OGC® DOCUMENT: 22-053

External identifier of this OGC® document: http://www.opengis.net/doc/PER/POI_UG/1.0

Open Geospatial Consortium

USER GUIDE FOR OGC POINTS OF INTEREST

USER GUIDE

DRAFT

Submission Date: 2023-05-03
Approval Date: 2017-06-29
Publication Date: 2017-01-23
Editor: Charles Heazel, Matthew Brian, John Purss

License Agreement

Permission is hereby granted by the Open Geospatial Consortium, ("Licensor"), free of charge and subject to the terms set forth below, to any person obtaining a copy of this Intellectual Property and any associated documentation, to deal in the Intellectual Property without restriction (except as set forth below), including without limitation the rights to implement, use, copy, modify, merge, publish, distribute, and/or sublicense copies of the Intellectual Property, and to permit persons to whom the Intellectual Property is furnished to do so, provided that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modified Intellectual Property must include, in addition to the above copyright notice, a notice that the Intellectual Property includes modifications that have not been approved or adopted by LICENSOR.

THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD. THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR ERROR FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER'S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY CONTRIBUTOR OF INTELLECTUAL PROPERTY RIGHTS TO THE INTELLECTUAL PROPERTY BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

This license is effective until terminated. You may terminate it at any time by destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual Property, or the operation of the Intellectual Property, infringe, or in LICENSOR's sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications. This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable.

Copyright notice

Note

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

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

CONTENTS

LIST OF TABLES

LIST OF FIGURES

I KEYWORDS

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

ogcdoc, OGC document, API, openapi, html

II SECURITY CONSIDERATIONS

No security considerations have been made for this document.

III SUBMITTING ORGANIZATIONS

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

- organization_1
- organization_2
- organization_3
- etc.

IV ABSTRACT

POI-CM is an open conceptual data model for representing information about points of interest (POI). It is defined through a Unified Modeling Language (UML) object model. This UML model extends the ISO Technical Committee 211 (TC211) conceptual model standards for spatial and temporal data. Building on the ISO foundation assures that the features described in the POI Models share the same spatial-temporal universe as described by related standards (e.g., CityGML).

The aim of developing the OGC POI conceptual model is to reach a common definition of the basic entities, attributes, and relations of "points of interest." In the broadest terms, a point of interest is a location about which information of general interest is available. A POI can be as simple as a set of coordinates and an identifier, or more complex such as a three-dimensional model of a building with names in various languages, information about open and closed hours, and a civic address.

This Users Guide provides extended explanations and examples for the individual concepts that are defined in the POI Conceptual Model Standard. Both the Conceptual Model Standard and this Users Guide are mutually linked to facilitate navigation between corresponding sections in these documents.

1 SCOPE

1 SCOPE

This document provides Engineering Guidance on the use of the POI Conceptual Model Standard.

The OGC POI Conceptual Model Standard specifies the representation of points of interest (POI) models. The POI Conceptual Model is expected to be the basis for a number of future implementation standards in which subsets of the Conceptual Model can be implemented. These future standards will be published separately to enable consistent and efficient storage and exchange of data.

The POI Conceptual Model Standard was designed to be concise and easy to use. As a result, most non-normative content has been removed. The purpose of this Users Guide is to capture that non-normative content and make it easy to access if and when needed.

2

NORMATIVE REFERENCES

2 NORMATIVE REFERENCES

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

Schema.org: http://schema.org/docs/schemas.html

G. Klyne, C. Newman: RFC 3339, *Date and Time on the Internet: Timestamps*. Internet Engineering Task Force (2002). https://raw.githubusercontent.com/relaton/relaton-data-ietf/master/data/reference.RFC.3339.xml

H. Butler, M. Daly, A. Doyle, S. Gillies, S. Hagen, T. Schaub: RFC 7946, *The GeoJSON Format*. Internet Engineering Task Force (2016). https://raw.githubusercontent.com/relaton/relaton-data-ietf/master/data/reference.RFC.7946.xml

3

INTRODUCTION

3 INTRODUCTION

There are many systems and applications that need to have information about locations in the world. For example, such “point of interest” (POI) data is needed in navigation systems, mapping, geocaching, location-based social networking games, and augmented reality browsers, among many others.

