Reference number of working document: ISO/TC 000/SC 0 N 000

Date: yyyy-mm-dd

Reference number of document: ISO/WD 7673-2

Committee identification: ISO/TC 23/SC 19/WG 1

Secretariat: XXXX

Agricultural Field Operations Data — Part 2: Observations and Measurements

Document type: International standard

Document subtype: if applicable

Document stage: (20) Preparation

Document language: E

*Élément introductif — Élément principal — Partie n: Titre de la partie*

Warning

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

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

**Copyright notice**

This ISO document is a working draft or committee draft and is copyright-protected by ISO. While the reproduction of working drafts or committee drafts in any form for use by participants in the ISO standards development process is permitted without prior permission from ISO, neither this document nor any extract from it may be reproduced, stored or transmitted in any form for any other purpose without prior written permission from ISO.

Requests for permission to reproduce this document for the purpose of selling it should be addressed as shown below or to ISO’s member body in the country of the requester:

*[Indicate :*

*the full address*

*telephone number*

*fax number*

*telex number*

*and electronic mail address*

*as appropriate, of the Copyright Manager of the ISO member body responsible for the secretariat of the TC or SC within the framework of which the draft has been prepared]*

Reproduction for sales purposes may be subject to royalty payments or a licensing agreement.

Violators may be prosecuted.

Contents Page

[Foreword vi](#_Toc161736920)

[Introduction vii](#_Toc161736921)

[1 Scope 1](#_Toc161736922)

[1.1 How the standard is structured 1](#_Toc161736923)

[1.2 Implementing ISO 19156 for agrifood systems 2](#_Toc161736924)

[2 Conformance 2](#_Toc161736925)

[3 Normative references 2](#_Toc161736926)

[4 Terms and definitions 2](#_Toc161736927)

[4.aggregation\_procedure (not ref.) aggregation procedure 2](#_Toc161736928)

[4.alias alias 3](#_Toc161736929)

[4.1 code component 3](#_Toc161736930)

[4.1 code component definition 3](#_Toc161736931)

[4.2 configuration data 3](#_Toc161736932)

[4.crop\_nutritionlorem ipsum 4](#_Toc161736933)

[4.crop\_protectionlorem ipsum 4](#_Toc161736934)

[4.data\_element lorem ipsum 4](#_Toc161736935)

[4.data\_loggerlorem ipsum 4](#_Toc161736936)

[4.data\_qualitylorem ipsum 4](#_Toc161736937)

[4.data\_quality\_metric lorem ipsum 4](#_Toc161736938)

[4.data\_stream lorem ipsum 4](#_Toc161736939)

[4.data\_type lorem ipsum 4](#_Toc161736940)

[4.3 dataset 4](#_Toc161736941)

[4.device lorem ipsum 4](#_Toc161736942)

[4.4 direct position 4](#_Toc161736943)

[4.element lorem ipsum 4](#_Toc161736944)

[4.enumeration\_item lorem ipsum 4](#_Toc161736945)

[4.5 feature 4](#_Toc161736946)

[4.feature\_of\_interest lorem ipsum 5](#_Toc161736947)

[4.gazetteer (is referenced to part 1) lorem ipsum 5](#_Toc161736948)

[4.6 grower data 5](#_Toc161736949)

[4.7 incoming observation 5](#_Toc161736950)

[4.x integration partner 5](#_Toc161736951)

[4.irrigation\_systemlorem ipsum 5](#_Toc161736952)

[4.8 measurement 5](#_Toc161736953)

[4.metadata lorem ipsum 5](#_Toc161736954)

[4.object lorem ipsum 5](#_Toc161736955)

[4.9 observation 5](#_Toc161736956)

[4.10 observation code 5](#_Toc161736957)

[4.observation\_code\_definition lorem ipsum 6](#_Toc161736958)

[4.11 observation collection 6](#_Toc161736959)

[4.observation\_configuration lorem ipsum 6](#_Toc161736960)

[4.12 observation procedure 6](#_Toc161736961)

[4.15 observed property 6](#_Toc161736962)

[4.observed\_quantity lorem ipsum 6](#_Toc161736963)

[4.13 observation result 6](#_Toc161736964)

[4.14 parameter 6](#_Toc161736965)

[4.16 property 7](#_Toc161736966)

[4.proximate\_feature\_of\_interest lorem ipsum 7](#_Toc161736967)

[4.17 quantity 7](#_Toc161736968)

[4.record lorem ipsum 7](#_Toc161736969)

[4.18 reference data 7](#_Toc161736970)

[4.samplelorem ipsum 7](#_Toc161736971)

[4.sample\_strategylorem ipsum 7](#_Toc161736972)

[4.scale (samplingPeriod) lorem ipsum 7](#_Toc161736973)

[4.19 semantic resource 7](#_Toc161736974)

[4.sensorlorem ipsum 7](#_Toc161736975)

[4.20 setup data 7](#_Toc161736976)

[4.systemlorem ipsum 8](#_Toc161736977)

[4.ultimate\_feature\_of\_interest lorem ipsum 8](#_Toc161736978)

[4.uniform\_resource\_identifier (URI) lorem ipsum 8](#_Toc161736979)

[4.universally\_unique\_identifier (UUID) lorem ipsum 8](#_Toc161736980)

[4.unique\_identifier lorem ipsum 8](#_Toc161736981)

[4.21 value 8](#_Toc161736982)

[5 Symbols (and abbreviated terms) 8](#_Toc161736983)

[6 Basic concepts about observations 9](#_Toc161736984)

[6.1 Fundamentals of observations 9](#_Toc161736985)

[6.2 Attributes of an ISO 19156 Observation 9](#_Toc161736986)

[6.2.1 phenomenonTime 9](#_Toc161736987)

[6.2.2 resultTime 9](#_Toc161736988)

[6.2.3 validTime 10](#_Toc161736989)

[6.2.4 parameter 10](#_Toc161736990)

[6.2.5 resultQuality 10](#_Toc161736991)

[6.3 Classes associated with an ISO 19156 Observation 10](#_Toc161736992)

[6.3.1 Result 10](#_Toc161736993)

[6.3.2 Feature of interest 10](#_Toc161736994)

[6.3.3 Observed property 10](#_Toc161736995)

[6.3.4 Procedure 10](#_Toc161736996)

[6.3.5 Metadata 11](#_Toc161736997)

[6.3.6 Observation context 11](#_Toc161736998)

[6.4 Simple Examples 12](#_Toc161736999)

[6.4.1 Example: Soil Moisture Measurements 12](#_Toc161737000)

[6.4.2 Example: Crop Scouting 13](#_Toc161737001)

[6.5 Guiding principles 14](#_Toc161737002)

[6.5.1 The preservation of meaning 14](#_Toc161737003)

[6.5.2 Minimizing redundancy 14](#_Toc161737004)

[7 Implementing observations and measurements in agriculture 17](#_Toc161737005)

[7.1 Class diagram for a simple agricultural implementation 17](#_Toc161737006)

[7.2 Observations Dataset 17](#_Toc161737007)

[7.3 Observation Collections 18](#_Toc161737008)

[7.4 Observations, Observation Codes and Observation Code Definitions 20](#_Toc161737009)

[7.5 Implementing the Result 21](#_Toc161737010)

[7.6 Implementing the Feature of Interest 21](#_Toc161737011)

[7.7 Implementing the Observed Property 21](#_Toc161737012)

[7.8 Implementing the Procedure 21](#_Toc161737013)

[7.9 Specifying units of measure 22](#_Toc161737014)

[7.10 Specialized observations 22](#_Toc161737015)

[7.11 Fundamental characteristics of sampling 22](#_Toc161737016)

[8 Semantics and Configuration 23](#_Toc161737017)

[8.1 Observation codes 23](#_Toc161737018)

[8.2 Code components and component codes 24](#_Toc161737019)

[8.3 Observation code definitions 25](#_Toc161737020)

[8.4 Observation component code definitions 26](#_Toc161737021)

[8.5 Observation Configuration 27](#_Toc161737022)

[8.5.2 Attributes of an ObservationConfiguration object used to *find* a desired Observation object: 28](#_Toc161737023)

[8.5.3 Metadata payloads and the mechanisms used to *contextualize and control* the payloads 29](#_Toc161737024)

[8.6 Making lists of semantic resources 31](#_Toc161737025)

[9 Reference and Setup Data 31](#_Toc161737026)

[9.1 Reference Data: DeviceModel and DeviceModelMetadataPayload 31](#_Toc161737027)

[9.1.1 DeviceModel 31](#_Toc161737028)

[9.1.2 DeviceModelMetadataPayload 32](#_Toc161737029)

[9.2 Setup Data: Device 33](#_Toc161737030)

[9.3 Setup Data: Locations / Gazetteer 34](#_Toc161737031)

[10 Examples of Encoded Meaning 35](#_Toc161737032)

[10.1 Introduction to the examples 35](#_Toc161737033)

[10.2 Basic examples 36](#_Toc161737034)

[10.3 Edge Cases 37](#_Toc161737035)

[11 Implementation notes 40](#_Toc161737036)

[11.1 Implementing Metadata at the Observation, Observation Collection, and Observation Dataset level 40](#_Toc161737037)

[11.1.1 Observation-level metadata 40](#_Toc161737038)

[11.1.2 Observation-collection-level metadata 40](#_Toc161737039)

[11.1.3 Dataset-level metadata 40](#_Toc161737040)

[Annex A (normative) Testing the implementation against ISO 19156 checklist 41](#_Toc161737041)

[A.1 General 41](#_Toc161737042)

[Annex B (informative) Sample Controlled Vocabularies for Code Components 48](#_Toc161737043)

[B.1 Controlled vocabulary for ObservationCodeComponent.componentType (Normative) 48](#_Toc161737044)

[B.2 Controlled vocabulary for ObservationCodeComponent.selector (Normative) 48](#_Toc161737045)

[Bibliography 49](#_Toc161737046)

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

Introduction

Agriculture, and agrifood systems in general, are under pressure to be more demonstrably principled and data-driven. Ever-growing pressure from regulators and business partners makes it increasingly necessary for crop and livestock producers, food, feed, fiber and fuel manufacturers, distributors and retailers, and the input providers for all of these stakeholders, to collect data that can be used to justify actions taken, drive decisions based on sound scientific principles, and maximize the efficiency of their business processes.

Observations and measurements (O&M) have a very important role in enabling a principled, data-driven approach to agrifood systems. Examples of O&M in agrifood systems include:

* Soil fertility laboratory measurements performed before planting.
  + Soil survey maps, which serve multiple agronomic purposes, can also be expressed as a collection of O&M data.
* Observations of soil moisture, temperature and trafficability, performed to determine if a crop can be planted.
  + Weather forecasts also represent a special case within the model of O&M.
* Once the crop is growing, fields are scouted for pests, diseases and signs of water-, heat-, or other abiotic stress.
* Determine the development stage of a growing crop, including whether it is ready to harvest.
* Post-harvest processes are also rich in O&M, such as the quality properties (e.g., moisture content, protein content, starch content, fiber length, percent broken grains) of a harvested commodity, and the environmental variables (e.g., temperature, humidity, oxygen and ethylene content) in the facilities and vehicles where a harvested commodity is stored, ripened, transported, and so forth.

Accurately and unambiguously representing these O&M in digital form is very important to enable data-driven, principled decision-making in the agrifood industry at scale. This capability has been advanced significantly with standards such as ISO 11783-10 and ISO 15143-3 for the subset of agricultural data produced by agricultural machinery such as planters, sprayers and harvesters, the domain of agrifood O&M as exemplified above is larger than what can be covered by those standards.

ISO has developed an abstract standard for representing O&M, ISO 19156. That standard, along with related ones such as ISO 19115 and ISO 19157 provides a comprehensive framework that systematizes the different aspects of O&M that must be represented to communicate accurately and unambiguously without loss of meaning. As described in ISO 19156 itself, however, additional work is necessary for the abstract concepts presented therein to be applicable to a specific domain (agrifood systems, in this case).

This standard ISO 7673-2 is an implementation standard for ISO 19156, providing a common data model and language that can enable integration of the disparate sources of O&M information typically used in agrifood systems.

