INTERNATIONAL STANDARD

ISO 19155

First edition
2012-11-01

Geographic information — Place Identifier (PI) architecture

Information géographique — Architecture d'identifiants de lieu (IL)

Reference number
ISO 19155:2012(E)

Contents

Page

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.

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

ISO 19155 was prepared by Technical Committee ISO/TC 211, *Geographic information/Geomatics*.

Introduction

The rapid development of information technology has blurred the boundaries between the real and virtual worlds in such a way that they cannot easily be disassociated from each other. Humans can reference places in both worlds and easily differentiate between them. However for computers to clearly differentiate these places, a set of matched linkages between them are required.

In the discipline of geography, space normally refers to the surface of the earth. However, in other disciplines, space can refer to different paradigms. In architecture, space may be the extent of a room or a building. In mathematics, space is defined as a set having structure. In the context of the World Wide Web space is defined by URLs/URIs that identify web pages.

Within this International Standard "space" is considered as a set having structure, in which a position or location identifies an element.

Currently, within the domain of ISO/TC 211, standards exist for precise positioning and locating using either coordinates or geographic identifiers. However, the concept of place is broader than both position and location. A "place" is referred to as a "position" when that place is identified using coordinates. Similarly, a "place" is referred to as a "location" when that place is identified using geographic identifiers. However, existing standards defined by ISO/TC 211 do not provide a mechanism for the representation of a virtual "place" such as a website, or a construct acting as a "common base" which can be used to refer to the other types of identifiers.

Within this International Standard, "place" is defined as an identifiable part of any space. This may include "places" existing not only in the real world but also those in the virtual world. Places are identified using either "position" by coordinates, "location" by geographic identifiers, or "virtual world identifiers" such as a URI.

In this International Standard, the identifier of a place is referred to as a Place Identifier (PI). A single "place" may be identified using several separate Place Identifiers. Clarification of these relationships is shown in Figure 1.

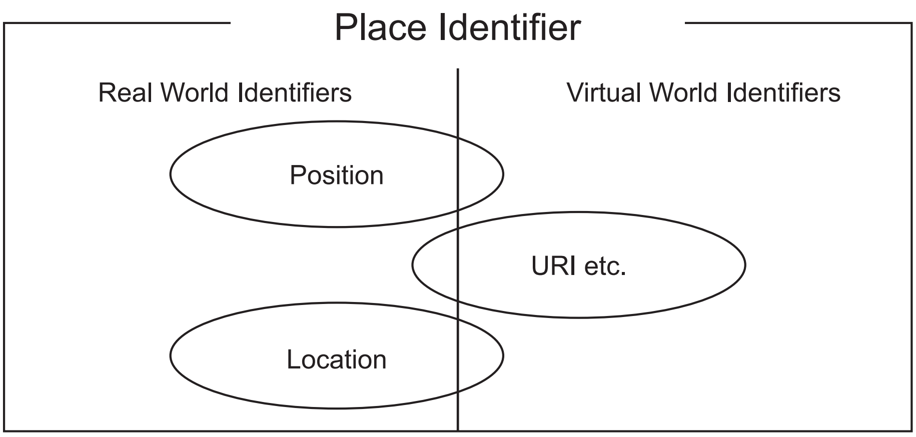

Figure 1 — Relationships among place, position, location and URI

Place descriptions are used for information retrieval. In reality, those identifiers often refer to the same place. Currently these relationships are difficult for machines to correctly distinguish, which impedes the discovery and retrieval of information. The conceptual architecture and reference model defined in this International Standard provides a mechanism for solving these problems.

When implemented, this architecture would enable the access and sharing of place descriptions using the Place Identifier as the standardized method.

Within the reference model, place descriptions are defined using a PI. A PI consists of a reference system (RS), a value, and the valid temporal period of that value.

The internal format and content of the value are determined by each community or domain. The content of the values are not subject to any kind of standardization or unification by this International Standard. The RS is also defined by each community, and should be unique across multiple communities. Subsequently, Place Identifiers are unique within each RS. However, the values of the Place Identifiers may be similar or even identical across multiple communities. This distributed concept ensures that each community would maintain their own Place Identifiers. Well formed Place Identifiers may be shared between communities.

Instead of specifying a framework for a globally unique type of identifier, the key idea of the architecture defined in this International Standard enables the original place descriptions to be easily maintained, without requiring difficult conversions and cross-community harmonization.

An encoding scheme based on Geography Markup Language (GML) (ISO 19136:2007) is normatively defined in this International Standard. In addition, a group of alternate encoding schemes are presented as informative annexes. Depending on the encoding method of choice, globally unique Place Identifiers may be created resulting from the requirements of the encoding method used.

Methods for the conversion of "located features" to Place Identifiers are not covered within the scope of this International Standard. While the direct relationship with the PI Architecture and other Spatial Data Infrastructures (SDIs) is not explained, an implementation of the PI Architecture can be considered part of an SDI. Various constructs, such as registries and databases, may be used to store Place Identifiers. The flexible structure of the Place Identifier will allow for data stored in common GI systems to be easily registered as Place Identifiers, however, the design and implementation of those procedures is out of scope of this International Standard.

Geographic information — Place Identifier (PI) architecture

1 Scope

This International Standard specifies an architecture that defines a reference model with an encoding method for an identifier of a place. The concept of "place" within this International Standard includes "places" not only in the real world but also those in the virtual world. These "places" are identified using either coordinate identifiers, geographic identifiers, or virtual world identifiers such as URI. In this International Standard, an identifier of a place is referred to as a Place Identifier (PI).

The reference model defines a mechanism to match multiple Place Identifiers to the same place. In addition, a data structure and set of service interfaces are also defined in this reference model.

This International Standard is applicable to location based services, emergency management services and other application domains that require a common architecture, across specific domains, for the representation of place descriptions using coordinate, geographic, or virtual world identifiers.

This International Standard is not about producing any kind of specific place description, nor about defining a unique, standardized description of defined places, such as an address coding scheme.

2 Conformance

2.1 Conformance clause

This International Standard specifies four conformance classes. The following conformance clauses should be followed in order to meet the requirements of this International Standard.

2.2 Conformance tests for Semantics

To conform to this International Standard, instances of PI_PlaceIdentifier, PI_ReferenceSystem, PI_MatchingTable, and PI_MatchedPISet shall satisfy the requirements of A.1.

2.3 Conformance tests for Data

To conform to this International Standard, data stored in the PI matching table and the reference system shall satisfy the requirements of A.2.

2.4 Conformance tests for Services

To conform to this International Standard, interfaces between services and users that the PI matching service and the reference system service implement shall satisfy the requirements of A.3.

2.5 Conformance tests for PI encoding

To conform to this International Standard, encoded instances of PI_PlaceIdentifier shall satisfy the requirements of A.4.

3 Normative references

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

ISO/TS 19103:2005, *Geographic Information — Conceptual schema language*

ISO 19111:2007, *Geographic information — Spatial referencing by coordinates*

ISO 19112:2003, *Geographic information — Spatial referencing by geographic identifiers*

ISO 19136:2007, *Geographic information — Geography Markup Language*

4 Terms and definitions

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

4.1
client
software component that can invoke an operation from a *server* (4.16)

[ISO 19128:2005, 4.1]

4.2
coordinate
one of a sequence of *n* numbers designating the position of a point in *n*-dimensional space

[ISO 19111:2007, 4.5]

NOTE In a *coordinate reference system* (4.4), the coordinate numbers are qualified by units.

