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: 2023-08-16
Approval Date: 2017-06-29
Publication Date: 2017-01-23
Editor: Charles Heazel, Matthew Brian, John Purss

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):

- Digital Flancers
- Google
- HeazelTec
- Pangaea Innovations
- PEREY Research Consulting
- US Army Geospatial Center

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.

G. Klyne, C. Newman: IETF RFC 3339, *Date and Time on the Internet: Timestamps*. RFC Publisher (2002). https://www.rfc-editor.org/info/rfc3339.

H. Butler, M. Daly, A. Doyle, S. Gillies, S. Hagen, T. Schaub: IETF RFC 7946, *The GeoJSON Format*. RFC Publisher (2016). https://www.rfc-editor.org/info/rfc7946.

ISO: ISO 19103:2015, Geographic Information – Conceptual Schema Language

ISO: ISO 19107:2003, Geographic Information – Spatial Schema

ISO: ISO 19109:2015, Geographic Information – Rules for Application Schemas

OGC: the OGC City Geography Markup Language (CityGML) Part 1: Conceptual Model Standard, OGC document 20-010

OMG: the OMG® Unified Modeling Language, Version 2.5, 2015, https://www.omg.org/spec/UML

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 across many use cases. The ability for a POI using this conceptual model to be shared and reused 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,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

PAYLOAD: NONSTANDARD ATTRIBUTES

5 PAYLOAD: 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. The mechanism for including non-standard attributes in a POI is to use the **POI_Payload** property.

5.1. Using POI_Payload

A **POI_Payload** is only partially specified in this POI Conceptual Model Standard. These properties are specified:

usesSchema	The value identifies a *schema* for the payload. How this idenfication is done and what the schema looks like depends on the particular implentation technology used. We'll see some examples below. The schema describes the *syntax* for the other properties of the payload, and the value for identifying the schema is typically a *URI*.
hasDefinition	The value is a companion to the schema referenced by **usesSchema**, and it describes the *semantics* of the various payload fields. Typically this will be plain text descriptions of the properties, what standards they adhere to, etc.

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 **hasPayload** association to a payload that looks like this in conceptual form:

Table 1

ATTRIBUTE	VALUE TYPE AND MULTIPLICITY	DEFINITION
usesSchema	CharacterString [1..1]	URI identifying the schema for this payload
hasDefinition	CharacterString [1..1]	URI identifying the definition of the payload attributes
isPublic	Boolean [1..1]	Is this POI a publicly visitable place

Suppose the implementation technology is JSON. Here is how the above payload might be used in a POI expressed in JSON:

```
{
      "type": "Feature",
      "geometry": {"type": "Point", "coordinates": {45.14, -94.69}},
      "properties": {
         "featureID": 693842,
```

```
        "name": {"name" : "Midtown Library"},
        "contactInfo" : {"role": "city representative"},
        "hasFeatureOfInterest": {"href": ""},
        "hasPayload": {
          "usesSchema" : "https://example.org/schema/egpoi.json",
          "hasDefinition" : "https://example.org/schemadef/egpoi",
          "isPublic:" : true
        }
      }
  }
```

Figure 1 — POI Example of a POI with a POI_Payload (JSON)

The value of the **usesSchema** property should reference a schema that might look like this:

```
{
    "$schema": "http://json-schema.org/draft/2020-12/schema#",
    "$id": "https://example.org/schema/egpoi.json",
    "title": "POI Payload for POI with IsPublic property",
    "typee": "object",
    "properties": {
      "isPublic": { "type": "boolean"}
    }
  }
```

Figure 2 — POIProperty Example Payload Schema (JSON)

The value of the *hasDefinition" property should reference a file that describes the semantics corresponding to the above schema. It might look something like this:

```
isPublic: This field is a boolean that should be true if the POI is publicly
visitable, else false.
```

Figure 3 — POIProperty Example Payload Schema (JSON)

5.2. Recommended Schema and Semantics for Common Nonstandard Attributes

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 for these common nonstandard attributes.

This POI Conceptual Model Standard has an Informative Annex which contains a number of conceptual model fragments taken from other OGC Standards which may be useful and recommended for using in payloads when the corresponding concept needs modeling. Among the useful ones are:

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

The rest of this section recommends how to handle four common POI attribute: address, telephone number, opening hours, and category.

5.2.1. 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 4 — 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

5.2.2. 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 5 — 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 6 — 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.

5.2.3. 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 7 — 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 8 — 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 9 — 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.

5.2.4. 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 10 — 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).

6

CHOOSE RESTAURANT USE CASE

6 CHOOSE RESTAURANT USE CASE

6.1. Use Case Overview

The **Choose Restaurant** use case is where a system is desired to help users find restaurants near them and information abou those restaurants to decide which, if any, they want to use.