POI data has traditionally been exchanged in proprietary formats by various transport mechanisms. This specification defines a flexible, lightweight, extensible POI data model. This will enable content publishers to effectively describe and efficiently serve and exchange POI data.

POI-CM is a common semantic information model for the representation of POI objects that can be shared over different applications. The latter capability is especially important with respect to the cost-effective sustainable maintenance of POI set models, allowing the possibility of selling the same data to customers from different application fields.

POI-CM is an open conceptual data model for the storage and exchange of POI models. It is defined through a Unified Modeling Language (UML) object model. This UML model extends the ISO Technical Committee 211 (TC211) conceptual model standards for spatial and temporal data. Building on the ISO fundation assures that the features described in the POI Models share the same spatial-temporal universe as features described by related standards (e.g., CityGML).

A POI is not a dataset. Rather, it is a feature type that enhances an existing dataset of features. POI-CM builds on the ISO General Feature Model (ISO 10109) and the ISO Geometry Model (ISO 19107). To avoid reinventing things, POI-CM also borrows some classes from ISO Common Data Types (ISO 19103) and OGC CityGML 3.0.

The POI Conceptual Model standard defines the conceptual model in UML and is the focus of this Users Guide. The the future, separate implementation standards can be published for each encoding to be defined. This separation permits generality as well as specificity. The first POI implementation standard will describe the JSON encoding. Other implementation standards (e.g. for relational database schema) are expected to follow.

4

HOW TO USE THIS RESOURCE

4 HOW TO USE THIS RESOURCE

The Users Guide to the POI Conceptual Model Standard is not intended to be read from start to finish. Rather, it is a resource structured to provide quick answers to questions that an implementer may have about the POI-CM Standard.

The POI-CM Standard includes hyperlinks that can be used to navigate directly to relevant sections of the Users Guide.

Some content in the Users Guide has been copied from the POI Conceptual Model Standard to make the content more accessible to the user. In order to make clear which content in the Users Guide has been copied, the copied text is provided within grey boxes.

5 NONSTANDARD ATTRIBUTES

5 NONSTANDARD ATTRIBUTES

An *attribute* is a named property of a feature. While this POI Conceptual Model Standard specifies some standard attributes (e.g., lifetime attributes), most applications will need to define and use some attributes that are not normatively defined by the standard. This section discusses some strategies for doing this.

There are two main ways to include nonstandard attributes:

1. Use this standard's **POIProperty** class to construct properties, which can than be associated with **POI** class instances using the **hasProperty** association.

2. Extend the schema in the implementation technology (JSON, XML, etc.) to allow for the needed types.

The rest of this section will explore these alternatives.

5.1. Using POIProperty

The UML for a POIProperty looks like this:

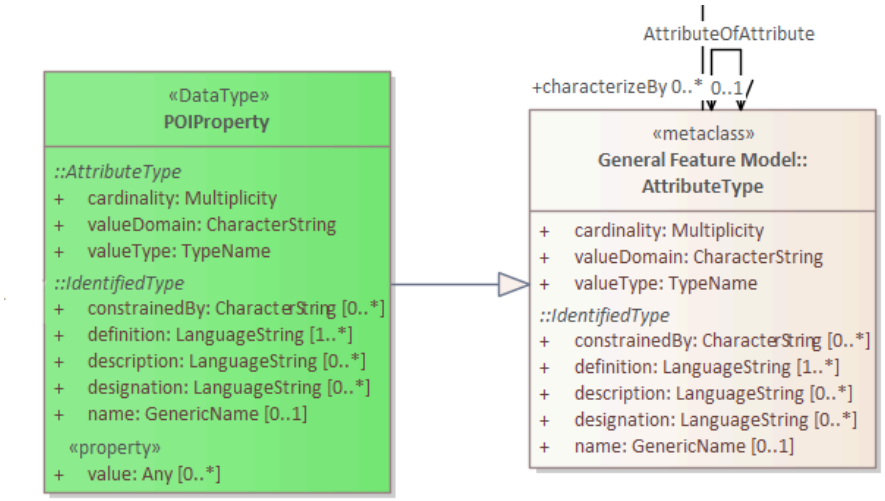

Figure 1

What this means is that a *value* of type POIProperty is self describing: it has attributes that describe the type and usage of the value, as well as the actual value for this instance. In more detail, these are the fields of a POIProperty and how to use them:

name	If this value has a name, what is it? It is optional and not usually needed for the nonstandard attributes that this section is about.
definition	What is the a concise definition of this property? One definition is mandatory; additional definitions might be provided in multiple langauges.
designation	What additional natural language designation is needed, to complement **name**? This is optional and not usually needed. Multiple designations can support multiple languages.
description	What is a description of the this property, including information beyond the concise definition but which may assist in understanding its scope and application. Descriptions are optional, with multiple descriptions allowed to support different languages.
constrainedBy	What constraints are made on this type to ensure integrity of data? As an example, a constraint might specify acceptable combinations of attribute values in one or more feeature instances. This can be a natural language string or something expressed in a formal constraint language. Constraints are optional.
cardinality	How many items can be in the value of one instance of this property? The **cardinality** is a **Multiplicity**, which a range of numbers. It can be a single number (e.g., `1`), an inclusive range of numbers (e.g., `0..2`), or an infinite range of numbers with a minimum (e.g., `1..*`).
valueDomain	What is the underlying domain used to express values? Examples are `"text"` or `"real"`. *TODO What are the valid things to put into this field?*
valueType	What is the name of the type of this property? A valueType is a TypeName, which is defined as "a LocalName that references either a recordType or object type in some form of schema." So, when you implement the conceptual model in a particular implementation technology, this valueType needs to refer to some type defined in that schema. For example, an JSON example schema for this standard includes types such as CI_Telephone, CI_Address, etc., that could be used here.
value	The actual value of the property. It might be empty, a single value, or multiple values, depending on the **cardinality** of this property. The type of the values is give by the **valueType**.

For example, suppose an application needed an attribute called **isPublic**, whose value is true or false depending on whether or not the POI is something the general public can visit. A particular POI could include a **hasProperty** association to a set of values, one of which would include:

```
"POIProperty" {
   "definition": "True if a POI is publicly visitible.",
   "cardinality": 1,
   "valueDomain": "boolean",
   "valueType": "boolean",
   "value": "true"
   }
```

Figure 2 — POIProperty Schema

As another example, a phone number may be needed for some POIs. The *Annex B (Informative) ISO Data Dictionary* of the POI-CM standard describes a number of DataTypes that should be used if they capture the meaning of a needed but nonstandard attribute. The Annex describes a **CI_Telephone** class, with these fields:

Table 1

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
number	CharacterString [1..1]	telephone number by which individuals can contact responsible organisation or individual
numberType	CI_TelephoneTypeCode [0..1]	type of telephone responsible organisation or individual

The **CI_TelephoneTypeCode** can be one of `facsimile`, `sms`, or `voice`.

So one could use a POIProperty like this for a phone number:

```
   POIProperty
      definition: "Telephone number by which individuals can contact
responsible the POI."
      cardinality: 1
      valueDomain: "text"
      valueType: CI_Telephone
      value:
         CI_Telephone
            number: "+1 555 555-5555"
            numberType: voice
```

Figure 3 — POIProperty JSON Example

Besides **CI_Telephone**, some other types in the informative Annex that might often appear as nonstandard attributes are:

- Date, Time, Decimal, Integer, Number, Real, Vector, CharacterString, URI, Boolean, DateTime, CI_Address, CI_Contact, CI_Date, CI_OnlineResource, CI_Organisation, MD_Identification, MD_Keywords,