4.3
coordinate operation
change of *coordinates* (4.2), based on a one-to-one relationship, from one *coordinate reference system* (4.4) to another

[ISO 19111:2007, 4.7]

NOTE Supertype of coordinate transformation and coordinate conversion.

4.4
coordinate reference system
coordinate (4.2) system that is related to an object by a datum

[ISO 19111:2007, 4.8]

NOTE For geodetic and vertical datums, the object will be the Earth.

4.5
gazetteer
directory of instances of a class or classes of features containing some information regarding position

[ISO 19112:2003, 4.2]

NOTE The positional information need not be *coordinates* (4.2), but could be descriptive.

4.6
geographic identifier
spatial reference (4.19) in the form of a label or code that identifies a location

[ISO 19112:2003, 4.3]

EXAMPLE ‘Spain’ is an example of a country name; ‘SW1P 3AD’ is an example of a postcode.

4.7
interface
named set of operations that characterize the behaviour of an entity

[ISO 19119:2005, 4.2]

4.8
place
identifiable part of any space

4.9
Place Identifier
PI
reference that identifies a *place* (4.8)

NOTE The same *place* may be referenced by multiple *Place Identifier* (4.9) instances. Each instance will be associated with a different reference system.

4.10
Place Identifier application
PI application
application providing *services* (4.17) that use *Place Identifiers* (4.9) to end *users* (4.21) or other applications

4.11
Place Identifier matching
PI matching
matching of a *Place Identifier* (4.9) specifying a *place* (4.8) with another type of PI identifying the same place

NOTE 1 A source PI can be matched with multiple target *Place Identifiers*.

NOTE 2 *PI matching* (4.11) can be made among *coordinates* (4.2), *geographic identifiers* (4.6) and identifiers in the virtual world such as URI.

4.12
Place Identifier platform
PI platform
group of *service* (4.17) *interfaces* (4.7) and data structures used for *PI matching* (4.11)

4.13
registration
assignment of a permanent, unique, and unambiguous identifier to an item

[ISO 19135:2005, 4.1.12]

4.14
request
invocation of an operation by a *client* (4.1)

[ISO 19128:2005, 4.10]

4.15
response
result of an operation returned from a *server* (4.16) to a *client* (4.1)

[ISO 19128:2005, 4.11]

4.16
server
particular instance of a *service* (4.17)

[ISO 19128:2005, 4.12]

4.17
service
distinct part of the functionality that is provided by an entity through *interfaces* (4.7)

[ISO 19119:2005, 4.1]

4.18
service metadata
metadata describing the operations and geographic information available at a *server* (4.16)

[ISO 19128:2005, 4.14]

4.19
spatial reference
description of position in the real world

[ISO 19111:2007, 4.43]

NOTE This may take the form of a label, code or *coordinate* (4.2) tuple.

4.20
spatial reference system
system for identifying position in the real world

[ISO 19112:2003, 4.6]

4.21
user
active object that initiates *service* (4.17) *requests* (4.14) to the system

[ISO 19132:2007, 4.58]

NOTE 1 Users are usually objects that act as proxies for people accessing the functionality of the system.

NOTE 2 These objects can be a *PI application* (4.10) or creators of *PI matching* (4.11) tables and reference systems.

5 Abbreviated terms and notation

5.1 Abbreviated terms

CRS Coordinate Reference System

HTTP Hypertext Transfer Protocol

PI Place Identifier

SRS Spatial Reference System

UML Unified Modelling Language

URI Uniform Resource Identifier

URL Uniform Resource Locator

XML eXtensible Markup Language

5.2 UML Notation

The UML notation used in this International Standard is specified in ISO/TS 19103:2005.

6 PI Reference Model

6.1 Background

Place may be understood as an element in any space, in either the real or virtual world. The Oxford English Dictionary defines "place," relevant to the content of this International Standard, with the following entries:

— a particular position or point in space,

— a portion of space occupied by someone,

— a position in a sequence, in particular,

— [in place names] a square or short street.

In this International Standard space is considered as a set having structure, in which a position or location identifies an element. Position is a place typically described by a point or geometry in a space. Location is a place typically described by a geographic identifier such as a street address, postcode, name of a landmark, etc.

"Place" in this International Standard is defined as an identifiable part of any space. This may include "places" existing not only in the real world but also those in the virtual world. This International Standard defines an identifier for a place which is referred to as a "Place Identifier (PI)."

The same place may be identified with multiple Place Identifiers. If the place is identified with coordinates, it is called "position" and if the place is identified with geographic identifiers, it is called "location". Additionally, the place may be identified with online resource identifiers such us URI.

In the General Feature Model of ISO 19109:2005, "position" is a spatial attribute of a feature, "location" is a location attribute of a feature, and a virtual identifier, such as a URI, is a thematic attribute of a feature. Therefore, a PI can be considered as an attribute of a feature.

Humans can more easily see the relationships where those identifiers refer to the same place, yet those relationships are more difficult for machines to distinguish correctly. This difficulty impedes the discovery and retrieval of information. Matching of Place Identifiers enables humans to access information using those Place Identifiers as retrieval keys.

This International Standard defines a reference model in which a method for matching between Place Identifiers is defined. This matching method is shown in Figure 2. Parts of the matching method are already defined in other International Standards. ISO 19111:2007 defines coordinate operations in matching with position. ISO 19112:2003 defines a gazetteer schema that enables matching between location and position. The reference model defined in this International Standard shall conform to previous standards (ISO 19111:2007 and ISO 19112:2003), and further enables the matching between different Place Identifiers.

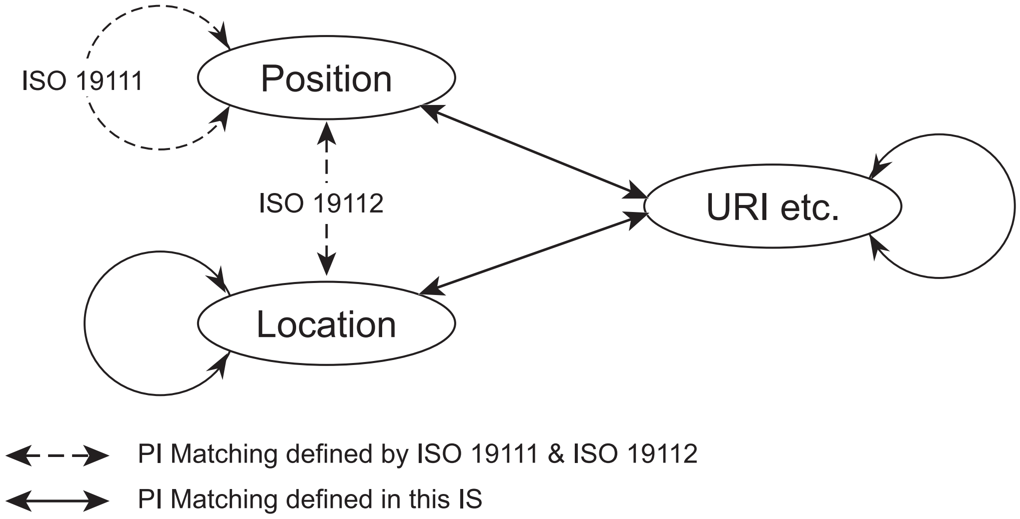

Figure 2 — Place Identifier Matching

6.2 PI Reference Model components

In this International Standard the PI Reference Model defines the basic structure of the PI architecture. An image of the PI Reference Model is shown in Figure 3.