Agricultural Irrigation — Part 2: Observations and Measurements

# 1 Scope

The purpose of this part of the standard is to enable the representation and exchange of observations and measurements (O&M) data (e.g., weather, soil moisture, crop scouting, crop quality measurements, food product properties), currently stored in a variety of proprietary original equipment manufacturer (OEM) formats, using an industry-wide data model and format that can be used by agrifood systems in general and digital agriculture (e.g, irrigation) data analysis and prescription-making programs in particular. This implementation is based on the ISO 19156 Standard for Observations and Measurements.

The standard also includes reference serialization schemas. The underlying intent is to serialize the object model, but not to prescribe a specific serialization technology.

The scope of this standard is primarily discrete observations of environmental (or other agrifood-related) variables along with the metadata, data quality elements and semantic resources necessary for their use in agrifood systems. This standard should enable a developer to represent observations and measurements accurately and unambiguously in terms of a set of objects adapted from the ISO 19156 standard.

## How the standard is structured

The specific material begins with a review of the ISO 19156 standard in clause 6, providing agrifood examples and guiding principles for what follows.

Clause 7 goes deeper into the specifics of the implementation, describing the basic classes used to exchange observations & measurements data.

Clause 8 describes how the meaning of an observation is represented in this standard and the semantic resources needed to do so at scale.

Clause 9 describes additional reference data used to describe equipment, as well as a quick review of master/setup data presented in Part 1 of this standard.

Clause 10 brings together the concepts described above using a set of examples.

Clause 11 provides additional implementation notes.

Annex A addresses conformance with the abstract test suite presented in Annex A of the ISO 19156 standard.

Annex B describes all the classes presented in the standard in detail.

Annex C presents a set of semantic resources that can be used to support implementation and will be used as the basis for subsequent standardization efforts.

## 1.2 Implementing ISO 19156 for agrifood systems

The ISO 19156 standard provides a fundamental, albeit abstract framework for representing observations and measurements. Implementing ISO 19156 in a specific domain involves adding the necessary detail to support the processes, and required data exchange, in that domain. This ISO 7673-2 standard provides the subset of concepts and elements from ISO 19156 employed to support activities in the agrifood (agriculture & food) sector. This includes data pertaining to weather and soil conditions, plant health, pest and disease presence, and so forth. A special emphasis is placed on the data needed to support data-rich irrigation activities. Note that "observations" of the application of irrigation water, i.e., those recording the irrigation activity itself, are documented in Part 3 of this standard.

# 2 Conformance

ISO 7673 defines an abstract object model. Applications intending to use the ISO 7673 object model will need an implementation of the object model that is appropriate for the particular application development platform. Future parts of the standard (or versions thereof) will include reference serialization implementations.

A particular implementation of the object model, whether it is for serialization or application development, would need to demonstrate some conformance to the business rules and constraints defined in 7673.

Conformance testing is essential for the expectation that two separate implementations can interoperate successfully. Given the large size and scope of ISO 7673, specification of conformance requirements and associated tests, will be covered in a separate part of the 7673 series.

# 3 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 6709: 2022. *Standard representation of geographic point location by coordinates*

ISO 8601-1:2019. *Date and Time. Representations for information interchange – Part 1: Basic Rules*

ISO 9834-8: 2014. *Information technology — Procedures for the operation of object identifier registration authorities — Part 8: Generation of universally unique identifiers (UUIDs) and their use in object identifiers*

ISO 19109: 2015. *Geographic information - Rules for Application Schema*

ISO 19112 :2019. *Geographic Information — Spatial referencing by geographic identifiers*

ISO 19156:2011. *Geographic information -- Observations and Measurements*

# 4 Terms and definitions

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

## 4.aggregation\_procedure (not ref.) aggregation procedure

lorem ipsum

## 4.alias alias

An identifier, not necessarily globally unique, but considered unique within the boundaries of a system (for example, a farmer’s farm management information system software), that is used to identify a resource (a device, a farm, a field, etc.) or a data record within that system.

Note: This standard seeks to associate globally unique identifiers to resources and data records, as one mechanism to eliminate ambiguity. Matching up aliases In incoming observations (4.X) with their corresponding unique identifiers is one of the challenges of processing observations data.

## 4.1 code component

A small unit of metadata, used as a building block for an observation code. Describes some aspect of an observation (e.g., observed property, feature of interest, observation procedure). Can also be used as a parameter of an observation.

## 4.1 code component definition

A small unit of metadata, used as a building block for an observation code. Describes some aspect of an observation (e.g., observed property, feature of interest, observation procedure). Can also be used as a parameter of an observation.

## 4.2 configuration data

the information that specifies the particular state of specific instances of things such as farm equipment and instruments (e.g. soil sensors, irrigation pivots, combines)

## 4.crop\_nutritionlorem ipsum

## 4.crop\_protectionlorem ipsum

## 4.data\_element lorem ipsum

## 4.data\_loggerlorem ipsum

## 4.data\_qualitylorem ipsum

## 4.data\_quality\_metric lorem ipsum

## 4.data\_stream lorem ipsum

## 4.data\_type lorem ipsum

## 4.3 dataset

identifiable collection of data. (ISO 19115-1:2014(en), clause 4.3)

Note1 to entry: Datasets are created for a specific use or purpose. As such, they are often attributed with metadata and data quality measures. Especially when produced as a result of a Data Product Specification.

## 4.device lorem ipsum

## 4.4 direct position

position described by a single set of coordinates within a coordinate reference system. (ISO 19129:2009(en), clause 4.1.11)

## 4.element lorem ipsum

## 4.enumeration\_item lorem ipsum

## 4.5 feature

abstraction of real-world phenomena. [ISO 19101:2002, clause 4.11] In this standard it will represent some part of the world that is the object of an observation: the **feature of interest**.

## 4.feature\_of\_interest lorem ipsum

## 4.gazetteer (is referenced to part 1) lorem ipsum

## 4.6 grower data

the information that represents basic information about the grower, farm, fields, and actors; typically synonymous with Setup data.

## 4.7 incoming observation

Data corresponding to an Obs object received by a processing system. That observation may be fully self-contained (i.e., not needing anything other than data type definitions for its meaning to be fully understood) or not. If the latter, it may be necessary to use Configuration objects to store the missing metadata so it can be added when data are received.

## 4.x integration partner

Lorem ipsum dolor

## 4.irrigation\_systemlorem ipsum

## 4.8 measurement

set of operations having the purpose of determining the value of a quantity (ISO/TS 19101-2:2018, clause 4.20)

## 4.metadata lorem ipsum

## 4.object lorem ipsum

## 4.9 observation

the act of measuring or otherwise determining the value of a property. [ ISO 19156:2011, 4.11 ]

## 4.10 observation code

This standard works under the assumption that at its simplest, and where spatial and temporal context are managed, an Observation is essentially a key-value pair. The observation code, i.e., the key in that simplified model, represents the meaning of the observation.

## 4.observation\_code\_definition lorem ipsum

## 4.11 observation collection

A data object that can group related observations and express the properties those observations have in common. This can be done for the purposes of facilitating search, or for reducing transmission bandwidth requirements (by removing the common properties from the child Observations and leaving them in the collection).

Note: Observation collections provide context to observations as per the ISO 19156 model.

## 4.observation\_configuration lorem ipsum

## 4.12 observation procedure

method, algorithm or instrument, or system of these, which may be used in making an observation [ISO 19156:2011, 4.12]

## 4.15 observed property

This is a property of the feature of interest wherein the process of observing the feature of interest results in a value that, generally speaking, quantifies (or categorizes) the property.

EXAMPLE: (from ISO 19156, 7.2.2.12): A feature type “Pallet” might be defined as having the property “mass” of type “Measure”. An observation providing the value of this property shall have observedProperty=“mass”, the result shall be of the type “Measure” and the scale (unit of measure) shall be suitable for mass measurements.

NOTE: While ISO 19156 does not define observed property as a term, the object model for Observation does include a property called “observedProperty”. 7673-2 uses the term “observed property” in a way consistent with the meaning of the property Observation::observedProperty described in 19156.

## 4.observed\_quantity lorem ipsum

## 4.13 observation result

estimate of the value of a property determined through a known observation procedure (From ISO 19156:2011, 4.14)

## 4.14 parameter

Optional key-value pairs that form part of an observation and help contextualize its result. Parameters may indicate the value of an environmental variable, a setting of an instrument, etc.

Example: The depth underground at which a soil water content observation is made.

## 4.16 property

facet or attribute of an object referenced by a name (ISO 19156:2011, 4.15)

## 4.proximate\_feature\_of\_interest lorem ipsum

## 4.17 quantity

property of a phenomenon, body, or substance, where the property has a magnitude that can be expressed as a number and a reference. (ISO/TS 19159-2:2016, clause 4.30)

NOTE A reference can be a measurement (**4.63**) unit, a measurement procedure, a reference material, or a combination of such.

## 4.record lorem ipsum

## 4.18 reference data

the information that a manufacturer makes available for the purchase, setup and/or use of their products.

## 4.samplelorem ipsum

## 4.sample\_strategylorem ipsum

## 4.scale (samplingPeriod) lorem ipsum

## 4.19 semantic resource

A data type definition, implemented as a Class, the instances of which exist in a semantic infrastructure. These definitions can be simple (e.g., describing the meaning of a temperature represented as a real value) or complex (describing an enumeration as part of a controlled vocabulary).

## 4.sensorlorem ipsum

## 4.20 setup data

the information needed to set up data exchange between the grower and machinery or other actors (e.g., crop advisors.) Unlike Reference Data, it is grower-specific, and includes grower data and configuration data.

## 4.systemlorem ipsum

## 4.ultimate\_feature\_of\_interest lorem ipsum

## 4.uniform\_resource\_identifier (URI) lorem ipsum

## 4.universally\_unique\_identifier (UUID) lorem ipsum

## 4.unique\_identifier lorem ipsum

## 4.21 value

element of a type domain [ISO 19156:2011, 4.18]

NOTE 1 A value considers a possible state of an object within a class or type (domain).

NOTE 2 A data value is an instance of a datatype, a value without identity.

NOTE 3 A value can use one of a variety of scales including nominal, ordinal, ratio and interval, spatial and temporal. Primitive datatypes can be combined to form aggregate datatypes with aggregate values, including vectors, tensors and images.

NOTE 4 This standard will represent values as strings, and use the information provided about the value's meaning (observed property, observation procedure, etc.) to help interpret that string.

NOTE 5 This initial version of the standard does not provide specific allowances for representing spatially-dense coverages (e.g. observations that can be expressed as gridded data) as anything other than key-value pairs associated with spatial extent.

# 5 Symbols (and abbreviated terms)

None

# 6 Basic concepts about observations

## 6.1 Fundamentals of observations

This ISO 7673-2 standard is derived directly from the object model defined in ISO 19156. The following clauses reiterate the salient components of the Observation and related classes and provide context for the implementation of 19156’s abstract model.

## 6.2 Attributes of an ISO 19156 Observation

The Observation class has five properties (shown inside the box representing the Observation class in the UML diagram of Figure 1):



Figure 1: Basic UML class diagram showing a simplified model of ISO 19156 Observations and measurements

### 6.2.1 phenomenonTime

This required property (term\_def 4.property) describes the time (instantaneous or a time interval) during which the result applies to the observed property (term\_def 4.15) of the feature of interest (term\_def 4.feature\_of\_interest); in other words, when the phenomenon or property was being sampled or observed.

Occasionally in agricultural applications it is necessary to use a dataset that does not originally have a clearly identified phenomenonTime. In these cases, implementors should use the best available estimate, and indicate the uncertainty using a data quality observation (term\_def 4.data\_quality) code component (term\_def 4.1) (8.2).

### 6.2.2 resultTime

This property describes the time (instant) when the result became available. This is typically when the procedure(s) associated with the observation (term\_def 4.12) are completed. In some observations this time is the same as the phenomenon time (e.g. when using a sensor). In other cases, the two can be very different, for example, in a laboratory test for soil organic matter, the phenomenonTime is when the soil sample was taken, while the resultTime is when the laboratory procedure was completed.

