 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

## Change Detection

task that finds the changes in an area between images taken at different times.

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

## 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

## 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

## 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: ECMA-404 The JSON data interchange syntax 2nd edition, December 2017

## JSON Schema

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

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

## 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.

# Conventions

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

## 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.

## 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 Modeling Language

URL Uniform Resource Locator

UTC Coordinated Universal Time

W3C World Wide Web Consortium

XML Extensible Markup Language

# 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.

## 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 ECMA-404. 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.

# Requirements for TrainingDML-AI JSON Encoding

## Requirements Class: Base

### 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 | /req/base/jsonbasetype/json |
| Requirement | /req/base/jsonbasetype/datetime |
| Requirement | /req/base/jsonbasetype/namedvalue |
| Requirement | /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 ECMA-404 |

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](https://datatracker.ietf.org/doc/html/rfc3339#section-5.6)].  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:

1. “2002-05-30T09:00:00”
2. “2002-05-30T09:30:10.5”
3. “2002-05-30T09:30:10Z”
4. “2002-09-24”
5. “2002-09-24Z”
6. “09:30:10”
7. “09:30:10.5”
8. “09:30:10Z”

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:

1. {“key”: “forest”, “value”: “RGB(0,255,255)”}
2. {“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](https://datatracker.ietf.org/doc/html/rfc3986#section-4.1)]. |

Examples:

1. “http://www.opengeospatial.org”
2. “/file.txt”

### 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 | /req/base/isometadatatype/band |
| Requirement | /req/base/isometadatatype/extent |
| Requirement | /req/base/isometadatatype/citation |
| Requirement | /req/base/isometadatatype/scope |

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:

1. {“name”: [{“code”: “red”}]}
2. {“name”: [{“code”: “B4”}]}
3. {“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 the GeoJSON bounding box encoding as defined in The GeoJSON Format [[RFC 7946 Section 5](https://datatracker.ietf.org/doc/html/rfc7946#section-5)],  or a JSON object matching the XML Schema type as defined in:  <https://schemas.isotc211.org/19115/-1/gex/1.3.0/extent.xsd> |

Examples:

1. [120.0, 30.0, 130.0, 40.0]
2. [120.0, 30.0, 10.0, 130.0, 40.0, 20.0]
3. {

“geographicElement”: [{

“westBoundLongitude”: -171.76409,

“eastBoundLongitude”: -157.86768,

“southBoundLatitude”: -14.42443,

“northBoundLatitude”: 21.31573

}]

}

1. {

"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 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:

1. {

“level”: “dataset”,

“levelDescription”: [{

“dataset”: “whu\_rs19”

}]

}

1. {

"level": "attribute",

"levelDescription": [{

"attributes": "completeness"

}]

}

1. {

"level": "coverage",

"extent": [{

"geographicElement": [{

"westBoundLongitude": -171.76409,

"eastBoundLongitude": -157.86768,

"southBoundLatitude": -14.42443,

"northBoundLatitude": 21.31573

}]

}]

}

1. {

"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]

}

}

}]

}]

}]

}

### 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 | /req/base/isoqualitytype/element |

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

|  |  |
| --- | --- |
| Requirement 9 | /req/base/isoqualitytype/element  Each QualityElement value SHALL be encoded as a JSON object with properties shown in Table 1. |

Table 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..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": "%"

}

}

]

}

### 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 | /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](https://datatracker.ietf.org/doc/html/rfc7946)]

|  |  |
| --- | --- |
| Requirement 10 | /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](https://datatracker.ietf.org/doc/html/rfc7946#section-3.2)]. |

Examples of Feature encodings are:

1. {“type”: “Feature”, “geometry”: {“type”: “Point”, “coordinates”: [120.0, 30.0]}, “properties”: {“class”: “station”}}
2. {“type”: “Feature”, “geometry”: {“type”: “LineString”, “coordinates”: [[120.0, 30.0], [130.0, 40.0]]}, “properties”: {“class”: “road”}}
3. {“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”}}