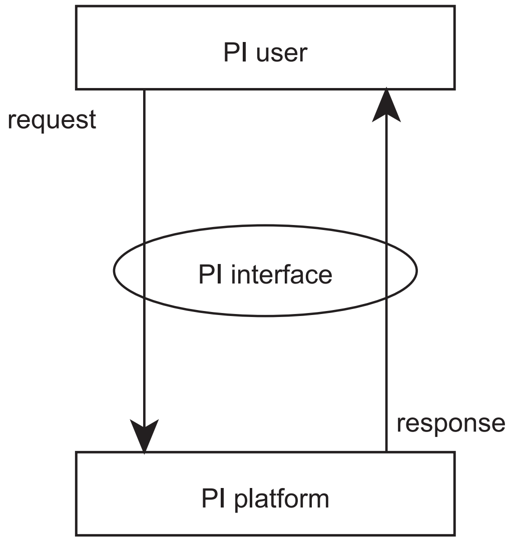

Figure 3 — PI Reference Model

The PI Reference Model consists of three components:

— the PI platform;

— the PI interface;

— a PI user.

The PI platform consists of data and services that are used for PI matching. The PI interface is a standardized interface for requests and responses between the PI platform and a PI user. A PI user is a client that sends requests to, and receives responses from the PI platform through the PI interface.

The PI Reference Model defined in this International Standard is a conceptual model and does not specify an implementation of those components. This model represents a logical structure, not a physical one.

All UML class names in this International Standard shall begin with "PI_" in order to distinguish them from other geographic information standard class names.

6.3 PI platform

The PI platform consists of data and services to register and manage Place Identifiers. The PI platform also specifies a mechanism to match multiple Place Identifiers that identify the same place.

PI_Data is the root class for data in the PI platform. Within PI_Data there are three subclasses: PI_PlaceIdentifier, PI_ReferenceSystem and PI_MatchingTable as shown in Figure 4. PI_PlaceIdentifier specifies the structure of the PI. PI_ReferenceSystem specifies the reference system of the PI. PI_MatchingTable consists of matched sets of multiple Place Identifiers. Each PI in the set identifies the same place.

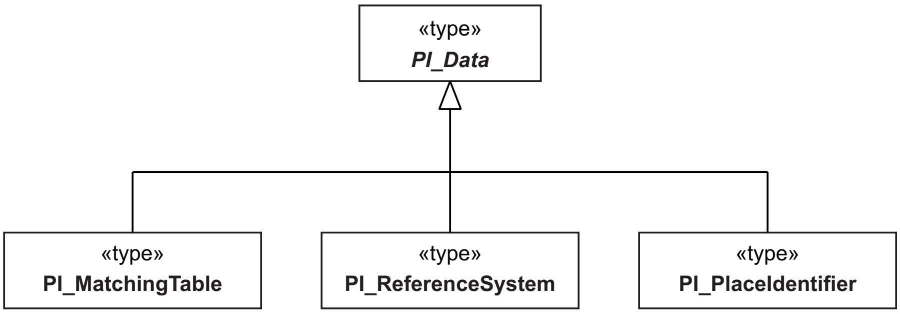

Figure 4 — PI_Data

Details of the PI_Data subclasses are specified in 7.2.2 to 7.2.5.

PI_Service is the root class of services in the PI platform. There are two subclasses: PI_MatchingService and PI_RSService.

The PI_MatchingService manages information for PI matching, and retrieves or transfers the desired PI instances as requested by a user. The PI_RSService manages reference system data obtained from the reference systems which is necessary to facilitate the matching of Place Identifiers. The structure of these services is shown in Figure 5.

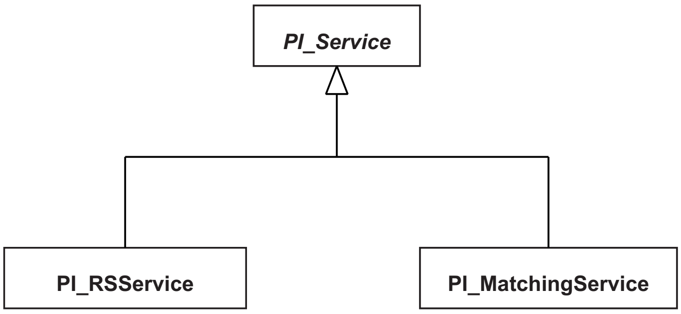

Figure 5 — PI_Service

Figure 5 shows the logical structure of PI services, not a physical structure.

6.4 PI interface

Interfaces are required for using the PI services and retrieving PI data. Within the PI Reference Model, five interfaces are defined:

a) PI update interface;

b) PI retrieval interface;

c) RS update interface;

d) RS retrieval interface;

e) Service metadata retrieval interface.

The aforementioned interfaces shall be used in the following cases:

— between a PI matching service or RS service, and a PI user;

— between a PI matching service and an RS service;

— between different PI matching services.

These interfaces are not limited to automatic processing by machine, as they may also include a user interface or require manual processing by humans.

Clause 8 specifies the parameter definitions for the interfaces and structures.

6.5 PI user

A PI user is a user or application that makes use of the PI platform. A PI user may take any of the following four roles:

a) PI application;

b) PI creator;

c) RS creator;

d) PI matching table creator.

A PI application is a user application that requests information from the PI platform regarding PI matching and the reference systems of related Place Identifiers. An example of a PI application may be an information retrieval application or a routing assistance application.

7 PI platform components

7.1 Overall layout of the PI platform

An overall view of the PI platform is shown in Figure 6.

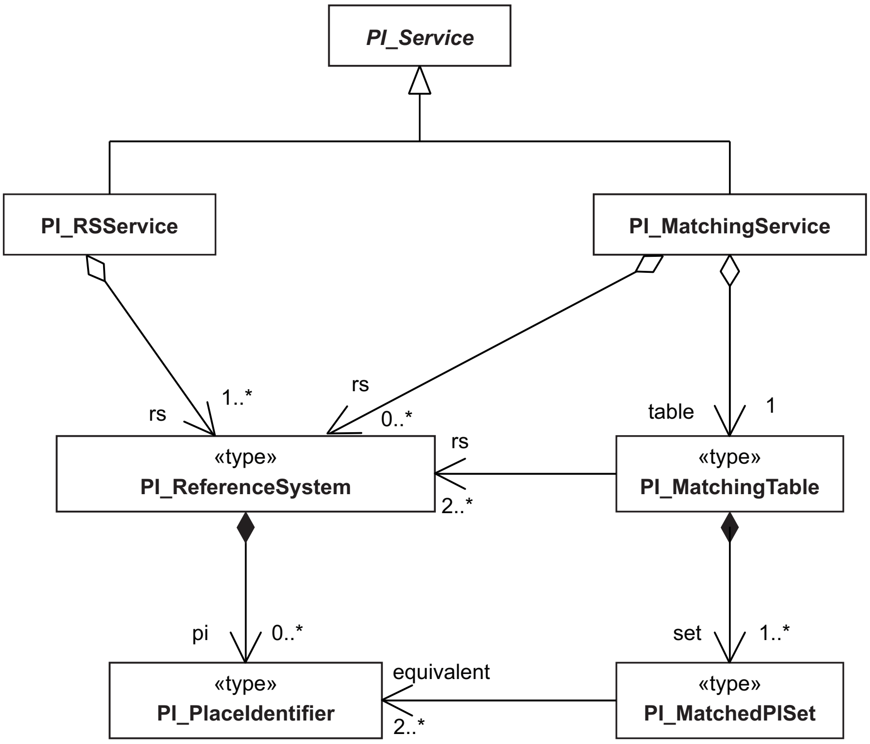

Figure 6 — Overall view of the PI Platform

Each component in the PI platform and the references related to 7.2 are listed in Table 1.