NOTE Although resultTime is required in ISO 19156, this standard allows it to be optional, assuming it to be the same as phenomenonTime if not specified.

### 6.2.3 validTime

This is an optional property. When present, it describes a time period during which the result is applicable for or intended to be used. A good use for this property is to specify the period of validity of a forecast.

### 6.2.4 parameter

An optional list of key-value pairs that can further describe the observation. Example: environmental parameters such as the depth at which a soil water content was measured. This standard implements parameters (term\_def 4.14) as *code components* (term\_def 4.1) (ObservationCodeComponent class, specialized key-value pairs described in clause 8.2) contained within the observation (term\_def 4.9).

### 6.2.5 resultQuality

An optional list of properties that indicates the quality of the observation. It allows for multiple measures and procedures for how quality is assessed, following procedures set forth in ISO 19157. This standard implements ISO 19157 data quality elements (term\_def 4.data\_quality\_element) as *code components* (ObservationCodeComponent class, specialized key-value pairs described in clause 8.2) contained within the observation, or as observations having other observations (term\_def 4.9), observation collections (term\_def 4.11) or datasets (term\_def 4.3) as their feature of interest (term\_def 4.feature\_of\_interest).

## 6.3 Classes associated with an ISO 19156 Observation

Observations are also associated with other objects. There are six fundamental classes of these objects:

### 6.3.1 Result

This holds the value (e.g., “25”) of the observed property (e.g., temperature), possibly along with a unit of measure (e.g., a code for degrees Celsius). The remaining classes below are meant to allow the receiver of the data to understand the **meaning** of the result.

### 6.3.2 Feature of interest

Represents the part of the world being observed. Can range from abstract (e.g., the atmosphere in the vicinity of a sensor) to something very specific (e.g. a particular soil core). Observations in the agricultural domain typically involve position information (e.g., geographical coordinates); however, in the ISO 19156 standard, the generic Observation class does not have an inherent position property. Relevant location information should be provided by the feature-of-interest, or by the observation process, according to the specific scenario.

### 6.3.3 Observed property

An Observed property is defined as a property of a feature of interest. In Object Oriented parlance, an Observed property is a property of an object, referenced by a name. (e.g., temperature, relative humidity).

### 6.3.4 Procedure

The ISO 19156 standard defines the observation process or procedure as the “method, algorithm or instrument, or system of these, which may be used in making an observation”. The class used for that purpose in ISO 19156, OM\_Process, is abstract, and has no properties. It serves as a base class that can be specialized to best fit the domain of specific implementations of the standard.

From ISO 19156 (7.2.3): The purpose of an observation process is to generate an observation result (term\_def 4.13). An instance of OM\_Process is often an instrument or sensor, but may be a human observer, a simulator, or a process or algorithm applied to more primitive results used as inputs.

Observations in agriculture frequently involve complex systems: one or more sensors may be producing a series of measurements, the results of which are combined, for example, to calculate evapotranspiration, and a set of these results are then aggregated in some way to produce a reported result such as total daily evapotranspiration. It is possible to build very complex descriptions of processes this way; this ISO 7673-2 standard provides three different kinds of procedures by default that will cover most applications for describing procedures used to make observations:

* **Observation procedure**: This refers to the method, algorithm, or instrument, or system of these, used to produce a single result. For example, in ***Figure 2*** a sensor (term\_def 4.sensor) such as a pyranometer converts incoming solar radiation to a voltage or other measurable signal, which in turn is represented by a number in a data logger (term\_ def 4.data\_logger) following an analog-to-digital conversion.
* **Sampling procedure**: Describes the mechanisms whereby a set of results (of the same observed property on the same feature of interest, using the same observation procedure) is collected into a sample, described by a sample distribution. For example, in ***Figure 2*** the Sampling procedure specifies that intermediate results of solar radiation will be collected every 10 minutes. This could take the form of the data logger collecting data at the specified interval and storing the results in a buffer. Sampling procedure can also involve things like the spatial distribution of sampling points.
* **Aggregation procedure**: Describes the statistic used to derive the reported result from the sample distribution. For example, in ***Figure 2*** the Aggregation procedure consists of applying an arithmetic mean every hour (to the sample of six values accumulated during that interval), thus obtaining the result, 123.

***Figure 2: Aspects of the Observation Procedure.***

This model is limited in its ability to represent very complex transformations. A comprehensive implementation of provenance can be achieved using a model such as the World-Wide Web Consortium’s PROV model (W3C, 2013).

### 6.3.5 Metadata

According to the World-Wide Web Consortium (2017), "*Metadata provides additional information that helps data consumers better understand the meaning of data, its structure, and to clarify other issues, such as rights and license terms, the organization that generated the data, data quality, data access methods and the update schedule of datasets. Publishers are encouraged to provide human-readable information in multiple languages, and, as much as possible, provide the information in the language(s) that the intended users will understand.*"

In Observations, metadata can be specified at the dataset (term\_def 4,3), record (term\_def 4.record), and property (term\_def 4.property) level. This will be discussed in detail in clause 11.1.

### 6.3.6 Observation context

Some observations are related and are thus best kept together; for example, a set of variables that are measured and communicated simultaneously and periodically by a weather station would be organized in a way that highlighted the relationship among them. The implementation of this idea through Observation Collections (term\_def 4.11) is explained in greater detail in clause 6.5.2.

## 6.4 Simple Examples

Table 1 shows a set of simple Observation examples, expressed initially in text form, followed by a tabular break-out in terms of the concepts described above.

### 6.4.1 Example: Soil Moisture Measurements

A sensor is placed underground at a certain depth. It measures volumetric soil water content and returns instantaneous values periodically.

Table 1 — Example of a soil water content Observation

|  |  |  |
| --- | --- | --- |
| **Property / element** | **Value** | **Comments** |
| phenomenonTime | 2023-04-14T04:00:04Z | This information should be represented in a standardized form (e.g. using ISO 8601 format) |
| resultTime | 2023-04-14T04:00:04Z | This information should be represented in a standardized form (e.g. using ISO 8601 format). Note how in the case of a sensor, the phenomenonTime and resultTime are typically assumed to be the same. |
| validTime | N/A | A validity interval is a good fit for forecasts (a valid form of Observation data, where a given forecast may be superseded by a more recent one), but not for measurements from instruments. |
| Parameter | Depth = 300 mm | The parameter entry should include:   * The data type / variable (i.e., depth underground), * The value of that variable, i.e., 300 * The unit of measure code (e.g., “mm”)   This ISO 7673-2 standard enables repre­senting parameters as *code components* (clause 8.2, 11.2). |
| resultQuality | This could involve a status / error code that captures problems with the sensor, logger, and communication, e.g., “OK” | ISO 19157 presents a framework of data quality elements. This ISO 7673-2 standard enables repre­senting data quality elements as *code components* (clause 8.2, 11.2) and observations (11.2). |
| Feature of interest (FOI) | Examples of applicable FOIs:   * FOI type = soil column * Latitude / longitude, or identifier of a known location | ISO 19156 states that observations are not always bound to a geographical position. This is applicable to agrifood data, e.g., protein content in a batch of feed, chemical analyses of incoming irrigation water, etc. |
| Observed property | Soil water content | Since agricultural soil water content data often involves gravimetric and volumetric measurements that are not comparable without additional information (e.g. bulk density), the observed property could be soil water content, or gravimetric/volumetric soil water content. |
| Procedure | Examples of procedures:   * Sensor manufacturer, model * The value is instantaneous | Clause 6.3.4 indicates that there are multiple possible types of procedures that it may be relevant to describe in an observation. |
| Metadata | Examples of metadata:   * Who installed the sensor * Sensor installation date | The ISO 19115 standard presents a metadata repre­sentation framework. This ISO 7673-2 standard enables representing metadata as *code components* (11.1). |
| Observation context | Related observations that are grouped together; e.g., same day. | Multiple soil water content measurements are typically made simultaneously at different depths. |
| Result | “42”, “prcnt” | The numeric result of the measurement must be associated with a unit of measure (in this case, a percent by volume), implemented with a code taken from a controlled vocabulary. |

### 6.4.2 Example: Crop Scouting

A crop scout is inspecting a field or cropzone (i.e., a part of a field), looking to determine the development stage of the corn crop planted there. The scout will make observations on multiple plants, the result of which is aggregated to produce the reported result.

Table 2 — Example of a field scouting Observation

|  |  |  |
| --- | --- | --- |
| **Property / element** | **Value** | **Comments** |
| phenomenonTime | 2023-04-14T02:32:04Z to  2023-04-14T02:35:00Z | This information should be stored in a standard­ized form (e.g., using ISO 8601 format) and represents the time interval the scout used to stage the multiple plants. |
| resultTime | 2023-04-14T02:35:00Z | Note how in this case, where the scout made the observation directly, the resultTime value matches that of the end of the phenomenonTime time interval. |
| validTime | 2023-04-14T02:32:04Z to  2023-04-15T02:32:04Z | In this case the scout is indicating that the state described by the observation is likely valid for 24 hours. |
| Parameter | N/A | N/A |
| resultQuality | “OK” if sample size was met | The ISO 19157 standard presents a framework of data quality elements. See 11.2. |
| Feature of interest (FOI) | Examples of applicable FOIs:   * A code indicating the crop that is being grown (e.g., “ZEAMX”) * Latitude / longitude * The field or cropzone identifier |  |
| Observed property | A code representing “Crop development stage”, e.g., “A\_CROP\_DEV\_STAGE” | Note: This can be encoded in the observationCode (7.2) |
| Procedure | Examples of procedures:   * 5 (Sample size) * PROC\_LEAF\_COLLAR (Staging, i.e., observation, method) * MEDIAN (Aggregation method) | Clause 6.3.4 indicates that there are multiple possible types of procedures may be relevantly described in an observation. The examples at left represent sampling method, observation method, and aggregation method:   * Sample size, i.e., how many plants were observed. * Staging method, i.e., how was the growth stage determined in each plant? (There are two most commonly used methods for corn, for example). * Aggregation method: How were the results of observing on samples aggregated to arrive at the reported result? |
| Metadata | Examples of metadata include the name of the scout and the scale used to encode results |  |
| Observation context | (See notes) | Multiple (in this example, 5) plants that are spatially near one another are observed and the result presented as a single Observation. This happens multiple times in the field; those Observations are reported together. |
| Result | “V2” | Phenological scales are typically discrete and may use alphanumerical codes such as “V2” or “R4” (e.g., Hanway, 1966) or numerical (e.g., BBCH, 2001). Note that despite BBCH using numeric codes, it is still a categorical scale, so a unit of measure is not needed. |

## 6.5 Guiding principles

### 6.5.1 The preservation of meaning

It is often the case when communicating observations data that the meaning of the data being transmitted is not conveyed along with the data. For example, it is typical for data to be exchanged using comma-separated-values (CSV) files, and all-too frequent for the CSV file headers to include a variable named, say, "T", without the consumer of the data knowing if the T stands for a temperature or something else; whether that temperature is an average value since the last measurement, or an instantaneous value, or something else; whether this temperature represents the air or the soil, and so forth. This lack of metadata (term\_def 4.metadata) restricts the usability of the dataset beyond the immediate people involved in making it, and beyond a relatively narrow time window following the creation of the dataset.

This standard introduces a system of *observation codes* (term\_def 4.10, 8.1) and their *code components* (term\_def 4.1, 8.2), meant to facilitate the sharing of meaning between the sender and receiver of observations data. Every code can have a persistent, unique identifier (term\_def 4.unique\_identifier), such as a uniform resource identifier (term\_def 4.uniform\_resource\_identifier, URI), that can be dereferenced on the receiving end to enable communication of all aspects of the observation (term\_def 4.9).

A desired outcome of the standard is for users to be able to unambiguously determine the meaning of a data file or contents of a data stream (term\_def 4.data\_steam).