## 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 specifiedin 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 |
| Dependency | /req/aitask |
| Dependency | /req/ailabeling |
| Dependency | /req/aidataquality |
| Dependency | /req/aitdchangeset |
| Requirement | /req/aitrainingdataset/trainingdataset |
| Requirement | /req/aitrainingdataset/metricsinliterature |
| Requirement | /req/aitrainingdataset/eotrainingdataset |

The AI\_TrainingDataset object is encoded as a JSON object with properties shown in Table 2.

|  |  |
| --- | --- |
| Requirement 11 | /req/aitrainingdataset/trainingdataset  Each AI\_TrainingDataset object SHALL implement the Mandatory properties shown in Table 2. |

Table AI\_TrainingDataset properties

|  |  |  |  |
| --- | --- | --- | --- |
| **JSON Property** | **Definition** | **Data type and values** | **Obligation** |
| type | Type of the training dataset. | “AI\_AbstractTrainingDataset” | 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 |
| 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 |
| classificationSchema | Classification schema for classes used in the AI training dataset. | CharacterString [0..1] | Optional |
| classes | Classes used in the AI training dataset. | NamedValue [0..\*] | Optional |
| 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}],

“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 12 | /req/aitrainingdataset/metricsinliterature  Each AI\_MetricsInLiterature value SHALL implement the Mandatory properties shown in Table 3. |

Table 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 13 | /req/aitrainingdataset/eotrainingdataset  Each AI\_EOTrainingDataset object SHALL implement the Mandatory properties both shown in Table 2 and Table 4. |

Table AI\_EOTrainingDataset properties

|  |  |  |  |
| --- | --- | --- | --- |
| **JSON Property** | **Definition** | **Data type and values** | **Obligation** |
| type | Type of the training dataset. | “AI\_EOTrainingDataset” | Mandatory |
| extent | Spatial extent of the EO training dataset. | EX\_Extent [0..1] | Optional |
| bands | Bands description of the images used in the EO training dataset. | MD\_Band [0..\*] | Optional |
| imageSize | Size of the images used in the EO training dataset. | ChracterString [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],

"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"}]}, …]

}

## 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 | /req/aitrainingdata/trainingdata |
| Requirement | /req/aitrainingdataset/trainingtypecode |
| Requirement | /req/aitrainingdata/eotrainingdata |

The AI\_TrainingData object is encoded as a JSON object with properties shown in Table 5.

|  |  |
| --- | --- |
| Requirement 14 | /req/aitrainingdataset/trainingdata  Each AI\_TrainingData object SHALL implement the Mandatory properties shown in Table 5. |

Table 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 |
| labels | Labels in the training data. | AI\_Label [1..\*] | Mandatory |
| 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"

}],

"dataURL": ["image/Airport/airport\_01.jpg"],

"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 15 | /req/aitrainingdataset/trainingtypecode  Each AI\_TrainingTypeCode value SHALL be a text string whose value is one of “training”, “validation”, “test” or “retraining”. |

Examples:

1. “training”
2. “validation”
3. “test”
4. “retraining”

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

|  |  |
| --- | --- |
| Requirement 16 | /req/aitrainingdataset/eotrainingdata  Each AI\_EOTrainingData object SHALL implement the Mandatory properties as defined in Table 5 and Table 6. |

Table AI\_EOTrainingData properties

|  |  |  |  |
| --- | --- | --- | --- |
| **JSON Property** | **Definition** | **Data type and values** | **Obligation** |
| type | Type of the EO training data. | “AI\_EOTrainingData” | Mandatory |
| extent | Spatial extent of the individual EO training sample. | EX\_Extent [0..1] | Optional |
| dataTime | Date time when the EO data was obtained. | DateTime [0..\*] | Optional |
| dataURL | URL of the EO data, including both relative and absolute paths, which can encompass local paths, network addresses, and more. | URL [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"],

"labels": [{

"type": "AI\_SceneLabel",

"class": "Airport"

}]

}

## 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 | /req/aitask/task |
| Requirement | /req/aitask/eotask |

The AI\_Task object is encoded as a JSON object with properties as shown in Table 7.