Table 1 — Components of the PI Platform

Components	Description	Reference
PI_PlaceIdentifier	PI (reference that identifies a place)	7.2.2
PI_ReferenceSystem	definition of a domain of PI instances	7.2.3
PI_MatchingTable	table of matched Place Identifiers	7.2.4
PI_MatchedPISet	a matched set composed of multiple Place Identifiers that identify the same place	7.2.5
PI_MatchingService	service to register, manage and match Place Identifiers	7.3.2
PI_RSService	service to register, manage and provide reference systems of Place Identifiers	7.3.3

7.2 Data

7.2.1 Introduction

The data structure for PI matching specified in 7.2.2 to 7.2.5 is a conceptual structure and does not represent an implementation solution.

7.2.2 PI_PlaceIdentifier

Figure 7 shows the structure of the Place Identifier. The PI_PlaceIdentifier type is the basic data type for the Place Identifier. Within the PI_PlaceIdentifier class, the following attributes are defined:

a) The attribute “rs” specifies the reference system of the PI.

b) The attribute “value” shall be the identifier for a place, and is unique within the context of the defined reference system.

c) The optional attribute “validPeriod” specifies the period in which the identifier is valid for the place.

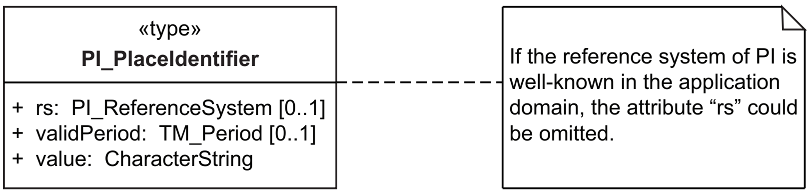

Figure 7 — PI_PlaceIdentifer

If the reference system of PI_PlaceIdentifer is well known within the application domain (community), the “rs” attribute can be omitted. Omission of the “rs” attribute prevents sharing of the Place Identifiers outside of the immediate application domain.

This International Standard contains three annexes that describe how “well formed” PI instances may be encoded.

The normative encoding for “well formed” PI instances is defined in Annex B.

— Annex B (normative): GML (ISO 19136:2007)

Two additional informative encodings are presented in Annexes C and D.

— Annex C (informative): ‘tag’ URI Scheme (IETF RFC 4151, 2005)

— Annex D (informative): Well Known Text (WKT) (ISO 19125-1:2004)

NOTE The encodings listed above do not represent an extensive list. The open format of the “value” attribute enables existing position or location formats to be encoded as Place Identifiers. While it is agreed that within the domain of ISO/TC 211, the use of GML as a normative encoding method is highly suitable, outside of that domain, encodings other than GML may be more acceptable. Conformance with this International Standard specifies that ISO GML be the supported encoding method when designing and implementing platforms defined in this International Standard. However, implementers may choose to support additional encoding methods to increase platform interoperability and appeal. Explanation of the development of those platforms and encodings is not within the scope of this International Standard.

7.2.3 PI_ReferenceSystem

The PI_ReferenceSystem is a class that defines a domain of Pl instances. The deletion of a reference system removes the identities of the PI instances within that reference system. Therefore the association between PI_ReferenceSystem and PI_PlaceIdentifier is a composition. The PI_ReferenceSystem shown in Figure 8 inherits RS_ReferenceSystem from ISO 19111:2007 and has one mandatory attribute, two optional attributes and one association role, as shown in the following list:

a) The attribute “name” is an identifier of the reference system described by RS_Identifier.

b) The optional attribute “domainOfValidity” defines the extent of the reference system.

c) The optional attribute “definition” clarifies the identification and structure of Place Identifiers contained in the reference system.

d) The association role “pi” is the reference to a PI_PlaceIdentifier that is contained within the reference system.

7.2.4 PI_MatchingTable

The PI_MatchingTable is a class that defines “PI matching.” A PI_MatchingTable contains sets of PI instances that identify the same place. The structure of a PI_MatchingTable is shown in Figure 8. A PI_MatchingTable consists of two association roles, as shown in the following list:

a) The ordered association role “rs” is the reference to PI_ReferenceSystem that indicates the order of PI instances matched in this table. This association defines the range of this matching table. The row of the matching table is instantiated from the order of this association.

b) The association role “set” is the inclusion of PI_MatchedPISet which is a matched set of multiple PI instances.

7.2.5 PI_MatchedPISet

A PI_MatchedPISet class is a set of PI instances contained in a PI_MatchingTable class. The structure of a PI_MatchedPISet is shown in Figure 8.

The association role “equivalent” describes the reference to PI instances of the same place.

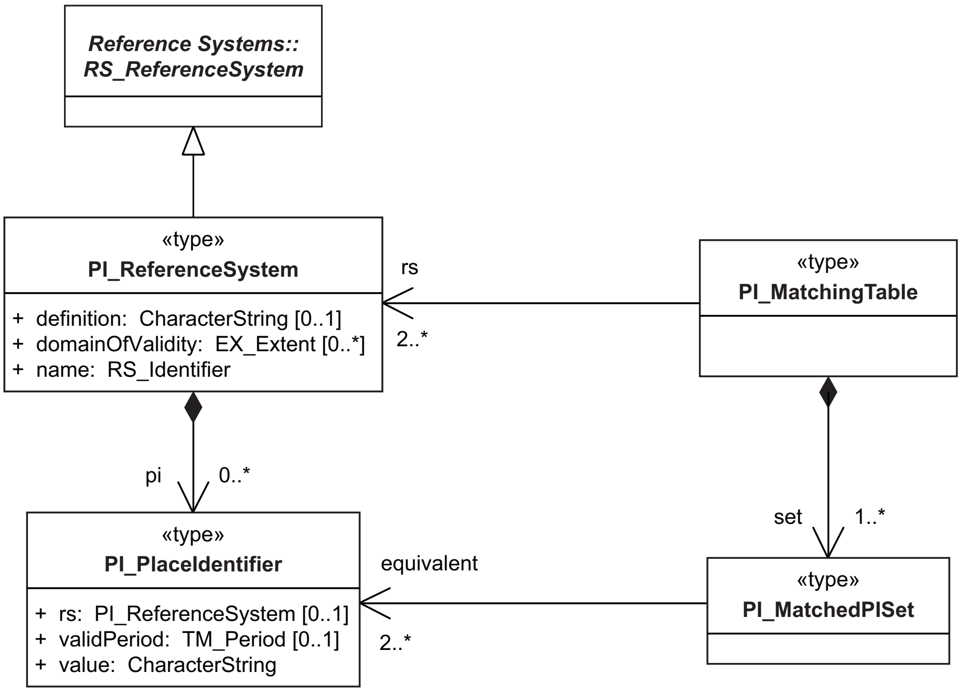

Figure 8 — PI_ReferenceSystem, PI_MatchingTable and PI_MatchedPISet

7.3 Services

7.3.1 Introduction

The various services for registration, management or retrieval of Place Identifiers are specified in 7.3.2 and 7.3.3.