In order to choose a restaurant to eat in or get delivery from, users need to be able to POIs that are restaurants, with certain constraints (e.g., within a certain distance from a point on earth where the user is or expects to be). From that set of possibilities, they may wish to examine further information about each restaurant, to narrow down the choice. This information may include some or all of the following:

- Serves cuisines that the user can eat and likes.
- Is open during the hours the user intends to visit, and/or serve the type of meal (breakfast, lunch, dinner, etc.) the user wants.
- Has service style the user prefers (take-out vs dine-in).
- Has a price level the user prefers.
- Has a bar and/or serves alcohol.
- Has delivery.
- Has order-ahead with pickup.
- Requires reservations or reservations recommended.
- Will have a busyness level that the user prefers at the time of intended vist.
- Has good reviews.
- Accepts particular payment methods.
- Has an ambience the user prefers (family-friendly, upscale, romantic, etc.).
- Has an all-you-can eat buffet.
- Shows sports on TV.
- Has a happy hour.
- Is Handicap-accessible.
- Has available free parking.

- Has health-related requirements (vaccination and/or masks).

To support this use case, a system will need a comprehensive set of restaurant POIs, each with attributes to help answer some or all of the information needs listed above.

The POI-CM standard specifies a way to represent a POI with its name, and its (latitude, longitude) position on earth. The rest of the information needed will have to be specified using non-standard attributes, as described in the Nonstandard Attributes section.

6.2. Nonstandard Attributes for Restaurant POIs

The most important nonstandard attributes needed for the restaurant use case are:

- Address
- Telephone Number
- Opening Hours
- Category
- Website

The recommended way to represent the first four of these was discussed in the Nonstandard Attributes section. Website is intended to be the "authoritative" web page for the restaurant in question — that is, the page that the restaurant would like users to visit if they want to learn more about their restaurant. That web page might answer many of the other attribute questions (e.g., "has a bar") in the list of the previous section. While one could use the **CI_Contact** type from the informative annex, a simpler recommendation is to extend the schema like by adding this:

```
"Website" {
   "url" : "CharacterString[1..1]"
}
```

Figure 11

where the **url** value is a Uniform Resource Locator.

6.3. Example

Putting it all together, here is an example of a restaurant POI using JSON.

```
{
  "type": "Feature",
  "geometry" : {
```

```
    "type" : "Point",
    "coordinates" : [
      45.1491802,
      -84.6907891
    ]
  },
  "featureID" : 693842,
  "name" : {
    "name" : "Chez Jacques"
  },
  "hasPayload" : {
    "usesSchema" : "https://genpoijson.org/schema/interchangepoi.json",
    "website" : {
      "url" : "http://chezjacquesrestaurant.com"
    },
    "telephoneNumber" : {
      "number" : "+13205555555"
    },
    "address" : {
      "deliveryPoint" : "400 Atlantic Ave",
      "city" : "Grove City",
      "administrativeArea" : "Minnesota",
      "postalCode" : "56243"
    },
    "category" : {
      "category" : "French Restaurant",
      "categorySystem" : "custom"
    },
    "openingHours" : {
      "openingHoursLines" : [
        "line" : "Mo-Fr 12:00-20:00",
        "line" : "Sa 12:00-22:00"
      ],
      "openingHoursFormat" : "schemadotorg"
    }
  }
}
```

Figure 12 — Restaurant POI (JSON)

The schema referenced in **hasPayload** might point to a JSON schema like this:

```
{
  "$schema": "http://json-schema.org/draft/2020-12/schema#",
  "$id": "https://genpoijson.org/schema/interchangepoi.json",
  "title": "Generic POI Payload for POI Interchange",
  "type" : "object",
  "properties": {
    "telephoneNumber": {
      "type": "object",
      "properties": {
        "number": {"type": "string"},
        "numberType": {"type": "string"}
      },
      "required": [ "number" ]
    },
    "address" : {
      "type": "object",
      "properties": {
        "administrativeArea": {"type": "string"},
        "city": {"type": "string"},
        "country": {"type": "string"},
        "deliveryPoint": {"type": "string"},
```

```
        "email": {"type": "string"},
        "postalCode": {"type": "string"}
      }
    },
    "category" : {
      "type": "object",
      "properties": {
        "category": { "type": "string"},
        "categorySystem": {
          "enum" : [ "naics", "osm", "ogcindoor", "geonames", "custom" ]
        }
      },
      "required": [ "category" ]
    },
    "openingHours" : {
      "type": "object",
      "properties": {
        "openingHoursLines": {
          "type": "array",
          "items": {
            "type": "string"
          }
        },
        "openingHoursFormat": {
          "enum" : [ "schemadotorg", "icalendaravailability" ]
        }
      }
    },
    "website" : {
      "type": "object",
      "properties": {
        "url": {
          "type": "string",
          "format": "uri"
        }
      },
      "required": [ "url" ]
    }
  }
}
```

Figure 13 — Schema for the Restaurant POI (JSON)

There is no *hasDefinition" property in the Payload. If there were, it could reference a text file with a form of the commentary in this section.