|  |  |
| --- | --- |
| Requirement 17 | /req/aitask/task  Each AI\_Task object SHALL implement the Mandatory properties shown in Table 7. |

Table 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.

|  |  |
| --- | --- |
| Requirement 18 | /req/aitask/eotask  Each AI\_EOTask object SHALL implement the Mandatory properties shown in Table 7 and Table 8. |

Table 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”

}

## 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 | /req/ailabel/label |
| Requirement | /req/ailabel/scenelabel |
| Requirement | /req/ailabel/objectlabel |
| Requirement | /req/ailabel/pixellabel |
| Requirement | /req/ailabel/imageformatcode |

The AI\_Label object is encoded as a JSON object with properties as shown in Table 9.

|  |  |
| --- | --- |
| Requirement 19 | /req/ailabel/label  Each AI\_Label object SHALL implement the Mandatory properties shown in Table 9. |

Table 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 20 | /req/ailabel/scenelabel  Each AI\_SceneLabel object SHALL implement the Mandatory properties shown in Table 10. |

Table 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 21 | /req/ailabel/objectlabel  Each AI\_ObjectLabel object SHALL implement the Mandatory properties shown in Table 11. |

Table 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]]]}},

“bboxType”: “Horizontal BBox”

}

The AI\_PixelLabel object is encoded as a JSON object with properties as shown in Table 12.

|  |  |
| --- | --- |
| Requirement 22 | /req/ailabel/pixellabel  Each AI\_PixelLabel object SHALL implement the Mandatory properties shown in Table 12. |

Table 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. | URL [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](https://www.ietf.org/rfc/rfc2046.txt)].

|  |  |
| --- | --- |
| Requirement 23 | /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](https://www.ietf.org/rfc/rfc2046.txt)]. |

Examples:

1. “image/tiff; application=geotiff”
2. “application/x-netcdf”
3. “image/png”
4. “image/jp2”

## 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 | /req/ailabeling/labeling |
| Requirement | /req/ailabeling/labeler |
| Requirement | /req/ailabeling/labelingprocedure |
| Requirement | /req/ailabeling/labelingmethodcode |

The AI\_Labeling object is encoded as a JSON object with properties shown in Table 13.

|  |  |
| --- | --- |
| Requirement 24 | /req/ailabeling/labeling  Each AI\_Labeling object SHALL implement the Mandatory properties shown in Table 13. |

Table 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 |
| 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”

}]

},

“labelers”: [{..}],

“procedure”: {..}

}

The AI\_Labeler object is encoded as a JSON object with properties as shown in Table 14.

|  |  |
| --- | --- |
| Requirement 25 | /req/ailabeling/labeler  Each AI\_Labeler object SHALL implement the Mandatory properties shown in Table 14. |

Table 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”

}

The AI\_LabelingProcedure object is encoded as a JSON object with properties as shown in Table 15.

|  |  |
| --- | --- |
| Requirement 26 | /req/ailabeling/labelingprocedure  Each AI\_LabelingProcedure object SHALL implement the Mandatory properties shown in Table 15. |

Table 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 “unkonwn”.

|  |  |
| --- | --- |
| Requirement 27 | /req/ailabeling/labelingmethodcode  Each AI\_LabelingMethodCode value SHALL be a text string whose value is one of “manual”, “semi-automatic”, “automatic” or “unknown”. |

Examples:

1. “manual”
2. “semi-automatic”
3. “automatic”
4. “unknown”

## Requirements Class: AI\_DataQuality

The AI\_DataQuality 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/aidataquality | |
| Dependency | JSON |
| Dependency | /req/base/jsonbasetype |
| Dependency | /req/base/isometadatatype |
| Dependency | /req/base/isoqualitytype |
| Requirement | /req/aidataquality/classbalancedegree |

The AI\_ClassBalanceDegree object is encoded as a JSON object with properties as shown in Table 16.

|  |  |
| --- | --- |
| Requirement 28 | /req/aidataquality/classbalancedegree  Each AI\_ClassBalanceDegree object SHALL implement the Mandatory properties as shown in Table 16. |

Table 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..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": "%"

}

}

]