These services shall be classified into the following two groups:

a) PI matching service (7.3.2),

b) RS service (7.3.3).

7.3.2 PI_MatchingService

The PI_MatchingService is a service that registers, manages and retrieves matched PI sets. The conceptual structure of the PI_MatchingService is shown in Figure 9. This is not a physical structure. Multiple PI matching services may exist and any one of those PI services can match to Place Identifiers that are managed by other PI matching services.

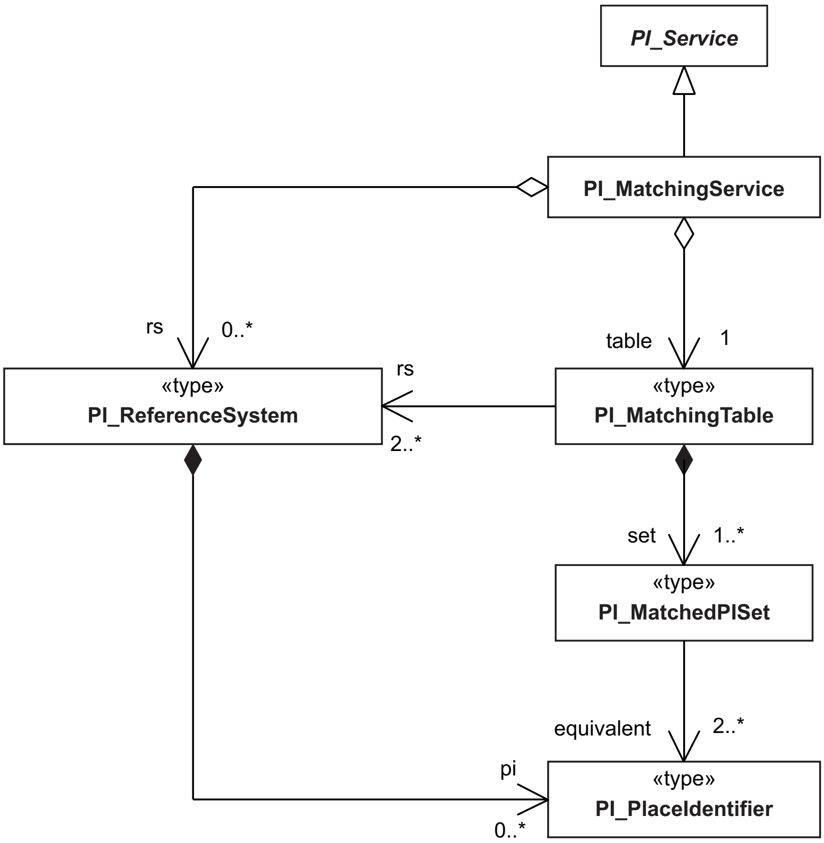

Figure 9 — PI matching service

A PI_MatchingService defines the following operations:

a) Matched PI Insertion

Add a set of matched PI instances that identify the same place to a PI matching table in this service.

b) Matched PI Retrieval

Search and retrieve appropriate matched PI instances to a PI user from a PI matching table.

c) Matched PI Deletion

Delete a set of matched PI instances from a PI matching table.

d) Service Metadata Acquisition

Provide service metadata about the PI_MatchingService to a PI user. A list of reference systems found in the PI matching table is also included in the service metadata.

7.3.3 PI_RSService

The conceptual structure of a PI reference system service is shown in Figure 10. A PI_RSService provides registration and management for an RS that is necessary for the understanding of a domain of PI instances.

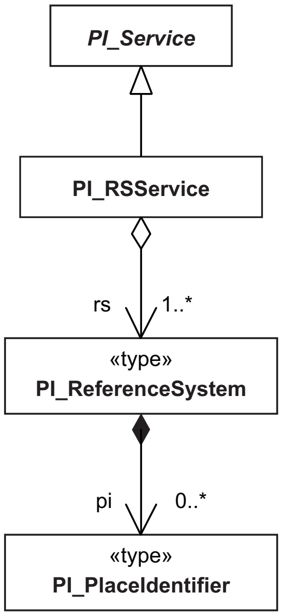

Figure 10 — PI reference system service

A PI_RSService defines the following operations:

a) RS Creation

Add a new reference system.

b) RS Retrieval

Retrieve RS instances and associated PI instances that meet the specified retrieval criteria

c) RS Update

Update a registered reference system using one of the following methods:

— rewrite the description of the reference system.

— add or delete a PI in this reference system.

d) RS Deletion

Delete the registered reference system. Deletion of a reference system deletes all matched PI instances in that reference system.

e) Service Metadata Acquisition

Provide service metadata about the PI_RSService to a PI user. A list of reference systems found in the PI_RSService is also included in the service metadata.

8 Interfaces for the PI platform

8.1 Overview

A PI matching service and RS service shall conform to the interfaces specified in Clause 8 to make use of the PI concepts specified in this reference model.

An overview of the interfaces is shown in Table 2. The interfaces are classified by purpose in terms of the functions they perform: "Create," "Retrieve," "Update," and "Delete."

Table 2 — Overview of PI Interfaces

Interface name	Purpose	Service	Description
AddMatchedPISet	Create	PI_MatchingService	Add a set of matched PI instances to the PI matching table.
GetMatchedPISet	Retrieve	PI_MatchingService	Return a set of PI instances that identifies the same place.
DeleteMatchedPISet	Delete	PI_MatchingService	Delete a set of matched PI instances from the PI matching table.
CreateRS	Create	PI_RSService	Add a new RS.
AddPI	Create	PI_RSService	Add a PI instance to the specified RS.
GetRS	Retrieve	PI_RSService	Return a list of reference systems that meets the retrieval conditions.
GetPI	Retrieve	PI_RSService	Return a list of PI instances that meets the retrieval conditions.
UpdateRS	Update	PI_RSService	Update the definition of the specified RS.
DeleteRS	Delete	PI_RSService	Delete an RS and associated PI instances.
DeletePI	Delete	PI_RSService	Delete a PI instance from the RS.
GetCapabilities	Retrieve	PI_MatchingService PI_RSService	Acquire service metadata from a PI matching service or RS service.

8.2 Interface for PI matching service

8.2.1 Introduction

PI_MatchingService shall use the interfaces specified for PI matching registration and PI retrieval using the "equivalent" relations among PI instances.

There are four interfaces required by the PI Matching Service:

a) Matched PI Insertion (8.2.2);

b) Matched PI Retrieval (8.2.3);

c) Matched PI Deletion (8.2.4);

d) Service metadata acquisition from PI Matching Service (8.2.5).

An interface for updating a matched PI set can be performed by using the creation and deletion interfaces.