The above examples showed the values in an implementation-agnostic form. In any particular use of this standard, an implementation technology will be chosen to serialize POI values. Accompanying this User Guide is an example **POI Schema3.json** schema that specifies how to

serialize POIs as JSON objects. Using that schema, a POI with a telephone number and a public visibility flag would be serialized as follows:

```
   {
      "type": "Feature",
      "geometry": {"type": "Point", "coordinates": {45.14, -94.69}},
      "properties": {
         "featureID": 693842,
         "name": {"name" : "Midtown Library"},
         "contactInfo" : {"role": "city representative"},
         "hasFeatureOfInterest": {"href": ""},
         "hasProperty" : [
            {
               "definition": "Telephone number by which individuals can
contact responsible the POI.",
               "cardinality": 1,
               "valueDomain": "text",
               "valueType": "CI_Telephone",
               "value": {"number": "+1 555 555-5555"}
            },
            {
               "definition": "True if a POI is publicly visitible.",
               "cardinality": 1,
               "valueDomain": "boolean",
               "valueType": "Boolean",
               "value": true,
            }
         ]
      }
   }
```

Figure 4 — POI Example with POI Properties

Note that while this works out of the box with standard schemas, it leads to very verbose representations of attribute values in POI instances.

5.2. Extending the Implementation Schema

A second approach to dealing with nonstandard attributes is to extend the schema used to implement the POI-CM in a particular implementation technology.

As a concrete example, suppose JSON is the implementation technology. Part of the supplied example JSON schema for this standard is:

```
    "POI": {
      "type": "object",
      "id": "#PointOfInterest",
      "required": [
          "featureId",
          "contactInfo",
          "hasFeatureOfInterest"
      ],
```

```
  "properties": {
    "featureID": {"type": "number"},
    "description": {"type": "string"},
    "name": {
      "$ref": "#/properties/GenericName"
    },
    "identifier": {
      "$ref": "#/properties/ScopedName"
    },
    "creationDate": {
       "$ref": "#/properties/DateTime"
    },
    "terminationDate": {
       "$ref": "#/properties/DateTime"
    },
    "validFrom" : {
       "$ref": "#/properties/DateTime"
    },
    "validTo": {
       "$ref": "#/properties/DateTime"
    },
    "contactInfo": {
       "$ref": "#/properties/CI_Responsibility"
    },
    "hasFeatureOfInterest": {
       "$ref": "#/properties/reference"
    },
    "hasMetadata": {
       "$ref": "#/properties/reference"
    },
    "hasProperty": {
       "$ref": "#/properties/POIProperty"
    },
    "keywords": {
       "$ref": "#/properties/MD_Keyword"
    },
    "constraints": {
       "$ref": "#/properties/MD_Constraints"
    },
    "symbology": {
       "$ref": "#/properties/reference"
    },
    "links": {
       "$ref": "#/properties/reference"
    }
  }
}
```

Figure 5 — POI Schema with JSON Extensions

One could consider adding new properties to this list to represent the attributes that are needed for a specific use case that a community of interest wants to agree upon. For example, one could add

```
"telephoneNumber": {
    "$ref": "#/properties/CI_Telephone"
    }
```

Figure 6 — Telephone Number Property

in the above list and then a property `"telephone"` could be used directly in a POI instead of as a self-describing attribute in the **hasProperty" value of a POI. The example schema already includes a schema fragment for *CI_Telephone**, and another of other useful ones (see previous section). If you need a type that isn't already provided, that type could also be inserted into the schema.

5.3. Recommendations for Some Common Nonstandard Attributes for POIs

There are a number of attributes that commonly are needed in use cases for POIs yet are not standardized in the POI-CM. This section suggests some recommended Schema and JSON encodings for these common nonstandard attributes.

5.3.1. Address

An address is a structured or semi-structured way of expressing where a place on earth can be found, usually referencing political areas, route (street) names, and numbers on routes. These are the things one uses to specify where mail is to be delivered, or packages are to be picked up. Special software called *geocoders* can convert an addresses into (latitude, longitude) position on earth.