}

## 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/tdtrainingdata |
| Requirement | /req/aitdchangeset/tdchangeset |

The AI\_TDChangeset object is encoded as a JSON object with properties shown in Table 17.

|  |  |
| --- | --- |
| Requirement 29 | /req/aitdchangeset/tdchangeset  Each AI\_TDChangeset object SHALL implement the Mandatory properties as shown in Table 17. |

Table 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”, “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”}, …]}]

}

1. Abstract Test Suite (Normative)
   1. Introduction

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

* 1. Conformance Class: Base

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

Conformance Class: JSON Base Type

|  |  |
| --- | --- |
| Conformance Class | |
| http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/base/jsonbasetype | |
| Requirements class | /req/base/jsonbasetype |

|  |  |
| --- | --- |
| Abstract Test | |
| Test ID | /conf/base/jsonbasetype/json |
| 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 | |
| Test ID | /conf/base/jsonbasetype/datetime |
| Requirement | /req/base/jsonbasetype/datetime |
| Test purpose | Verify that JSON instance documents claiming conformance to this specification contain valid DateTime values according to Date and Time on the Internet: Timestamps [[RFC 3339 Section 5.6](https://datatracker.ietf.org/doc/html/rfc3339#section-5.6)]. |
| Test method | Inspect the instance document to verify the above requirement. |

|  |  |
| --- | --- |
| Abstract Test | |
| Test ID | /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/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 | |
| Test ID | /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](https://datatracker.ietf.org/doc/html/rfc3986#section-4.1)]. A URL value can be absolute or relative and may have an optional fragment identifier. |
| Test method | Inspect the instance document to verify the above requirement. |

Conformance Class: ISO Metadata Type

|  |  |
| --- | --- |
| Conformance Class | |
| http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/base/isometadatatype | |
| Requirements class | /req/base/isometadatatype |

|  |  |
| --- | --- |
| Abstract Test | |
| Test ID | /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/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 | |
| Test ID | /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/1.0/ex_extent.json>. |
| 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 | |
| Test ID | /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/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 | |
| Test ID | /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/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. |

Conformance Class: ISO Quality Type

|  |  |
| --- | --- |
| Conformance Class | |
| http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/base/isoqualitytype | |
| Requirements class | /req/base/isoqualitytype |

|  |  |
| --- | --- |
| Abstract Test | |
| Test ID | /conf/base/isoqualitytype/element |
| 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/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. |

Conformance Class: Geospatial Type

|  |  |
| --- | --- |
| Conformance Class | |
| http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/base/geospatialtype | |
| Requirements class | /req/base/geospatialtype |

|  |  |
| --- | --- |
| Abstract Test | |
| Test ID | /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. |

* 1. Conformance Class: AI\_TrainingDataset

The AI\_TrainingDataset conformance class tests that the training dataset object is encoded according to the requirements.

|  |  |
| --- | --- |
| Conformance Class | |
| http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/aitrainingdataset | |
| Requirements class | /req/aitrainingdataset |
| Dependency | /conf/base/jsonbasetype |
| Dependency | /conf/base/isometadatatype |
| Dependency | /conf/aitrainingdata |
| Dependency | /conf/aitask |
| Dependency | /conf/ailabeling |
| Dependency | /conf/aidataquality |
| Dependency | /conf/aitdchangeset |

|  |  |
| --- | --- |
| Abstract Test | |
| Test ID | /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/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 | |
| Test ID | /conf/aitrainingdataset/metricsinliterature |
| Requirement | /req/aitrainingdataset/metricsinliterature |
| 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/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 | |
| Test ID | /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/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. |

* 1. Conformance Class: AI\_TrainingData

The AI\_TrainingData conformance class tests that the training data objects are encoded according to the requirements.

|  |  |
| --- | --- |
| Conformance Class | |
| http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/aitrainingdata | |
| Requirements class | /req/aitrainingdata |
| Dependency | /conf/base/jsonbasetype |
| Dependency | /conf/base/isometadatatype |
| Dependency | /conf/ailabel |
| Dependency | /conf/ailabeling |
| Dependency | /conf/aidataquality |

|  |  |
| --- | --- |
| Abstract Test | |
| Test ID | /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/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 | |
| Test ID | /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/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 | |
| Test ID | /conf/aitrainingdata/eotrainingdata |
| Requirement | /req/aitrainingdata/eotrainingdata |
| 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/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. |