The PI_MatchingService performs creation and deletion of matched PI sets, and does not add or delete PI instances. Insertion and deletion of PI instances is performed by PI_RSService.

Acquisition of a list of RS instances managed by the PI matching table is performed by using the Service Metadata interface.

8.2.2 Matched PI insertion (AddMatchedPISet)

Matched PI insertion is an interface to register a set of PI instances that identify the same place. The details of this interface are shown in Table 3.

Table 3 — Insertion of Matched PI instances (AddMatchedPISet)

Function Name	AddMatchedPISet		
Synopsis	Add a set of PI instances that identify the same place (Matched PI instances) to the PI matching table.		
Parameters	**Name**	**Type**	**Description**
	pi	*PI[ ]*	set of PI instances
Returned Type	Boolean		
Restrictions	If the processing fails, the PI matching table database shall be returned to the previous state.		
Related Items	PI structure		

8.2.3 Matched PI retrieval (GetMatchedPISet)

Matched PI retrieval is an interface to retrieve a set of PI instances that identify the same place as the input PI. The details of this interface are shown in Table 4. The *rs* parameter and the *validPeriod* parameter are optional and are effective in confining the return value.

Table 4 — Retrieval of Matched PI instances (GetMatchedPISet)

Function Name	GetMatchedPISet		
Synopsis	Return a set of PI instances that identify the same place as the input PI, which is registered in the PI matching table.		
Parameters	**Name**	**Type**	**Description**
	pi	*PI*	source PI instance
	rs[0..*]	*RSIdentifier[ ]*	name of a reference system of a target PI instance
	validPeriod[0..1]	*TemporalExtent*	temporal extent of target PI instances
Returned Type	*PI[ ]*		
Restrictions	• Return an array with zero elements, when matched PI does not exist. • “rs” parameter and “validPeriod” are ignored if null. • Returned array includes the input PI.		
Related Items	PI structure, RSIdentifier structure, TemporalExtent structure		

8.2.4 Matched PI deletion (DeleteMatchedPISet)

Matched PI deletion is an interface to delete a set from the PI matching table. The details of this interface are shown in Table 5.

Table 5 — Deletion of PI matched PI instances (DeleteMatchedPISet)

Function Name	DeleteMatchedPISet		
Synopsis	Delete the specified set of PI instances from the PI matching table.		
Parameters	**Name**	**Type**	**Description**
	pi	*PI[ ]*	list of PI instances
Returned Type	Boolean		
Restrictions	If the processing fails, the PI matching database shall be returned to the previous state.		
Related Items	PI structure		

8.2.5 Service metadata acquisition from PI matching service (GetCapabilities)

The Service metadata acquisition interface enables a user to make an inquiry of what functions are offered by the PI_MatchingService and to receive that response from the service. Details of this interface are shown in Table 6.

Table 6 — Service Metadata Acquisition (GetCapabilities)

Function Name	GetCapabilities		
Synopsis	Acquire metadata from a PI matching service.		
Parameters	**Name**	**Type**	**Description**
	version	*String*	version of an interface Specify 1.0 when acquiring service metadata conforming to this International Standard. Use null to acquire the latest service metadata from a service.
	updateSequence	*String*	an updated sequential number Specify the sequential number of the service metadata cached by a requester. Use null to acquire the latest service metadata from a service.
Returned Type	*PIMatchingServiceMetadata*		
Restrictions	• Acquire metadata from a PI matching service. • The updated sequential number shall be a character string which can be freely specified provided that a service is able to understand the increment or decrement value. The following are the returned results based on the updated sequential number given in an argument and the updated sequential number maintained in a service. • Parameter value and service comparisons, with return results are shown below: a) parameter < service Return the latest service metadata from a service b) parameter = service Return null c) parameter > service Return an exception “InvalidUpdateSequence” If the parameter is not comparable “InvalidUpdateSequence” is returned.		
Related Items	PIMatchingServiceMetadata structure		

8.3 Interface for PI reference system service

8.3.1 Introduction

PI_RSService shall implement the following eight interfaces related with RS registration and RS retrieval.

a) Reference System Creation (8.3.2),

b) Reference System Retrieval (8.3.3),

c) Reference System Update (8.3.4),

d) Reference System Deletion (8.3.5),

e) PI Insertion (8.3.6),

f) PI Retrieval (8.3.7),

g) PI Deletion (8.3.8),

h) Service Metadata Acquisition from Reference System Service (8.3.9).

A PI_RSService performs not only creation and deletion of RS but also insertion and deletion of PI instances belonging to the RS. The acquisition of RS instances is performed using Service Metadata Acquisition interface.

8.3.2 Reference System Creation (CreateRS)

Reference System Creation is an interface to create a reference system. The details of this interface are shown in Table 7.

Table 7 — Creation of a reference system (CreateRS)

Function Name	CreateRS		
Synopsis	Create a new reference system.		
Parameters	**Name**	**Type**	**Description**
	rs	*RS*	reference system
Returned Type	Boolean		
Restrictions	If the processing fails, the reference system database shall be returned to the previous state.		
Related Items	RS		

8.3.3 Reference System Retrieval (GetRS)

Details of an interface to search for an RS are shown in Table 8.

Table 8 — RS retrieval (GetRS)

Function Name	GetRS		
Synopsis	Retrieve registered RS instances.		
Parameters	**Name**	**Type**	**Description**
	name	*RSIdentifier*	name of this reference system
	domainOfValidity[0..*]	*Extent[ ]*	extent of this reference system
	definition[0..1]	*String*	definition of this reference system
	pi[0..*]	*PI[ ]*	list of PI instances included in this reference system
Returned Type	RS[ ]		
Restrictions	Return an array with zero elements, when there are no successfully retrieved RS instances.		
Related Items	RSIdentifier, Extent, PI		

8.3.4 Reference System Update (UpdateRS)

Details of an interface to update a reference system are shown in Table 9.

Table 9 — Reference System Update (UpdateRS)

Function Name	UpdateRS		
Synopsis	Update information of RS.		
Parameters	**Name**	**Type**	**Description**
	rs	*RSIdentifier*	name of the target reference system
	information	*String*	description of the reference system
Returned Type	Boolean		
Restrictions	If the processing fails, the reference system database shall be returned to the previous state.		
Related Items	RSIdentifier		

8.3.5 Reference System Deletion (DeleteRS)

Details of an interface to delete a specified reference system are shown in Table 10. PI instances belonging to the specified reference system are also deleted through this interface.

Table 10 — Reference System Deletion (DeleteRS)

Function Name	DeleteRS		
Synopsis	Delete a specified reference system and related PI instances.		
Parameters	**Name**	**Type**	**Description**
	rs	*RSIdentifier*	name of the target reference system
Returned Type	Boolean		
Restrictions	If the processing fails, the reference system database shall be returned to the previous state.		
Related Items	RSIdentifier		

8.3.6 PI insertion (AddPI)

Details of interface to add PI to a specified reference system are shown in Table 11.

Table 11 — PI insertion (AddPI)

Function Name	AddPI		
Synopsis	Insert a PI to a specified reference system.		
Parameters	**Name**	**Type**	**Description**
	rs	*RSIdentifier*	name of a reference system for insertion into
	pi	*PI*	PI instance to be inserted
Returned Type	Boolean		
Restrictions	• If the processing fails, the reference system database shall be returned to the previous state. • The value of “rs” shall be equal to the value of “rs” in the PI. • If defined, the valid period of the PI shall be included in the temporal extent of the reference system.		
Related Items	PI, RSIdentifier		

8.3.7 PI retrieval (GetPI)

Details of an interface to search for a PI in a reference system are shown in Table 12.