There are many addressing systems in use in the world. A schema to represent them all precisely would be quite complicated. The recommendation here is to use the **CI_Address** class from the Informative Annex:

```
"CI_Address" {
    "administrativeArea": "CharacterString" [0..1],
    "city": "CharacterString" [0..1],
    "country": "CharacterString" [0..1],
    "deliveryPoint": "CharacterString" [0..1],
    "electronicMailAddress": "CharacterString" [0..1],
    "postalCode": "CharacterString" [0..1]
    }
```

Figure 7 — CI_Address Schema

where the **country**, **adminstrativeArea** (state or province), and **city** give a structuring of three of the political areas containing the POI, and the **postalCode** is the postal or zipcode that some countries use in addresses (varies by country). The **deliveryPoint** is an unstructured way of expressing the rest of the address. E.g., it might be "123 Main St., Unit 3" or "Market Square". The language of the address should be either a common language implicit in the entire dataset (e.g., English), or a language in use in the country in question.

TODO: check out ISO 19160:1 A conceptual model for addressing

5.3.2. Telephone Number

The telephone number is the number to use to contact the POI to ask questions, get service, etc. The recommendation here is to use the **CI_Telephone** class from the Informative Annex:

```
"CI_Telephone": {
    "number": "CharacterString" [1..1],
    "numberType": "CI_TelephoneTypeCode" [0..1]
    }
```

Figure 8 — CI_Telephone Schema

where the **number** contains the dial numerals needed to reach that place. The *ITU-T E.164 standard* (ref) specifies a suitable format for telephone numbers. It starts with a recommended **+** sign, followed by up to fifteen digits (with no spaces or other punctuation). The digits will typically be a country code, then an area code, then a local number. For example, the US local number 555-1234 with an area code of 212 would be represented by this character string:

```
+12125551234
```

Figure 9 — ITU-T E.164 Telephone Number

The optional **numberType** is a one of **facsimile**, **sms**, **voice**, where **voice** is the default if the **numberType** is left out.

5.3.3. Opening Hours

The “opening hours” of a POI are the times when the POI is “open for business”, or, more generally just the times at which the general public can visit a POI. There may be more than one open interval on a day (e.g., meal times for a restaurant). Often, opening hours can be different for each day of the week, but are the same week after week. But occasionally POIs have more complicated opening hours (e.g., “closed the first Monday of every month from May to October”). Also, POIs often have special hours for vacations and holidays.

There are several standards to choose from to express business hours. A simple standard, which covers the usual case of weekly hours that repeat, is the Schema.org **openingHours** property (ref). This standard also assumes that the timezone of the opening hours is clear (presumably, the timezone of the POI in question). An example of opening hours expressed in this format is:

```
openingHours: Tu-Fr 9:00-17:00
openingHours: Sa,Su 9:00-19:00
```

Figure 10 — Simple Opening Hours Example

A more general standard, which handles non-weekly repeating as well as exceptions for vacations, holidays, etc., is the *iCalendar* specification (RFC 5545), in particular its *Calendar*

Availability component (RFC 7953). While one could specify an entire calendar using these standards, the needs of specifying opening hours are served well enough by just giving the Availability part. For example, to specify opening hours in France that one might informally specify as “M: 11am-7:30pm, T-Sat: 10am-7:30pm, Sun: closed; closed Aug 1 — Aug 31”, the value according this this standard would be:

```
openingHours:
   BEGIN:VAVAILABILITY
   UID:uid11
   DTSTAMP:20220101T000000Z
   PRIORITY:0
   BEGIN:AVAILABLE
   UID:uid12
   DTSTART;TZID=Europe/Paris:20220103T110000
   DTEND;TZID=Europe/Paris:20220103T193000
   RRULE;FREQ=WEEKLY;BYDAY=MO
   END:AVAILABLE
   BEGIN:AVAILABLE
   UID:uid13
   DTSTART;TZID=Europe/Paris:20220104T100000
   DTEND;TZID=Europe/Paris:20220104T193000
   RRULE;FREQ=WEEKLY;BYDAY=TU,WE,TH,FR,SA
   END:AVAILABLE
   END:VAVALABILITY
   BEGIN:VAVAILABILITY
   UID:uid14
   DTSTAMP:20220101T000000Z
   PRIORITY:5
   BEGIN:AVAILABLE
   UID:uid15
   DTSTART;TZID=Europe/Paris:20220801T000000
   DTEND;TZID=Europe/Paris:20220831T235959
   RRULE;FREQ=YEARLY;BYMONTH=8
   END:AVAILABLE
   END:VAVALABILITY
```

Figure 11 — Opening Hours Example

The increased expressability of the Calendar Availability standard comes at the expense of verboseness, so implementers might like a choice between the two standards.