### 6.5.2 Minimizing redundancy

The ambiguity of transmitted data elements (term\_def 4.data\_element) described above is often due to systems (term\_def 4.system) being designed for limited resources and/or low data rates. When a data logger (term\_def 4.data\_logger) has very limited memory and available transmission bandwidth, it is common to limit the metadata (term\_def 4.metadata) being stored / transmitted, and to establish some shared assumptions between the sender and receiver of the data. Although resources (on-board memory storage, communications bandwidth, per-byte transmission costs) available to in-field data loggers have improved significantly in recent years, this data standard acknowledges the continuing operation of, and need to accommodate, limited-resource platforms. This is done in part by encapsulating meanings as described previously, and also by organizing the data in a way that eliminates redundant information.

When a set of observations share a set of characteristics—for example, 100 data points coming simultaneously from a data logger that may share their result time, their location, references to the datalogger, grower/farm/field they correspond to, and so forth—it may not be practical to repeat the redundant information for each observation. This is particularly relevant in resource-limited scenarios, such as when a datalogger is sourcing data from the field through a costly or otherwise bandwidth-limited transport mechanism. This standard provides two different mechanisms for minimizing redundancy:

**Observation Configuration** (ObservationConfiguration type): Captures data that will *not* change over the scope of the document, such as location for a non-mobile sensor, sensor model, etc. This class will be described in detail in clause 8.5.

**Observation Collection** (ObsCollection type): Groups data that *may* change over the scope of the document, albeit *less frequently than the value of individual observations*. These collections of observations can be nested, with the collections bearing less-frequently-changing data containing one or more collections containing more-frequently-changing data or collections thereof. The innermost collections must contain one or more individual observations.

EXAMPLE 1 Corresponding to the data example from clause 6.4.1, this example involves a set of soil moisture data captured at two depths, in three locations in a single field or paddock, reported at two different datetimes (Figure 3). The dataset (4.3) is composed of three collections, each representing a location. Each one of these locations contains two collections, each representing a unique phenomenon time. These innermost collections have two observations corresponding to the different depths of interest, each with its particular set of metadata representing its meaning (including a parameter indicating depth) and its value.

In summary, the data for Example 1 are organized as follows:

Outermost level: ObsDataset

Outer collection: ObsCollection with same Location

Inner collection: ObsCollection with same phenomenonTime

Individual observations: Obs with a Setup Code, Value



***Figure 3: Example of organizing observations into collections. In this case there are two levels of collections: the outer level represents observation locations, and the inner level represents specific datetimes. Thus the observations contained in the innermost collections share the same location and phenomenonTime.***

EXAMPLE 2 In this example, corresponding to the one shown in clause 6.4.2, a crop scout is inspecting two cropzones over the course of three different days. The scout will make observations at two or more positions within each cropzone and will make two or more observations at each location (for example, crop development stage and signs of water stress). This example dataset is composed of three collections, each representing a date. Each one of these collections in turn contains two collections, each corresponding to a cropzone. These collections contain collections corresponding to two or more locations visited within the cropzone. Finally, these innermost collections contain the time-tagged individual observations (Figure 4).

Outermost level: ObsDataset

Outer collection: ObsCollection with same Date

Middle collection: ObsCollection with same Farm / Field / Cropzone

Inner collection: ObsCollection with same geographical direct position

Individual observations: Time, Setup Code, Value

In principle, the optimal arrangement of observations properties into collections will depend on the needs of particular users. A typical application may seek to minimize the size (in bytes, or of a proxy such as the number of nodes on the tree) of the dataset. For the purposes of storage and retrieval on the receiving end of the data transmission, implementations would likely “rehydrate” the original observations by adding back all the properties that were “factored out” of a collection.



***Figure 4: Example of organizing observations into collections. In this case there are three levels of collections: the outer level represents observation dates, the intermediate level represents cropzones, and the inner level represents specific sampling positions within that cropzone. Thus the observations contained in the innermost collections share the same date, cropzone, and geographical direct position.***

Collections are described in further detail in Clause 11.

# 7 Implementing observations and measurements in agriculture

## 7.1 Class diagram for a simple agricultural implementation

The following clauses present an implementation of ISO 19156 for the agrifood sector using a set of simple data objects. These objects are organized hierarchically, as shown in Figure 5 below.



Figure 5: UML Class diagram showing how the ISO 19156 model is implemented in this standard. Classes shaded in light blue represent reference data. The ones shaded in yellow are comments.

## 7.2 Observations Dataset

An Observations Dataset (also called by their class, *ObservationDataset*) is an overarching container of data that are related by a common purpose. Datasets can have metadata and data quality metrics (term\_def 4.data\_quality\_metric), expressed using code components (see term\_def 4.1 and Clause 8.2). Figure 6 below shows a more detailed class diagram.



Figure 6: UML class diagram of the ObservationDataset class.

The properties of ObservationDataset are:

**id**: string

The unique identifier of the ObservationConfiguration object. It is a string, such as a universally unique identifier (UUID, term\_def 4.universally\_unique\_identfier) or a Uniform Resource Identifier (URI, term\_def 4.uniform\_resource\_identifier).

**description**: string

A human-readable description of the ObservationConfiguration object.

**timeScopes [0..\*]**: TimeScope.

This array allows adding relevant time-related metadata to the Dataset object, including:

* Creation
* Start
* End
* Inspection
* Validity range

**observationCollectionRefs [0..\*]**: string

This array references the ObservationCollection instances that comprise the ObservationDataSet.

**codeComponents [0..\*]**: ObservationCodeComponents

An optional list of instances of the CodeComponent class.

**contextItems [0..\*]**: ContextItem

An optional list of instances of the ContextItem class. Used to represent geopolitical-context-dependent data. Please refer to Part 1 of this standard for more detail.

**notes [0..\*]**: String

An optional list of instances of the ContextItem class. Used to represent geopolitical-context-dependent data. Please refer to Part 1 of this standard for more detail.

## 7.3 Observation Collections

Observation collections (also called *ObservationCollections*) are composed of related observations (or child ObservationCollections) that share something in common or are grouped for some purpose. These have been previously introduced in clause 6.4 and are described in further detail in clause 11.

Figure 7 shows the properties of an ObservationCollection:

The properties of ObservationDataset are:

**id**: CompoundIdentifier

The unique identifier of the ObservationConfiguration object. It is an instance of the CompoundIdentifier class (See Part 1 of this standard for a detailed description) and includes a reference identifier string that is used to reference the object in other objects, and a set of UniqueIds or aliases that may be used to refer to this object in other systems.

**observationConfigurationCode** [0..1]: Code

lorem ipsum dolor.

**observationCode** [0..1]: Code

lorem ipsum dolor.

**deviceRef** [0..1]: String

lorem ipsum dolor.

**spatialExtent** [0..1]: String. This enables representing a particular aspect of the feature of interest, the spatial position or footprint shared by the Observations in the ObservationCollection. The data type is an ISO 19107 GeometryType (e.g., a Point, a Line, a Polygon, a MultiPolygon, etc.). The chosen serialization should use an accepted industry standard such as WKT.

**timeScopes [0..\*]**: TimeScope.

This array allows adding relevant time-related metadata to the Dataset object, including:

* Creation
* Start
* End
* Inspection
* Validity range

**observationCollectionRefs [0..\*]**: string

This array references the ObservationCollection instances that comprise the ObservationDataSet.

**observationRefs [0..\*]**: string

This array references the ObservationCollection instances that comprise the ObservationDataSet.

**codeComponents [0..\*]**: ObservationCodeComponents

An optional list of instances of the CodeComponent class.

**contextItems [0..\*]**: ContextItem

An optional list of instances of the ContextItem class. Used to represent geopolitical-context-dependent data. Please refer to Part 1 of this standard for more detail.

**valueUoM** [0..1]]: Code. It is common to receive measurement (term\_def 4.8) (i.e., observations with a numeric result) data that do not explicitly include unit of measure codes (e.g., a temperature column in a CSV data file, which includes numbers, but no further indication of whether the data are expressed in degrees Celsius, Fahrenheit or Kelvin). Including a valueUoM code in the

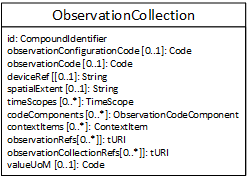


Figure : The ObservationCollection class. Note that its properties are very similar to those of the Observation, except for the absence of a value property, inherent to the Observation, and references to child Observation or ObservationCollection objects.

## 7.4 Observations, Observation Codes and Observation Code Definitions

Figure 6 below shows a class diagram for the Observation Class.



Figure 8: UML class diagram of the Observation class.

In this standard Observations (class Observation) typically have an *observationCode* property, a token (string with no leading or trailing spaces) that represents part or all of the meaning of the result, for example, a code to represent air temperature, or a code to represent volumetric soil water content, or the count of pest species observed. In other words, it is a code that serves as a proxy for a combination of feature of interest, observed property, procedure, and even optionally the unit of measure of the result, but does not include the result value.

The observationCode can have a corresponding semantic resource (term\_def 4.19), a reference data (term\_def 4.18) object called an *observation code definition* (class ObservationCodeDefinition, see Figure 5*)*, that can contain a machine-actionable description of the combination of elements (term\_def 4.element that the observationCode serves as a proxy for.

For example, an observationCode representing mean annual air temperature measured with a weather station sensor at a height of 2 meters and with a value expressed in degrees Celsius could have a corresponding ObservationCodeDefinition including:

* Feature of interest, expressed as an abstraction, i.e., representing “air”.
* Observed property, such as “temperature”
* Observation method: sensor
* Sampling method: Unspecified
* Aggregation method / time window: One year
* Aggregation method / statistic: mean
* Parameter: Height of 2 meters
* Default unit of measure: C (degrees Celsius)

## 7.5 Implementing the Result

As noted previously, observations have a result with which they have a compositional relationship, i.e., the Result is fully owned by and contained within the Observation object.

NOTE 1 This standard implements the value as a property of the result distinct from the code that represents its unit of measure (clause 7.3). It also allows optionally specifying the unit of measure as part of an observation code definition; this is covered in detail in clause 8.3.

NOTE 2 This standard implements the result *by value*. This means that the result does not exist independently of the Observation object and is contained within it.

NOTE 3 In this standard, result values are always encoded as strings. The underlying (simple) data type of the value encoded in that string is declared in an object called the *observation code definition* (class ObservationCodeDefinition, clause 8.3).

## 7.6 Implementing the Feature of Interest

This standard provides four ways of expressing the feature of interest (6.3.2) of an observation:

* **implicitly**, by referencing abstract features of interest (e.g., “air”, “soil”) within an observation code (see clause 8.2 for a description of how abstract features of interest are represented in observation code definitions using code components).
* **explicitly, by value**, by adding the spatial scope to the observation in the form of a position (i.e., latitude / longitude / elevation), linestring or multipolygon.
* **explicitly, by reference**, by adding a reference to the grower, farm, field, cropzone, device, sample, specimen, etc. to the observation.
* **implicitly, by code**: by referencing abstract features of interest or modifiers thereto (e.g., “greenhouse” as a modifier to “air”) using *code components* (ObservationCodeComponents class) added to the observation. (See clause 8.2 for a description of how abstract features of interest can be added to observations using code components)

## 7.7 Implementing the Observed Property

This standard provides two ways of expressing the observed property (6.3.3) of an observation:

* **implicitly**, by referencing observed properties (e.g., “temperature”, “species presence”, “number of kernels”) **within an observation code definition** (see clause 8.2 for a description of how observed properties are represented in observation code definitions using code components).
* **Implicitly, on the observation:** by referencing observed properties(e.g., a pest being scouted for) using code components (ObservationCodeComponent class) added to the observation. (See clause 8.2)

## 7.8 Implementing the Procedure

This standard provides three ways of expressing the procedure (6.3.4) of an observation:

* **implicitly**, by referencing observation, aggregation, sampling or other custom methods **within an observation code definition** (See clause 8.2 for a description of how methods are represented in observation code definitions using code components).
* **implicitly, with a code on the observation**, by referencing observation, aggregation, sampling or other custom methods using code components (ObservationCodeComponent class) added to the observation. (See clause 8.2).
* **explicitly, with a code referencing a method object:** by referencing external method description objects (the definition of which is outside the scope of this version of the standard), e.g., a particular laboratory test used to produce an estimate, using code components (ObservationCodeComponent class) added to the observation. (See clause 8.2)

## 7.9 Specifying units of measure

Including a code to represent the unit of measure (when applicable) of an Observation’s value property reduces ambiguity and thus helps preserve the meaning of the data. The code can be included explicitly as a property of the Observation (see Annex B) or included as part of the ObservationCodeDefinition (and thus, be included implicitly in the observation). The choice of the approach to follow is dependent on the complexity of the underlying application.