Table 12 — PI retrieval (GetPI)

Function Name	GetPI		
Synopsis	Retrieve registered PI instances whose values are perfectly matched to the input value.		
Parameters	**Name**	**Type**	**Description**
	value	*String*	value of PI instance
	rs	*RSIdentifier*	name of a reference system
	validPeriod	*TemporalExtent*	valid period of a PI instance
Returned Type	*PI[ ]*		
Restrictions	Return an array with zero elements, when there are no successfully retrieved PI instances.		
Related Items	RSIdentifier, TemporalExtent		

8.3.8 PI deletion (DeletePI)

Details of an interface to delete a PI from its reference system are shown in Table 13.

Table 13 — PI deletion (DeletePI)

Function Name	DeletePI		
Synopsis	Delete a PI from a reference system.		
Parameters	**Name**	**Type**	**Description**
	pi	*PI*	PI to be deleted
Returned Type	Boolean		
Restrictions	• If the processing fails, the reference system database shall be returned to the previous state. • The attribute “rs” in the PI shall be populated with a reference system name.		
Related Items	PI		

8.3.9 Service metadata acquisition from reference system service (GetCapabilities)

Details of an interface to acquire service metadata from a reference system service are shown in Table 14.

Table 14 — Acquisition of service metadata

Function Name	GetCapabilities		
Synopsis	Acquire metadata from an RS service.		
Parameters	**Name**	**Type**	**Description**
	version	*String*	version of an interface Specify 1.0 when acquiring service metadata conforming to this International Standard. Use null to acquire the latest service metadata from a service.
	updateSequence	*String*	an updated sequential number Specify the sequential number of the service metadata cached by a requester. Use null to acquire the latest service metadata from a service.
Returned Type	*RSServiceMetadata*		
Restrictions	• Acquire metadata from an RS service. • The updated sequential number shall be a character string which can be freely specified provided that a service is able to understand the increment or decrement value. The following are the returned results based on the updated sequential number given in an argument and the updated sequential number maintained in a service. • Parameter value and service comparisons, with return results are shown below: — parameter < service Return the latest service metadata from a service — parameter = service Return null — parameter > service Return an exception “InvalidUpdateSequence” • If the parameter is not comparable “InvalidUpdateSequence” is returned.		
Related Items	RSServiceMetadata structure		

8.4 Structures

8.4.1 Introduction

Structures that are used by the interfaces defined in 8.2 and 8.3 are specified in 8.4.2 to 8.4.19.

8.4.2 PI structure

Details of the PI structure based on PI_PlaceIdentifier are shown in Table 15.

Table 15 — PI structure

Purpose	PI		
Elements	**Name**	**Type**	**Description**
	value	*String*	a series of characters in an identifier
	rs[0..1]	*RSIdentifier*	reference system that defines this value to be identifiable.
	validPeriod[0..1]	*TemporalExtent*	valid period of this PI
Related Items	TemporalExtent		

8.4.3 RS structure

Details of the RS structure based on PI_ReferenceSystem are shown in Table 16.

Table 16 — RS structure

Purpose	Reference System		
Elements	**Name**	**Type**	**Description**
	name	*RSIdentifier*	name of this reference system
	domainOfValid-ity[0..*]	*Extent[ ]*	extent of this reference system
	definition[0..1]	*String*	definition of this reference system
	pi[0..*]	*PI[ ]*	list of PI instances included in this reference system.
Related Items	RSIdentifier, Extent, PI		

8.4.4 MatchedPISet structure

Details of the matched PI set structure are shown in Table 17.

Table 17 — MatchedPISet structure

Purpose	Collection of PI instances that form a set		
Elements	**Name**	**Type**	**Description**
	set	*PI[ ]*	matched PI set that identifies a single place
Related Items	PI		

8.4.5 PIMatchingServiceMetadata structure

The details of the PIMatchingServiceMetadata structure are shown in Table 18.

Table 18 — PIMatchingServiceMetadata structure

Purpose	Service metadata for PI Matching Service		
Elements	**Name**	**Type**	**Description**
	version	*String*	version of a PI matching service Specify 1.0 when acquiring service metadata conforming to this International Standard.
	rs	*RS[ ]*	reference system information
	matchedPIsets	*MatchedPISet[ ]*	matched PI sets
	canUpdateMatchedPIRecord	*Boolean*	Boolean value for the registration of a matched PI set
	updateSequence	*String*	sequential number
Related Items	RS, MatchedPISet		

8.4.6 RSServiceMetadata structure

The details of the RSServiceMetadata structure are shown in Table 19.

Table 19 — RSServiceMetadata structure

Purpose	Metadata of reference system services		
Elements	**Name**	**Type**	**Description**
	version	*String*	version of this reference system service Specify 1.0 in the service metadata as per this International Standard.
	canUpdateRS	*Boolean*	registration possibility of a reference system
	canUpdatePI	*Boolean*	registration possibility of PI
	rs	*RSIdentifier[ ]*	a list of reference systems included in this reference system service
	updateSequence	*String*	update sequence
Related Items	Although the update sequence is a random character string, to achieve the cache functionality, it is necessary for the service to judge the size of the value of the update sequence.		

8.4.7 RSIdentifier structure

The details of the RSIdentifier structure are shown in Table 20.

Table 20 — RSIdentifier structure

Purpose	Reference system identifier		
Elements	**Name**	**Type**	**Description**
	Code	*String*	alphanumeric value identifying an instance in the namespace
	codeSpace[0..1]	*String*	name or identifier of the person or organization responsible for the namespace

8.4.8 Extent structure

Details of the Extent structure are shown in Table 21.

Table 21 — Extent structure

Purpose	Extent		
Elements	**Name**	**Type**	**Description**
	description	*String*	description
	geographicElement	*GeographicExtent[ ]*	geographic extent
	temporalElement	*TemporalExtent[ ]*	duration
	verticalElement	*VerticalExtent[ ]*	perpendicular extent
Related Items	GeographicExtent structure, TemporalExtent structure, and VerticalExtent structure		

8.4.9 GeographicExtent structure

Details of the GeographicExtent structure are shown in Table 22.

Table 22 — GeographicExtent structure

Purpose	Geographic extent		
Elements	**Name**	**Type**	**Description**
	boundingPolygon	*Polygon*	polygon
	boundingBox	*GeographicBoundingBox*	box
	description	*String*	descriptive
Restrictions	Assign a value other than null to only one of the boundingPolygon, boundingBox, or description.		
Related Items	Polygon structure, GeographicBoundingBox structure		

8.4.10 TemporalExtent structure

Details of the TemporalExtent structure are shown in Table 23.

Table 23 — TemporalExtent structure

Purpose	Time range		
Elements	**Name**	**Type**	**Description**
	begin	*DateTime*	start time Use null to specify time periods before the end time.
	end	*DateTime*	end time Use null to specify time periods after the start time.
Restrictions	• Either Begin or End shall have a value other than null. • If values are to be assigned to both Begin and End, then it shall be ensured that Begin=< End.		