There is no class in the Informative Annex for Opening Hours. A suggested conceptual model for Opening Hours that offers the choice between the above two standards is:

```
 "OpeningHours": {
     "openingHoursLines": "CharacterString" [0..],
     "openingHoursFormat": "OpeningHoursFormatCode" [0..1]
     }
```

Figure 12 — Recommended Opening Hours Schema

where **OpeningHoursFormatCode** is a **CodeList** with literals **schemadotorg** and **icalendaravailability**, with the default being **schemadotorg**. Note that while technically the Calendar Availability value is one string, it is inconvenient to deal with such a long value (with line breaks) in JSON, so it is convenient to have the value be a sequence of strings that represent

lines to be concatenated together, with line breakes between them, in order to form the actual specification string. Similarly, the the schema.org format, multiple lines are convenient to be able to represent different weekday ranges that have differing time reanges.

One of the two methods described earlier — Using POIProperty or Extending the Implementation Schema — could be used. If using the latter and using JSON for implementation, the recommendation is to use this additional schema:

```
"openingHours": {
    "openingHoursLines: [
        "line" : "string"
        ],
    "openingHoursFormat": "string"
    }
```

Figure 13 — Alternate Opening Hours Schema

5.3.4. Category

The "Category" of a POI is a word that describes the main purpose, use, or description of the POI. It is a word that would fill in the blank in the statement: "This POI is a _____". Example categories might be **School** or **Clothing Store**. Usually one would like the most specific category that applies (e.g.,. preferring **Men's Clothing Store** over **Clothing Store**, but the latter over **Store**).

There are tens of thousands of possible categories, and there is no generally accepted list that this recommendation can confidently point to. Some examples of some standard category lists are:

Table 2 — Category Lists

NAICS:	The North American Industry Classification System. This is used by the US Census to classify businesses according to their economic activity. They are numeric codes with English language descriptors. While they are meant to classify activities that are not necessarily connected to particular POIs, this classification system is still applicable to POIs, though maybe not at the deepest level of specificity desired. ref
OpenStreetMap:	Open Street Map uses a "Free tagging system" to associate multiple key/value pairs with features (which could be POIs). While not comprehensive and endlessly extensible, it is usually possible to find a key=attribute string that could be used as a category: e.g., building=stadium. craft=winery, or shop=butcher. ref
OGC Indoor Mapping Occupant Category:	The OGC Indoor Mapping OGC Community standard (ref) has an **Occupant category** list that has a number of useful categories for POIs.
GeoNames Ontology:	The GeoNames geographic database (ref) has an OWL ontology for Features (which are akin to POIs). It has many kinds of POIs but not many types of commercial shops and restaurants.

None of these is comprehensive enough or granular enough to serve the use case of "I'm looking for a POI that offers this product, service or experience" for the full range of things people need

to find. In the absence of anything better, the NAICS list seems best and the recommendation would be to use that as the code list. However, in order to allow for ultimate flexibility, the following schmea is recommended.:

```
"category": {
    "category": "CharacterString" [0..1],
    "categorySystem": "CategorySystemCode" [0..1]
    }
```

Figure 14 — Recommended Category Schema

where **CategorySystemCode** is a **CodeList** with literals **naics**, **osm**, **ogcindoor**, **geonames**, and **custom**, where **custom** is the default if none is listed, and means that the category system is basically freeform (recommended as English language text).

One of the two methods described earlier — Using POIProperty or Extending the Implementation Schema — could be used to use this class. If using the latter and using JSON for implementation, the recommendation is to use this additional schema:

```
"category": {
    "category": "string",
    "categorySystem": "string"
    }
```

Figure 15 — Alternate Category Schema