## 7.10 Specialized observations

When thinking about observations and measurements, it’s common to imagine a result with a simple data type (term\_def 4.data\_type) such as an integer (e.g., number of insects observed on a plant leaf), a real number (e.g., volumetric soil water content), or even a categorical or enumerated value (term\_def 4.21) (e.g. “SOUTH” for an approximate wind direction, or “MODERATE” for a water stress severity rating). There are, however, several other possible data types that are described in ISO 19156 and are relevant to agriculture, such as a polygon to describe a wet area in a field, or an image file representing the input to a weed detection model. Annex A describes how this standard implements the different result data types presented in ISO 19156.

## 7.11 Fundamental characteristics of sampling

The ISO 19156 standard distinguishes between an ultimate feature of interest (def\_term 4.ultimate\_feature\_of\_interest) that may be impractical to make direct observations on (e.g., an agricultural field or paddock that we want to make fertility management decisions about) and a proximate feature of interest (term\_df 4.proximate\_feature\_of\_interest (e.g., a soil core) that has readily observable properties (e.g., concentration of nutrients of interest). Properly documenting the context in which those proximate features of interest are obtained, i.e., documenting the sampling strategy (term\_def 4.sample\_strategy) and metadata (term\_def 4.metadata) regarding the sample (term\_def 4.sample) itself, is an important part of maximizing data quality (term\_def 4.data\_quality) and the usefulness of the observations data associated with the samples.

This standard supports multiple ways of specifying ultimate and proximate features of interest (including samples), as explained in clause 6.3.2. It also enables specifying procedures used in sampling (6.3.4).

# 8 Semantics and Configuration

The ISO 19156 standard is very flexible in its description of how to specify features of interest, observed properties, and so forth. This flexibility responds to the differences among disciplines regarding how observations are represented.

This ISO 7673-2 standard seeks to implement the semantics (i.e., aspects related to meaning) of an agricultural and food systems observations & measurements model using controlled vocabularies wherever possible.

Moreover, this standard implements different levels of granularity with which it enables specifying meaning:

* At its simplest, the meaning of an observation (see Figure 6) can be captured with a single controlled vocabulary element, the observationCode.
* A next level of complexity is represented by the observationCode with some additional meaning being provided by code components (8.2, below).
* A third method is to use a single *observation configuration code*, or obsConfigCode, to reference a packaging together of an observationCode and multiple code components, the observation configuration object (8.5, below). This can be combined with code components on the observation itself.
* At its most granular, an observation may use only codeComponents to specify its meaning, with neither an observationCode nor an observationConfigurationCode involved.

Note: The references made above to using controlled vocabularies to represent semantics reflect the current, less-than-mature state of data exchange in agrifood systems. There exist more sophisticated mechanisms for representing concepts and the relationships among them; namely, ontologies. While this standard does not represent semantic resources using ontologies, it does allow for each semantic resource (4.19), e.g., a data type definition such as the crop development stage variable of the example shown in 6.4.2, or an enumerated value within such as the enumerated value corresponding to flowering or harvest maturity in that same example) to have a persistent, unique identifier that can be used within an ontology.

## 8.1 Observation codes

As mentioned above, the observationCode property of an Observation encodes all or part of its meaning.

It’s important to note that the amount of meaning thus encoded is variable and application-dependent. See clause 8.1 Observation codes for elaboration of how this variability affects documents.

As mentioned in 7.2 above, the observationCode property of an Observation encodes all or part of its meaning.

It’s important to note that the amount of meaning thus encoded is variable, and application dependent. For example, assume an irrigation application where soil water is measured at three different depths, such as 30, 60 and 90 cm. This scenario can be modelled in different ways, for example:

* Three different observation codes (term\_def 4.10), one per depth. The codes could look like:
  + E\_SOIL\_WATER\_CONTENT\_30
  + E\_SOIL\_WATER\_CONTENT\_60
  + E\_SOIL\_WATER\_CONTENT\_90
* A single observation code, with a parameter (term\_def 4.14) added to specify the depth. The code could look like:
  + E\_SOIL\_WATER\_CONTENT

Which model is best will depend on the application: in the above example, using a parameter independently of the observationCode allows for more flexibility, but the data payloads are larger, and querying becomes more complex.

## 8.2 Code components and component codes

The smallest particle of meaning in the system (term\_def 4.system) is the code component (class ObservationCodeComponent). This object enables specifying particles of meaning such as features of interest (term\_def 4.feature\_of\_interest), observed properties (term\_def 4.15), parameters (term\_def 4.14), and so forth within Observation, ObservationCollection and ObservationDataset objects. This will be explained in greater detail in clause 8.2.

Note that analogously to how an ObservationCode can have an ObservationCodeDefinition that it can reference, an ObservationCodeComponent can reference a corresponding ObservationCodeComponentDefinition.

-----------------------------------

Figure 7 shows the properties of the ObservationCodeComponent class. This class is a mechanism for representing small units of meaning associated with an Observation, ObservationCollection, or even ObservationDataset. At its most basic, it is a key-value pair, with the componentCode property as the key, and the value and its unit of measure (when needed) expressing the rest of the information.

The componentType and selector properties serve as a multi-level key which, together, mean the same as the componentCode property, albeit broken out in a way that may simplify searching within systems, populating drop-down lists in user interfaces, etc.:

* componentType corresponds to the basic hierarchy of components of meaning within the system (feature of interest, observed property, procedure, metadata, data quality, parameter, etc.) It should be used as an enumerated variable, with its values taken from a code list (See Annex C.1).
* selector is also meant to have an enumerated value, taken from a controlled vocabulary specific to the componentType.

The description is an optional, human-friendly property that enables adding text information that can help interpret the code component.

The valueType property is optional, and only relevant in situations where a code component is being used only by value (i.e., either without the componentCode, or with a proprietary componentCode that a receiving individual may not recognize) and indicates the basic underlying data type of the code component’s value (e.g., a unit of distance for a depth parameter, a unit of time for the duration of an aggregation window, etc.). In situations where the code component is used by reference, only the componentCode is needed, but there must be a corresponding ObservationCodeComponentDefinition object available to the consumer of the data, to provide the necessary description of the meaning of the code component.

In summary, there are two distinct use cases for code components, as summarized in Table 3.

Table 3 — Use cases for CodeComponents

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Use case** | **componentCode** | **componentType** **& selector** | **valueUoM** | **valueType** | **Context** |
| Implicit, by reference (Default) | Yes | Optional (redundant) | Optional (may be redundant) | No | Semantically rich: an ObservationCodeComponentDefinition is available to the data consumer. |
| Explicit, by value | Optional | Yes | Yes | Yes | Semantically poor: The consumer of the data must be able to recognize the componentCode and/or the componentType / selector. |

Summarizing the above, ObservationCodeComponent objects provide additional meaning to Obs, ObsCollection and ObsDataset objects. They can do that explicitly and by value, packaging all the known additional information (componentType + selector + value + valueUoM + valueType), or they can do that implicitly and by reference, using the CodeComponent as a key-value pair that communicates only the necessary information, because the data consumer has the corresponding ObservationCodeComponentDefinition available.



Figure 9: UML class diagram of the ObservationCodeComponent class.

## 8.3 Observation code definitions

Observation codes as discussed in 8.1 are merely an identifier, a shorthand way of referring to an object called an observation code definition (term\_def 4.observation\_code\_definition (Class ObservationCodeDefinition) shown in Figure 9.



Figure 10: Observation Code Definition, shown in the form of a UML class diagram.

The ObservationCodeDefinition class has the following properties:

* **observationCode [0..1]**: Code. This enables explicitly representing the meaning of an Observation (term\_def 4.9) that is received bearing only an alias (term\_def 4.alias) (e.g., a channel number) that does not represent its meaning. Once the ObservationConfiguration object corresponding to that Observation is found, the observationCode in the ObservationCOnfigurationPayload
* **pId**: String.
* **description**: String.
* **observationCodeComponentCodes [0..\*]**: Code.
* **nestedDefinitions** [0..1]]: Code. It is common to receive measurement (term\_def 4.8) (i.e., observations with a numeric result) data that do not explicitly include unit of measure codes (e.g., a temperature column in a CSV data file, which includes numbers, but no further indication of whether the data are expressed in degrees Celsius, Fahrenheit or Kelvin). Including a valueUoM code in the ObservationConfigurationPayload enables including the unit of measure code into the Observation.
* **valueType** [0..1]]: Code. It is common to receive measurement (term\_def 4.8) (i.e., observations with a numeric result) data that do not explicitly include unit of measure codes (e.g., a temperature column in a CSV data file, which includes numbers, but no further indication of whether the data are expressed in degrees Celsius, Fahrenheit or Kelvin). Including a valueUoM code in the ObservationConfigurationPayload enables including the unit of measure code into the Observation.
* **valueUoM** [0..1]]: Code. It is common to receive measurement (term\_def 4.8) (i.e., observations with a numeric result) data that do not explicitly include unit of measure codes (e.g., a temperature column in a CSV data file, which includes numbers, but no further indication of whether the data are expressed in degrees Celsius, Fahrenheit or Kelvin). Including a valueUoM code in the ObservationConfigurationPayload enables including the unit of measure code into the Observation.
* **valueUoM** [0..1]]: Code. It is common to receive measurement (term\_def 4.8) (i.e., observations with a numeric result) data that do not explicitly include unit of measure codes (e.g., a temperature column in a CSV data file, which includes numbers, but no further indication of whether the data are expressed in degrees Celsius, Fahrenheit or Kelvin). Including a valueUoM code in the ObservationConfigurationPayload enables including the unit of measure code into the Observation.
* **valueUoM** [0..1]]: Code. It is common to receive measurement (term\_def 4.8) (i.e., observations with a numeric result) data that do not explicitly include unit of measure codes (e.g., a temperature column in a CSV data file, which includes numbers, but no further indication of whether the data are expressed in degrees Celsius, Fahrenheit or Kelvin). Including a valueUoM code in the ObservationConfigurationPayload enables including the unit of measure code into the Observation.
* **codeComponents** [0..1]: Code. This is where the semantic “heavy lifting” can occur, since multiple CodeComponents (bearing information regarding feature of interest, observed property, observation, sampling and aggregation methods, data quality, etc.) can be included.
* **spatialExtent** [0..1]: String. This enables representing a particular aspect of the feature of interest, the spatial position or footprint of the Observation, using position information that has been captured *a priori*. The data type is an ISO 19107 GeometryType (e.g., a Line, a Polygon, a MultiPolygon, etc.). The chosen serialization should use an accepted industry standard such as WKT.

## 8.4 Observation component code definitions

Observation code components as discussed in 8.2 can be used by reference, by referring to an object called an observation code component definition (Class ObservationCodeComponentDefinition), shown in Figure 9.



Figure 11: UML class diagram showing the ObservationCodeComponentDefinition class, along with its relationship to ObservationCodeComponent.

## 8.5 Observation Configuration

While it sometimes happens that an incoming observation (4.7) is completely self-contained and is received containing all necessary metadata, it is very frequent to receive observations that are somehow incomplete. Examples include:

* Observations originating in a device (term\_def 4.device) that is not aware of its feature of interest, e.g.,position (i.e., latitude, longitude). Position information must be added for the Observation to be complete.
* Observations that originate in a device that is not aware of the property being observed. An example is a data logger (term\_def 4.data\_logger) that can transmit a channel-value pair but cannot communicate what property is being measured on that channel number.

In these cases, it is convenient to define a set of metadata, the *ObservationConfigurationPayload*, that should be added to the Observations originating in that device.

Figure 10 below shows the definition of the ObservationConfigurationPayload class. It represents the metadata that should be inserted into incoming Observation objects.