* 1. Conformance Class: AI\_Task

The AI\_Task conformance class tests that the task objects are encoded according to the requirements.

|  |  |
| --- | --- |
| Conformance Class | |
| http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/aitask | |
| Requirements class | /req/aitask |
| Dependency | /conf/base/jsonbasetype |

|  |  |
| --- | --- |
| Abstract Test | |
| Test ID | /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/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 | |
| Test ID | /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/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. |

* 1. Conformance Class: AI\_Label

The AI\_Label conformance class tests that the label objects are encoded according to the requirements.

|  |  |
| --- | --- |
| Conformance Class | |
| http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/ailabel | |
| Requirements class | /req/ailabel |
| Dependency | /conf/base/jsonbasetype |
| Dependency | /conf/base/geospatialtype |

|  |  |
| --- | --- |
| Abstract Test | |
| Test ID | /conf/ailabel/label |
| Requirement | /req/ailabel/label |
| 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/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 | |
| Test ID | /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/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 | |
| Test ID | /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/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 | |
| Test ID | /conf/ailabel/pixellabel |
| Requirement | /req/ailabel/pixellabel |
| 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/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 | |
| Test ID | /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](https://www.ietf.org/rfc/rfc2046.txt)]. |
| Test method | Inspect the instance document to verify the above requirement. |

* 1. Conformance Class: AI\_Labeling

The AI\_Labeling conformance class tests that the labeling objects are encoded according to the requirements.

|  |  |
| --- | --- |
| Conformance Class | |
| http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/ailabeling | |
| Requirements class | /req/ailabeling |
| Dependency | /conf/base/jsonbasetype |
| Dependency | /conf/base/isometadatatype |

|  |  |
| --- | --- |
| Abstract Test | |
| Test ID | /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/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 | |
| Test ID | /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/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 | |
| Test ID | /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/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 | |
| Test ID | /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/1.0/ai\_labelingMethodCode.json](http://schemas.opengis.net/trainingdml-ai/1.0/ai_trainingTypeCode.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. |

Conformance Class: AI\_DataQuality

The AI\_DataQuality conformance class tests that the data quality objects are encoded according to the requirements.

|  |  |
| --- | --- |
| Conformance Class | |
| http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/aidataquality | |
| Requirements class | /req/aidataquality |
| Dependency | /conf/base/jsonbasetype |
| Dependency | /conf/base/isometadatatype |
| Dependency | /conf/base/isoqualitytype |

|  |  |
| --- | --- |
| Abstract Test | |
| Test ID | /conf/aidataquality/classbalancedegree |
| Requirement | /req/aidataquality/classbalancedegree |
| 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/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. |

* 1. Conformance Class: AI\_TDChangeset

The AI\_TDChangeset conformance class tests that the TD changeset objects are encoded according to the requirements.

|  |  |
| --- | --- |
| Conformance Class | |
| http://www.opengis.net/spec/TrainingDML-AI-2/1.0/conf/aitdchangeset | |
| Requirements class | /req/aitdchangeset |
| Dependency | /conf/base/jsonbasetype |
| Dependency | /conf/aitrainingdata |

|  |  |
| --- | --- |
| Abstract Test | |
| Test ID | /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/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. |

1. Example (Informative)

TrainingDataset Encoding Examples

* + 1. WHU-RS19 Dataset

[The WHU-RS19 dataset](https://captain-whu.github.io/BED4RS/) 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>.

* + 1. DOTA-v1.5 Dataset

[The DOTA-v1.5 dataset](https://captain-whu.github.io/DOTA/) 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>.

* + 1. KITTI 2D Object Detection Dataset

[The KITTI 2D object detection dataset](http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=2d) 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>.

* + 1. GID Dataset

[The GID dataset](https://x-ytong.github.io/project/GID.html) 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>.