Figure 12: The ObservationConfigurationPayload class. Note how all the properties are optional. Note also how, since it is used by value in configuration objects, the ObservationConfigurationPayload does not need its own unique identifier.

The ObservationConfigurationPayload object has the following properties, which are all optional:

* **observationCode [0..1]**: Code. This enables explicitly representing the meaning of an Observation (term\_def 4.9) that is received bearing only an alias (term\_def 4.alias) (e.g., a channel number) that does not represent its meaning. Once the ObservationConfiguration object corresponding to that Observation is found, the observationCode in the ObservationCOnfigurationPayload
* **valueUoM** [0..1]]: Code. It is common to receive measurement (term\_def 4.8) (i.e., observations with a numeric result) data that do not explicitly include unit of measure codes (e.g., a temperature column in a CSV data file, which includes numbers, but no further indication of whether the data are expressed in degrees Celsius, Fahrenheit or Kelvin). Including a valueUoM code in the ObservationConfigurationPayload enables including the unit of measure code into the Observation.
* **codeComponents** [0..1]: Code. This is where the semantic “heavy lifting” can occur, since multiple CodeComponents (bearing information regarding feature of interest, observed property, observation, sampling and aggregation methods, data quality, etc.) can be included.
* **spatialExtent** [0..1]: String. This enables representing a particular aspect of the feature of interest, the spatial position or footprint of the Observation, using position information that has been captured *a priori*. The data type is an ISO 19107 GeometryType (e.g., a Line, a Polygon, a MultiPolygon, etc.). The chosen serialization should use an accepted industry standard such as WKT.

An important point about these properties is that these metadata must be known *a priori* (e.g., through field capture when installing the device) and represented unambiguously.

In summary, the ObservationConfigurationPayload class shown in Figure 10 above represents the kind of information that can enrich an incoming observation (term\_def 4.7).

The ObservationConfiguration class, shown in Figure 11, enables a) finding and b) contextualizing and controlling a configuration payload. Broadly, its properties (with the shown cardinalities and data types) are as follows:



Figure 13: ObservationConfiguration class. This object can be interpreted as one that, in the context of an incoming observation, helps find, contextualize and/or control the ObservationConfigurationPayload that must be added into the Observation to make it complete.

### 8.5.2 Attributes of an ObservationConfiguration object used to *find* a desired Observation object:

**observationConfigurationCode** [0..1]: This optional token, assumed to come from a controlled vocabulary, serves as a proxy / alias for the ObservationConfiguration object’s identifier. The use case for this primarily involves metadata-poor situations.

**barcodeId** [0..1]: string. This may be a machine-readable (e.g., through a barcode or RFID scanner) code that identifies the device (term\_def 4.device) in the customer’s system, and which is presumed to be available in the field so it can be scanned to enable electronic in-field metadata (term\_def 4.metadata) capture.

**deviceRef**: The unique identifier of the device (e.g., a sensor) that is producing the observation in question.

**deviceReportingForRefs**: This identifies devices that observations represented by this ObservationConfiguration object could be considered as reporting for. Examples:

* An organization wishes to track the lifecycle of a particular temperature & relative humidity sensor so as to replace it before its performance degrades, to be able to manage any manufacturer- or batch-specific problems, etc. Moreover, the organization wishes to associate incoming observations to that device, but those incoming observations are abstracted away from the originating device by the integration partner (4.IntegrationPartner), and reported as originating in another device (e.g., a weather station console).
* An organization has a weather station that consists of a gateway, a main unit, a number of nodes linked wirelessly to that unit, and various sensors connected to each of the above. The data (i.e., Obs and ObsCollection objects) are associated (via the deviceRef property) with the device that originated them, but a user wishes to query by a more abstract concept, the “weather station”. In this case a virtual device can be created to represent the idea of the weather station, and all the individual sensors can be configured so they contain the reference to the virtual device in their deviceReportingForRefs list, thus making it simple to find all the observations associated with the station.

**parentObservationConfigurationRef** [0..1]: This property allows identifying the parent device or observation source when a system is organized hierarchically (e.g., sensors that report to nodes, that report to a central unit). Note how the reference is to an ObservationConfiguration object, and not to a Device object.

**reportsAsParentAtSource** [0..1]: Boolean. This property indicates whether the data originating in this device is being reported by the integration partner as reported by the device’s parent. Note: This forces the idea of a parent-child relationship to represent the data flow. This was the original idea, but it is not entirely clear that this property the best way to express this phenomenon. The purpose of this information is to enable a receiving system to allocate data to the correct sensor in situations where, for example to track device maintenance, there is a desire to **COMPLETE**

**isSensor**: Boolean. This property specifies whether the device of interest is acting as a sensor. Note that this and its sibling properties below are communicated at the configuration level instead of the device level because a device may sometimes perform a certain function or not (e.g., repeater) depending on how the system is configured in the field.

**isLogger**: Boolean. This property specifies whether the device of interest is acting as a data logger. Note that this and its sibling properties re communicated at the configuration level instead of the device level because a device may sometimes perform a certain function or not (e.g., repeater) depending on how the system is configured in the field.

**isGateway**: Boolean. This property specifies whether the device of interest is acting as a communication gateway. Note that this and its sibling properties re communicated at the configuration level instead of the device level because a device may sometimes perform a certain function or not (e.g., repeater) depending on how the system is configured in the field.

**isEdgeProcessor**: Boolean. This property specifies whether the device of interest is performing edge processing functions in the field. Note that this and its sibling properties re communicated at the configuration level instead of the device level because a device may sometimes perform a certain function or not (e.g., repeater) depending on how the system is configured in the field.

**isRepeater**: Boolean. This property specifies whether the device of interest is acting as a communications repeater. Note that this and its sibling properties re communicated at the configuration level instead of the device level because a device may sometimes perform a certain function or not (e.g., repeater) depending on how the system is configured in the field.

### 8.5.3 Metadata payloads and the mechanisms used to *contextualize and control* the payloads

#### discardData: Boolean. This property is a mechanism for specifying data that should not be kept / persisted. For example, if an 8-channel data logger has been set up in the field with sensors attached to only 6 of its channels and reports data from all 8, an ObservationConfiguration object could be instantiated for each of the two undesired channels with discardData set to true. This would indicate to a farm management information system to not store data being reported for those two channels.

**overwriteChildMetadata**: Boolean. Specifies that an incoming Observation object’s metadata must be overwritten by the corresponding ObservationConfiguration object’s metadata.  
  
This standard enables metadata to be specified at different levels. For example, metadata such as a unit of measure can be specified at the ObservationConfiguration level but can also be specified directly in the incoming observation.  
  
In general, in a situation like this where a “parent” (the device’s configuration object) and a “child” (the lower-level entity; e.g., an Observation) have conflicting metadata, the “child“ prevails. This behavior can be reversed using the overwriteChildMetadata property: when set to true, the incoming Observation’s metadata will be replaced with the “parent’s” metadata. A good use case for this is a sensor that is installed in a field with an onboard GNSS receiver that allows it to report its direct position; sometimes these units can have problems due to poor reception, limited number of satellites, etc. and report erroneous direct position values. In these situations a trustworthy direct position can be captured in the field when the unit is installed and included in the ObservationConfigurationPayload of an ObservationConfiguration with overWriteChildMetadata set to true, in which case the spatialExtent of the Observations will arise from the static values stored in the ObservationConfiguration object.

**autoCalcReportingInterval**: Boolean. Indicates whether the device or Observations source should have its time aggregation window calculated. Many sensors produce values at fixed time intervals, whether because their analog signals are sampled and converted to digital values on a schedule that can be set on the data logger in the field, or because they are “smart” sensors that contain the electronics necessary to digitize the value of their signal and subsequently push it to the cloud on a regular schedule. There are other sensors, however, that create observations on an irregular schedule, such as rain gauges that produce observations when they accumulate a set amount of rainfall (e.g., 0.1 mm). This can create problems in implementations that associate aggregation time CodeComponent objects to Observations, because the value of the aggregation time window is not known a priori. This property, when set to true, indicates that the device or source of Observations being configured is operating under a variable schedule and that the aggregation interval should be calculated based on the time between successive Observations’ phenomenon times, and not set a priori using a CodeComponent in a ObservationConfigurationPayload object.

#### 8.5.4 Other

**id**: string

The unique identifier of the ObservationConfiguration object. It is a string, such as a universally unique identifier (UUID) or a Uniform Resource Identifier (URI).

**description**: string

A human-readable description of the ObservationConfiguration object.

**timeScopes**: TimeScope [0..\*]

This array allows adding relevant time-related metadata to the Configuration object, including:

* Inspection
* Maintenance
* Decomissioning
* Validity range

The validity range in particular is very important. It is covered in detail in the following clause.

#### 8.5.5 Validity range of an ObservationConfiguration object

The validity range of an ObservationConfiguration object is a half-open interval of datetime values used to specify the range of datetimes in which the object accurately describes the configuration of a device or other observation source (as shown in clause 9.3 of Part 1 of this standard).

The interval is half-open because in that way it enables an unbroken, unambiguous timeline spanning multiple, temporally adjacent ObservationConfiguration objects.

The meaning of the two timestamps in a aTimeScope with its context set to VALIDITY\_RANGE follows.

**start**: DateTime. This required property specifies the datetime when an ObservationConfiguration object becomes valid. It is a closed interval, in that the ObservationConfiguration is valid for values of datetime greater than or equal to the specified validityStart.

**end** [0..1]: DateTime. This optional property specifies when an ObservationConfiguration object ceases to be valid, typically because there is a new one that describes reality (e.g., a new sensor, or a new position in space) from that moment onward. The validityEnd is meant to be an open interval, in that validity is expected to be for date values less than the specified validityRange.

## 8.6 Making lists of semantic resources

It is often convenient to make lists of valid values for observation codes (term\_def 4.10), enumeration items (term\_def 4.enumeration\_item), code components (term\_def 4.1), and their parts. Supporting this from a normative data model is outside the scope of the initial version of this standard.

# 9 Reference and Setup Data

Reference data, described in Part 1 of this standard, refers to data that describes all instances of a thing, such as a specific chemical that is applied to a crop, or an identifier for the crop itself. This part of the standard introduces a new reference data class, the DeviceModel, and another class that supports it, the DeviceModelPayload, both described in Clause 9.1 (Figure 13).

Setup data describes infrequently changing, generally producer-specific information about instances of things, such as farms and fields and people, as described in Part 1 of this standard. This part of the standard introduces a new setup data class, the Device (term\_def 4.20).

## 9.1 Reference Data: DeviceModel and DeviceModelMetadataPayload

### 9.1.1 DeviceModel

The DeviceModel class (Figure 13) represents the properties shared by all instances of a given kind of device (e.g., a sensor).



Figure 14: UML class diagram of the DeviceMod9el class.

The properties of DeviceModel are:

**id**: CompoundIdentifier. This property contains a unique identifier for the instance of DeviceModel and can also include an additional set of aliases (4.1), i.e. an additional set of unique identifiers used in external systems to represent the same idea (See Part 1 of this standard for a detailed explanation of the CompoundIdentifier class).

**description** [0..1]: An optional human-readable description of the device model being represented; e.g., “Ultra Max 5000 Soil Water Content Sensor”.

**defaultDeviceModelMetadataPayloads [0..\*]**: DeviceModelMetadataPayload. This is a list of references to DeviceModelMetadataPayload objects (see Figure 14 below) that can provide default metadata for one or more properties observed by the device. This property is a list because there are sensors (e.g., temperature and humidity sensors) that make observations for more than one observed property.

**manufacturerRef [0..1]**: String. This is a reference to an instance of the Manufacturer class (See Part 1 of this standard) and is used to identify the manufacturer of a particular DeviceModel instance. It should match the value of the referenceIdentifier property of the CompoundIdentifier id of the Manufacturer instance.

**Brand [0..1]**: String. This is a human-readable string containing the brand name (e.g., “Super Ultra Sensors”) of the DeviceModel instance.

**Series [0..1]**: String. This is a human-readable string containing the series name (e.g., “4000”) of the DeviceModel instance.

**ContextItems [0..\*]:** ContextItem. This is a list of ContextItem objects that allow capturing geopolitical-context-dependent information such as country-specific registration numbers and agency certifications. See Part 1 of this standard for an in-depth explanation.

### 9.1.2 DeviceModelMetadataPayload

This class contains metadata associated with a DeviceModel. It is very similar to the ObservationConfigurationPayload class (Clause 8.X) but lacks the ability to communicate a spatial extent; this is appropriate because a spatial extent would describe a particular device instance, and not a device model.



Figure : DeviceModelMetadataPayload class. Note the absence of an identifier.   
(This class is intended for use only by value.)

The properties of DeviceModelPayload are:

**observationCode [0..1]**: Code. This code represents the meaning of observations produced by this sensor model. It can represent a combination of various aspects of the observation such as its observed property, methods, and unit of measure. It is expected to match the code property of an ObservationCodeDefinition object held in semantic infrastructure available to the users of the observations being produced and exchanged.

**valueUoM**: Code. This is an optional machine-readable unit of measure code (e.g., “sec”, “m”, “kg1[m2]-1”) that describes the values produced by the device for the observationCode described above.

**observationCodeComponents [0..\*]**: ObservationCodeComponent. This is a list of objects that can supplement (or replace) the meaning provided by observationCode. Typical examples include communicating resolution, value ranges, sampling period, and power requirements. A more complex application might be describing the depth of each of the sections of a multi-depth soil water content sensor; in this use case, each section would have its own DeviceModelMetadataPayload, all containing the same observationCode, but each containing code components that describe its own depth or range of depths.

## 9.2 Setup Data: Device

Figure 13 below shows the Device class. It is a simple description of an instance of a DeviceModel (note the deviceModelRef property). It is important to note that Device does not contain state information; that is represented using ObservationConfiguration objects (8.5).



Figure 16: UML class diagram of the Device class.

The properties of Device are:

**id**: CompoundIdentifier. This property contains a unique identifier for an instance of Device and can also include an additional set of aliases (4.1), i.e. unique identifiers used in external systems to represent the same idea (See Part 1 of this standard for a detailed explanation of the CompoundIdentifier class).

**serialNumber** [0..1]: String. Contains the serial number of the device in question as communicated by the manufacturer.

**description** [0..1]: String. An optional human-readable description of the device being represented; e.g., “Tensiometer 27”.

**isVirtual** [0..1]: Code. The observations and measurements model presented by ISO 19156 is very rich and flexible; that standard points out that forecasts, nowcasts, hindcasts, and other data products that are not necessarily obtained through direct observation with a sensor easily fit the observations model. To enable this kind of use case, the isVirtual property indicates whether a device is virtual; i.e., a data artifact that represents a source of data.

For example, a gridded reanalysis of rain gauge and weather radar precipitation data may enable a fine-grid rainfall estimate data product. One way of using this operationally on the farm might be in the form of “virtual rain gauges”, where a virtual Device is configured at a desired latitude / longitude, and appears as a rain gauge to a user, but without any physical form.

**deviceType**: Code. This is a code describing the type or category (e.g., weather station, sensor, etc.) of the device in question. It should be machine-readable and come from a controlled vocabulary known to any parties that would come in contact with data form the device.

**deviceModelRef [0..1]**: String. This is a reference to a DeviceModel object corresponding to the Device instance in question. Using this enables assigning the DeviceModel’s default metadata properties to the Device instance, which can subsequently assist in interpreting incoming data for that Device (e.g., by communicating the unit of measure that the device reports in).

**ContextItems [0..\*]**: ContextItem. This is a list of ContextItem objects that allow capturing geopolitical-context-dependent information such as country-specific registration numbers and agency certifications. See Part 1 of this standard for an in-depth explanation.

## 9.3 Setup Data: Locations / Gazetteer

The ISO 19112 standard presents the idea of a *location* as a position with an identifier, and gazetteers as lists of locations. Gazetteers, in the context of 7673, are a list of locations that can be shared; i.e., used by reference. This use by reference is critical to 7673 as it allows a more efficient definition of locations without repeatedly creating instances of the position definition. Instead, the position and identifier are defined once and the reference is used in its place

Complete details of the application of Gazetteer are found in ISO 7673-1. As noted in 7673-1, this standard does not claim to use a complete implementation of ISO19112:2003.

# 10 Examples of Encoded Meaning

## 10.1 Introduction to the examples

The following two clauses contain examples that demonstrate application of the data model for composition of observation codes, component codes and their attendant properties. Clause 10.2, Basic Examples, contains several samples of quantities commonly used in agriculture where automated data acquisition and telemetry systems produce these data. Clause 10.3 Edge Cases shows several examples of quantities that represent more complex applications of OMCodeComponent compositions. These examples are not normative. Rather, they are intended to explain the application of OMCodeComponent using familiar quantities.

## 10.2 Basic examples

The following examples illustrate some basic observation code definitions for quantities common to agricultural environmental monitoring.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **#** | **observationCode** | **description** | **observationCodeComponent** | | | |
| **componentType** | **celector** | **value** | **uomCode** |
| 1 | A\_SOIL\_MOISTURE\_VOL\_30\_CM | Soil moisture content at a depth of 30 centimeters (unspecified aggregation) | FEATURE\_OF\_INTEREST | ABSTRACT | SOIL |  |
| OBSERVED\_PROPERTY | INTENSIVE | VOLUMETRIC\_MOISTURE\_CONTENT |  |
| PARAMETER | DEPTH | 300 | mm |
| 2 | A\_SOIL\_MOISTURE\_VOL\_30\_CM | Soil moisture content at a depth of 30 centimeters, averaged over 15 minutes | FEATURE\_OF\_INTEREST | ABSTRACT | SOIL |  |
| OBSERVED\_PROPERTY | INTENSIVE | VOLUMETRIC\_MOISTURE\_CONTENT |  |
| AGG\_METHOD | STATISTIC | MEAN |  |
| AGG\_METHOD | WINDOW | 900 | sec |
| PARAMETER | DEPTH | 300 | mm |
| 3 | A\_ET\_C | Daily crop evapotranspiration (ET) | FEATURE\_OF\_INTEREST | CROP | (code for crop) |  |
| OBSERVED\_PROPERTY | EXTENSIVE | EVAPOTRANSPIRATION |  |
| AGG\_METHOD | STATISTIC | SUM |  |
| AGG\_METHOD | WINDOW | 86400 | sec |
| 4 | A\_CUMUL\_ET1 | Cumulative crop ET (cumulative from season start to season end)  Note: Observations should have a parameter specifying the season start date. | FEATURE\_OF\_INTEREST | CROP | (code for crop) |  |
| OBSERVED\_PROPERTY | EXTENSIVE | EVAPOTRANSPIRATION |  |
| AGG\_METHOD | STATISTIC | SUM |  |
| AGG\_METHOD | WINDOW | GROWING\_SEASON |  |
| 5 | A\_CUMUL\_ET2 | cumulative crop ET (cumulative from season start to "today") | FEATURE\_OF\_INTEREST | CROP | (code for crop) |  |
| OBSERVED\_PROPERTY | EXTENSIVE | EVAPOTRASPIRATION |  |
| AGG\_METHOD | STATISTIC | SUM |  |
| AGG\_METHOD | WINDOW\_START\_TIME | (ISO 8601 datetime for season start) |  |
| AGG\_METHOD | WINDOW\_END\_TIME | (ISO 8601 datetime for “now”) |  |
| 6 | A\_MEAN\_WIND\_SPEED | wind speed at 2m averaged according to ASAE EP505.1 | FEATURE\_OF\_INTEREST | ABSTRACT | AIR |  |
| OBSERVED\_PROPERTY | INTENSIVE | WIND\_SPEED |  |
| AGG\_METHOD | INTENSIVE | ASABE\_EP505 |  |
| PARAMETER | height | 2 | m |

## 10.3 Edge Cases

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **#** | **observationCode** | **description** | **observationCodeComponent** | | | |
| **componentType** | **selector** | **value** | **uomCode** |
| 7 |  | growing degree day | METHOD | AGGREGATION\_METHOD | SUM |  |
| FEATURE\_OF\_INTEREST | ? | ATMOSPHERE |  |
| OBSERVED\_PROPERTY | EXTENSIVE | GROWING\_DEGREE\_DAYS |  |
| METHOD | AGGREGATION\_WINDOW | 86400 | sec |
| PARAMETER | LOWER\_BASE | 45 | C |
| PARAMETER | UPPER\_BASE | 90 | C |
| 8 |  | growing degree day | METHOD | AGGREGATION\_METHOD | SUM |  |
| FEATURE\_OF\_INTEREST | ? | ATMOSPHERE |  |
| OBSERVED\_PROPERTY | EXTENSIVE | GROWING\_DEGREE-DAYS |  |
| OBSERVED\_PROPERTYModifier | Allen Method |  |  |
| METHOD | AGGREGATION\_WINDOW | 1 | day |
| PARAMETER | LOWER\_BASE | 45 | C |
| PARAMETER | UPPER\_BASE | 90 | C |
| 9 |  | growing degree day | METHOD | CALCULATION | GDD\_ALLEN |  |
| FEATURE\_OF\_INTEREST | ? | ATMOSPHERE |  |
| OBSERVED\_PROPERTY | INTENSIVE | TEMPERATURE |  |
| OBSERVED\_PROPERTYModifier | Degree Day |  |  |
| METHOD | AGGREGATION\_WINDOW | 1 | day |
| PARAMETER | LOWER\_BASE | 45 |  |
| PARAMETER | UPPER\_BASE | 90 |  |
| 10 |  | Disease Value (Blight) | FEATURE\_OF\_INTEREST | CROP | PYUCO (EPPO code for *Pyrus communis* / pear) |  |
| OBSERVED\_PROPERTY | DISEASE\_LEVEL |  |  |
| OBSERVED\_PROPERTY | TAXON | ERWIAM (EPPO code for *Erwinia amylovora* / fire blight)) |  |
| METHOD | AGGREGATION\_METHOD |  |  |
| METHOD | CALCULATION |  |  |
| PARAMETER | model | Maryblyt |  |
| 11 |  | Radar Composite Reflectivity | AggMethod | Composite |  |  |
| FEATURE\_OF\_INTEREST | Power |  |  |
| OBSERVED\_PROPERTY | Reflectivity |  |  |
| AggWindow |  | 1 | hour |
| 12 |  | Radar Doppler Velocity | AggMethod | Composite |  |  |
| FEATURE\_OF\_INTEREST | Power |  |  |
| OBSERVED\_PROPERTY | Velocity |  |  |
| OBSERVED\_PROPERTYModifier | Dealiased |  |  |
| AggWindow |  | 5 | minutes |
| PARAMETER | method | FourDD |  |
| 13 |  | lightning flash rate | AggMethod | Rate |  |  |
| FEATURE\_OF\_INTEREST | Lightning |  |  |
| OBSERVED\_PROPERTY | Instance/Flash |  |  |
| AggWindow |  | 300 | Sec |
| 14 |  | Categorical Rain | AggMethod | Categorical |  |  |
| FEATURE\_OF\_INTEREST | Precipitation |  |  |
| OBSERVED\_PROPERTY | Type |  |  |
| AggWindow |  | 3600 | Sec |
| PARAMETER | model | HRRRv2 |  |
|  |  | Ground Heat Flux | AggMethod | Average |  |  |
| FEATURE\_OF\_INTEREST |  | SOIL |  |
| OBSERVED\_PROPERTY |  | HEAT\_FLUX |  |
| AggWindow |  | 900 | sec |
|  |  |  |  |
| 15 |  | lifted index | AggMethod | Average |  |  |
| FEATURE\_OF\_INTEREST | Atmosphere/Air | ATMOSPHERE |  |
| OBSERVED\_PROPERTY | ? | LIFTED\_INDEX |  |
| AggWindow |  | 3600 | Sec |

# 11 Implementation notes

Criteria for deciding what components should be included in an observation code and which should be used as parameters of the observation.