* + 1. Toronto3D Dataset

[The Toronto3D dataset](https://github.com/WeikaiTan/Toronto-3D) 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>.

* + 1. WHU-Building Dataset

[The WHU-Building dataset](http://gpcv.whu.edu.cn/data/building_dataset.html) 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>.

* + 1. California Change Detection Dataset

[The California Change Detection Dataset](https://arxiv.org/abs/1909.05948) 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>.

* + 1. WHU MVS Dataset

[The WHU MVS dataset](http://gpcv.whu.edu.cn/data/WHU_MVS_Stereo_dataset.html) 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>.

* + 1. iSAID Dataset

[The iSAID dataset](https://captain-whu.github.io/iSAID/) 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 manily 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>.

DataQuality Encoding Example

* + 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>.

TDChangeset Encoding Example

* + 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>.

Non-EO Imagery TrainingDataset Encoding Examples

* + 1. ERA5 Dataset

[The ERA5 dataset](https://cds.climate.copernicus.eu/cdsapp#!/search?type=dataset&text=ERA5) 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>.

* + 1. SCIERC Dataset

[The SCIERC dataset](https://nlp.cs.washington.edu/sciIE/) 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/SCIRec.json>.

* + 1. nuScenes Dataset

[The nuScenes dataset](https://www.nuscenes.org/nuscenes) 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>.

1. Revision History (Informative)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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. |
|  |  |  |  |  |
|  |  |  |  |  |

1. Bibliography

[1] Yue, P., ed., 2023. OGC Training Data Markup Language for Artificial Intelligence (TrainingDML-AI) Part1: Conceptual Model Standard, OGC 23-008r3. Wayland, MA: Open Geospatial Consortium Inc. <https://docs.ogc.org/is/23-008r3/23-008r3.html>

[2] Freed, N., 1996. RFC 2046. Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types. <https://www.ietf.org/rfc/rfc2046.txt>

[3] Klyne, G., 2002. RFC 3339. Date and Time on the Internet: Timestamps. <http://www.ietf.org/rfc/rfc3339.txt>

[4] Berners-Lee, T., 2005. RFC 3986. Uniform Resource Identifier (URI): Generic Syntax. <http://www.ietf.org/rfc/rfc3986.txt>

[5] Bray, T., ed., 2014. RFC 7159. The JavaScript Object Notation (JSON) Data Interchange Format. <http://www.ietf.org/rfc/rfc7159.txt>

[6] Butler, H., ed., 2016. RFC 7946. The GeoJSON Format. <http://www.ietf.org/rfc/rfc7946.txt>

[7] ISO, 2019. ISO 19107:2019. Geographic information — Spatial schema. <https://www.iso.org/standard/66175.html>

[8] ISO, 2022. ISO 19157-1:2023. Geographic information — Data quality. <https://www.iso.org/standard/78900.html>

[9] ISO, 2014. 19115-1:2014, Geographic information — Metadata — Part 1: Fundamentals. <https://www.iso.org/standard/53798.html>

[10] Landry, T., ed., 2018. OGC Testbed-14: Machine Learning Engineering Report, OGC 18-038r2. Wayland, MA: Open Geospatial Consortium Inc. <https://docs.ogc.org/per/18-038r2.html>

[11] Meek, S., ed., 2019. OGC Testbed-15: Machine Learning Engineering Report, OGC 19-027r2. Wayland, MA: Open Geospatial Consortium Inc. <https://docs.ogc.org/per/19-027r2.html>

[12] Schumann, G., ed., 2020. OGC Testbed-16: Machine Learning Training Data Engineering Report, OGC 20-018. Wayland, MA: Open Geospatial Consortium Inc. <https://docs.ogc.org/per/20-015r2.html>

[13] Yue, P., Shangguan, B., Hu, L., Jiang, L., Zhang, C., Cao, Z., Pan, Y., 2022. Towards a training data model for artificial intelligence in earth observation. International Journal of Geographical Information Science, 1-25. <https://doi.org/10.1080/13658816.2022.2087223>

1. [www.opengeospatial.org/cite](http://www.opengeospatial.org/cite) [↑](#footnote-ref-1)