## 11.1 Implementing Metadata at the Observation, Observation Collection, and Observation Dataset level

### 11.1.1 Observation-level metadata

(Show how to use ObservationCodeComponents for this)

### 11.1.2 Observation-collection-level metadata

(Show how to use ObservationCodeComponents for this)

### 11.1.3 Dataset-level metadata

The ISO 19156 standard references the ISO 19115 standard for metadata. The ISO 19115 standard is very comprehensive, but at the same time quite complex. This present standard uses a subset of a simpler metadata standard, the Dublin Core, to provide the following information:

DCPublisher: An entity responsible for making the resource available.

DCType: The nature or genre of the resource. Name used by provider to identify this particular dataset format / contents. This is NOT the name of a particular instance document.

DCFormat: The distributed file format of variable values in the named data set.

DCTitle: A name given to the resource. Typically, a Title will be a name by which the resource is formally known.

DCDescription: Description may include but is not limited to: an abstract, a table of contents, a graphical representation, or a free-text account of the resource. Any miscellaneous notes about the recording, collection, and storage of values in a record and/or the retrieval and use of the named data set.

# 

# Annex A (normative) Testing the implementation against ISO 19156 checklist

A.1 General

Table 4 below presents the tests laid out in ISO 19156 Annex A, examples used to test, and comments.

Table 4 — lorem ipsum

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Test** | **Purpose** | **Example** | **Comments** |
| 1 | A.1.1 Observation interchange | Verify that an interchange schema correctly implements the mandatory properties, associations and constraints of OM\_CategoryObservation | OK | ATTRIBUTES  **parameter**: Implemented through the CodeComponent class, using componentType = PARAMETER  **phenomenonTime**: ISO 19156 implements it as a TM\_Object, which can be a timestamp or a time interval. This standard provides a property of type DateTime for the more common single timestamp use case, and a time interval capability through an optional TimeScope object with its dateContext property set to PHENOMENON.  **resultQuality**: Implemented through a list of observationCodeComponent.  **resultTime**: Implemented as a TM\_Instant (i.e., single timestamp) in ISO 19156. This standard implements it through an optional TimeScope object with dateContext set to RESULT  **validTime**: This is implemented in ISO 19156 as a TM\_Period (i.e., a double timestamp). This standard implements this through an optional TimeScope object with dateContext=VALIDITY\_RANGE.  ASSOCIATIONS  **feature of interest (FOI)**: ISO 19156 implements this using instances of the GFI\_Feature class. This standard implemented the feature of interest in four ways, depending on the use case:  **implicit**: If the feature of interest is implicit (e.g., the air in a weather measurement, the soil implicit in a soil water content) the FOI is communicated via CodeComponents on the the observation code, with ComponentType = FEATURE\_OF\_INTEREST, and, e.g., selector=AIR.  **explicit by code component**: In this case, the FOI is communicated via a CodeComponent on the observation, with componentType = FEATURE\_OF\_INTEREST, and, e.g., selector=SOIL\_CORE, and the value containing a unique identifier.  **explicit by value**: In this case, the precise position and shape on the geographical plane is communicated via the spatialExtent property.  **explicit by reference**: In this case, the level of detail available (ranging from a reference to the Grower, to Farm, Field, CropZone, etc.) is communicated via a reference to a Grower object, and/or a reference to a Place object.  **observed property (OP)**: This standard implements observed property through controlled vocabularies, either explicitly via a CodeComponent on the observation, with componentType = OBSERVED\_PROPERTY, and selector/value set accordingly; or implicitly, asssumed (or declared) set as a CodeComponent under the ObservationCode that represents the meaning of the Observation.  **procedure**: This standard supports multiple procedure or method types (e.g., observation method, sampling method, aggregation method). Each is denoted through controlled vocabularies, either explicitly via a CodeComponent on the observation, with componentType = OBS\_METHOD (or other code), and selector/value set accordingly; or implicitly, asssumed (or declared) set as a CodeComponent under the ObservationCode that represents the meaning of the Observation.  **metadata**: This standard implements metadata using CodeComponent objects associated with controlled vocabularties.  **observation context**: Related observations are kept together using ObsCollections as described in Clause 6.5.2  **result**: Results are encoded using a character string. The underlying simple data type of this value is represented using the valueType property on an observationCodeDefinition object.  CONSTRAINTS  The ISO 19156 standard includes contraints at the data model level (e.g., as shown for OM\_Observation in Figure 2). This standard does not include constraints at the schema or data model level; they are assumed enforced by business rules. |
| 2 | A.1.2 Measurement interchange | Verify that an interchange schema correctly implements the mandatory properties, associations and constraints of OM\_Measurement. | OK | The OM\_Measurement class differs from its parent class in that the data type of its result / value is a Number.  This standard supports this class. While all results are encoded as strings, the ObservationCodeDefinition object corresponding to the observationCode property of the Measurement has a property, valueType, that can be set to "NUMBER" to communicate that the result should be interpreted accordingly. |
| 3 | A.1.3 Category observation interchange | Test Purpose: Verify that an interchange schema correctly implements the mandatory properties, associations and constraints of OM\_CategoryObservation | OK | The OM\_CategoryObservation class differs from its parent class in that the data type of its result / value is ScopedName. (Defined in ISO 19103)  This standard supports this class, as an enumerated value; (i.e., valueType="ENUM"); essentially, a string assumed to belong to a vocabulary associated with the ObservationCode definition. |
| 4 | A.1.4 Count observation interchange | Test Purpose: Verify that an interchange schema correctly implements the mandatory properties, associations and constraints of OM\_CountObservation. | OK | The OM\_CountObservation class differs from its parent class in that the data type of its result / value is an Integer.  This standard supports this class. While all results are encoded as strings, the *ObservationCodeDefinition* object corresponding to the observationCode property of the Observation has a property, valueType, that can be set to COUNT to communicate that the result should be interpreted as an integer count value. |
| 5 | A.1.5 Truth observation interchange | Test Purpose: Verify that an interchange schema correctly implements the mandatory properties, associations and constraints of OM\_TruthObservation. | OK | The OM\_TruthObservation class differs from its parent class in that the data type of its result / value is a Boolean (true / false) value.  This standard supports this class. While all results are encoded as strings, the *ObservationCodeDefinition* object corresponding to the observationCode property of the Observation has a property, valueType, that can be set to BOOLEAN to communicate that the result should be interpreted as a truth value. |
| 6 | A.1.6 Temporal observation interchange | Verify that an interchange schema correctly implements the mandatory properties, associations and constraints of OM\_TemporalObservation. | OK | ~~The OM\_TruthObservation class differs from its parent class in that the data type of its result / value is a TM\_Object (Defined in ISO 19108)~~  This standard supports this class. However, since the TM\_Object supports the concepts of time stamp, time interval, and duration, this standard uses three different mechanisms to represent them. |
| 7 | A.1.7 Geometry observation interchange | Verify that an interchange schema correctly implements the mandatory properties,  associations and constraints of OM\_GeometryObservation.  Reference: ISO 19156, 8.2.2 | ? Result is a shape? | ISO 19156 mandates that the result be of type GM\_Object  The equivalent on our end is of type Shape  We need to encode the result, as a WKT string (or GeoJSON geometry) in the string value of the Observation result.  The data type description tells us how to interpret this. |
| 8 | A.1.8 Complex observation interchange |  |  |  |
| 9 | A.1.9 Discrete coverage observation interchange |  |  |  |
| 10 | A.1.10 Point coverage observation interchange |  |  |  |
| 11 | A.1.11 Time series observation interchange |  |  |  |
| 12 | A.2.1 Sampling feature interchange | Verify that an interchange schema correctly implements the mandatory properties, associations and constraints of a concrete subclass of SF\_SamplingFeature.  Reference: ISO 19156, Clauses 9, 10 and 11. |  |  |
| 13 | A.2.2 Sampling feature collection interchange | Verify that an interchange schema correctly implements the mandatory properties, associations and constraints of SF\_SamplingFeatureCollection.  Reference: ISO 19156, 9.2.4. |  |  |
| 14 | A.2.3 Spatial sampling feature interchange | Verify that an interchange schema correctly implements the mandatory properties, associations and constraints of a concrete subclass of SF\_SpatialSamplingFeature.  Reference: ISO 19156, 10.2. |  |  |
| 15 | A.2.4 Sampling point interchange | Verify that an interchange schema correctly implements the mandatory properties, associations and constraints of SF\_SamplingPoint.  Reference: ISO 19156, 10.2.2. | FOI? / Gazzetteer? |  |
| 16 | A.2.5 Sampling curve interchange |  | FOI? / Gazzetteer |  |
| 17 | A.2.6 Sampling surface interchange |  | FOI? / Gazzetteer |  |
| 18 | A.2.7 Sampling solid interchange | Verify that an interchange schema correctly implements the mandatory properties, associations and constraints of SF\_SamplingSolid.  Reference: ISO 19156, 10.2.2. |  |  |
| 19 | A.2.8 Specimen interchange | Verify that an interchange schema correctly implements the mandatory properties, associations and constraints of SF\_Specimen.  Reference: ISO 19156, 11.2. | FOI? / Gazzetteer | This standard does not provide a schema corresponding to SF\_Specimen. It does allow for referencing a specimen as feature of interest using a Code component, though. |

# Annex B (informative) Sample Controlled Vocabularies for Code Components

## B.1 Controlled vocabulary for ObservationCodeComponent.componentType (Normative)

The table below shows a list of possible componentType values used in the context of the ObservationCodeComponent class. Values from this list shall be used to populate code component definitions and code components.

|  |  |  |
| --- | --- | --- |
| componentType value | Description | Comments |
| FEATURE\_OF\_INTEREST | Feature of interest (6.3.2) |  |
| OBS\_PROPERTY | Observed property (6.3.3) |  |
| OBS\_METHOD | Represents the Observation Method aspect of Procedure (6.3.4) |  |
| SAMPLING\_STRATEGY | Represents the Sampling Method aspect of Procedure (6.3.4) |  |
| AGG\_METHOD | Represents the Aggregation method of Procedure (6.3.5) |  |
| METADATA | Metadata elements (6.3.5) |  |
| METADATA\_DEVICE | A subset of metadata specific to devices | Example: the firmware version of the smart sensor that produced an observation. |
| EVENT | A subset of metadata specific to Observations that are modelled as events or occur in the context of a state machine. | Example: An automated, image-recognition-based insect trap communicates event information that can help contextualize its observations, such as when it takes pictures, when the sticky medium has been refreshed, etc. |
| PARAMETER | Parameters (6.2.4) | Example: Depth underground at which a soil water sensor is measuring. |

## B.2 Controlled vocabulary for ObservationCodeComponent.selector (Normative)

Complete

# Bibliography

[1] ASABE EP505.1 APR2015 – Measurement and Reporting Practices for Automatic Agricultural Weather Stations

[2] Hanway, John (1966) How a Corn Plant Develops. Iowa State University, Ames, IA. Online at <https://dr.lib.iastate.edu/handle/20.500.12876/90185>

[3] ISO 11783-10 - Task controller and management information system data interchange

[4] ISO/IEC Directives, Part 2, *Rules for the structure and drafting of International Standards*, 2001

[5] ISO/IEC TR 10000‑1, *Information technology — Framework and taxonomy of International Standardized Profiles — Part 1: General principles and documentation framework*

[6] ISO 10241, *International terminology standards — Preparation and layout*

[11] ISO 31 (all parts), *Quantities and units*

[15] ISO/IEC 13249-3:2011 - *Information technology -- Database languages -- SQL multimedia and application packages -- Part 3: Spatial (WKT)*

[16] ISO 19135-1:2015 - *Geographic information - Procedures for item registration*

[17] ISO/IEC 19510:2013 - *Information technology – Object Management Group Business Process Model and Notation*

[18] ISO/IEC 19757-3:2006 - *Information technology -- Document Schema Definition Language (DSDL) -- Part 3 - Rule-based validation – Schematron*

[19] ISO 22006:2009 - *Quality management systems--Guidelines for the Application of ISO 9001:2008 to crop production*

[20] WMO, 2008. *No. 8: Guide to Meteorological Instruments and Methods of Observation,7th Edition*. Geneva

[21] World-Wide Web Consortium (W3C) 2013. *PROV Model Primer*. (<https://www.w3.org/TR/prov-primer/>)