8.4.11 VerticalExtent structure

Details of the VerticalExtent structure are shown in Table 24.

Table 24 — VerticalExtent structure

Purpose	Vertical extent		
Elements	**Name**	**Type**	**Description**
	minimumValue	*Double*	the lowest value
	maximumValue	*Double*	the highest value
	unitOfMeasure	*String*	unit of measure
	verticalDatum	*VerticalDatum*	vertical datum
Related Items	VerticalDatum structure		

8.4.12 Polygon structure

Details of the Polygon structure are shown in Table 25.

Table 25 — Polygon structure

Purpose	Polygon		
Elements	**Name**	**Type**	**Description**
	exterior	*Ring*	exterior Use of null is not allowed here.
	interior	*Ring[ ]*	interior Assigned if defining an island polygon. Assign null if polygon does not contain any islands.
Restrictions	• Define an island polygon (a polygon having one exterior and multiple interiors). • All the coordinate reference systems shall be the same for both the exterior and any interior polygons.		
Related Items	Ring structure		

8.4.13 Ring structure

Details of the Ring structure are shown in Table 26.

Table 26 — Ring structure

Purpose	Ring		
Elements	**Name**	**Type**	**Description**
	coords	*Position[ ]*	coordinate line
Restrictions	• Rings define closed structures which do not intersect. • The origin point and the end point shall be the same coordinates – they shall be joined. • The coordinates of line which make up the ring shall consist of more than three points.		
Related Items	Position structure		

8.4.14 Position structure

Details of the Position structure are shown in Table 27.

Table 27 — Position structure

Purpose	Position		
Elements	**Name**	**Type**	**Description**
	coord	*Double[ ]*	coordinate
	crs	*String*	spatial reference system by coordi-nates
Restrictions	Store the values using a coordinate array in the sequence in which they are defined in the spatial reference system.		
Related Items			

8.4.15 GeographicBoundingBox structure

Details of the GeographicBoundingBox structure are shown in Table 28.

Table 28 — GeographicBoundingBox structure

Purpose	Geographic boundary		
Elements	**Name**	**Type**	**Description**
	min	*Double[ ]*	minimum coordinates
	max	*Double[ ]*	maximum coordinates
	crs	*String*	coordinate reference system
Restrictions	• Define the region by a minimum-bounding rectangle with the positions of the minimum and maximum coordinates. • Store the values using a coordinate array in the sequence in which they are defined in the spatial reference system. • Each coordinate shall satisfy the following condition: min[i] =< max[i] (where "i" is the index of the array).		

8.4.16 Ellipsoid structure

Details of the Ellipsoid structure are shown in Table 29.

Table 29 — Ellipsoid structure

Purpose	Ellipsoid		
Elements	**Name**	**Type**	**Description**
	ellipsoidID	*String*	identifier of an ellipsoid for a datum
	identifier	*RSIdentifier[ ]*	an identifier which references elsewhere the object's defining information; alternatively an identifier by which this object can be referenced
	name	*RSIdentifier[ ]*	the primary name by which this object is identified
	alias	*String[ ]*	an alternative name by which this object is identified
	secondDefiningParameter	*SecondDefiningParameter*	definition of the second parameter that describes the shape of this ellipsoid
	remark	*String*	comments on or information about this object, including data source information
Related Items	SecondDefiningParameter structure, RSIdentifier structure		

8.4.17 SecondDefiningParameter structure

Details of the SecondDefiningParameter structure are shown in Table 30.

Table 30 — SecondDefiningParameter structure

Purpose	Second Parameter that describes the shape of an ellipsoid		
Elements	**Name**	**Type**	**Description**
	inverseFlattening	*Scale*	inverse flattening value of the ellipsoid
	semiMinorAxis	*Length*	length of the semi-minor axis of the ellipsoid
	isSphere	*Boolean*	attribute has the value "true" if the figure is a sphere The ellipsoid is degenerate and is actually a sphere. The sphere is completely defined by the semi-major axis, which is the radius of the sphere.

8.4.18 VerticalDatum structure

Details of the VerticalDatum structure are shown in Table 31.

Table 31 — VerticalDatum structure

Purpose	Datum		
Elements	**Name**	**Type**	**Description**
	identifier	*RSIdentifier[ ]*	an identifier which references elsewhere the object's defining information; alternatively an identifier by which this object can be referenced
	name	*RSIdentifier[ ]*	the primary name by which this object is identified
	alias	*String[ ]*	an alternative name by which this object is identified
	anchorDefinition	*String*	the datum definition – a description, possibly including coordinates of an identified point or points, of the relationship used to anchor the coordinate system to the Earth or alternate object
	realizationEpoch	*Date*	the time after which this datum definition is valid
	domainofValidity	*Extent*	area or region or time frame in which this datum is valid
	scope	*String*	description of usage, or limitations of usage, for which this datum is valid. If unknown, enter "not known"
	remark	*String*	comments on or information about this object, including data source information
	primeMeridian	*PrimeMeridian*	prime meridian
	ellipsoid	*Ellipsoid*	ellipsoid
Related Items	Extent structure, PrimeMeridian structure, and Ellipsoid structure.		

8.4.19 PrimeMeridian structure

The details of the PrimeMeridian structure are shown in Table 32.

Table 32 — PrimeMeridian structure

Purpose	Prime meridian		
Elements	**Name**	**Type**	**Description**
	identifier	*RSIdentifier[ ]*	an identifier which references elsewhere the object's defining information; alternatively an identifier by which this object can be referenced
	name	*RSIdentifier[ ]*	the primary name by which this object is identified
	alias	*String[ ]*	an alternative name by which this object is identified
	greenwichLongtitude	*Angle*	Prime Meridian Greenwich longitude
	remark	*String*	comments on or information about this object, including data source information

8.4.20 Angle structure

The details of the Angle structure are shown in Table 33.

Table 33 — Angle structure

Purpose	Angle		
Elements	**Name**	**Type**	**Description**
	uom	*String*	the reference quantities used to express the value of angles
	value	*Double*	value of this angle

8.4.21 Length structure

The details of the Length structure are shown in Table 34.

Table 34 — Length structure

Purpose	Length		
Elements	**Name**	**Type**	**Description**
	uom	*String*	the reference quantities used to express the value of the length
	value	*Double*	value of this length

8.5 Exception handling

8.5.1 Exception handling introduction

Information related to the exception handling methods used in the interfaces specified in 8.2 and 8.3 are shown in 8.5.2 to 8.5.5.

8.5.2 InvalidUpdateSequence

Details of the exception for an InvalidUpdateSequence are shown in Table 35.

Table 35 — InvalidUpdateSequence

Purpose	Anomaly in the update sequence of Service metadata.
Description	This exception occurs when there is an anomaly in the size of update sequence or when the size relationship cannot be successfully measured. For the method of expressing the exceptions, follow the common specifications which are defined for each process.
Related Items	GetCapabilities function

8.5.3 InvalidArgument

Details of the exception for an InvalidArgument are shown in Table 36.

Table 36 — InvalidArgument

Purpose	Exception when an incorrect value is assigned to the argument of the function.
Description	This exception occurs when null is assigned to an argument where null is not permitted. For the method of expressing the exceptions, follow the common specifications which are defined for each process.

8.5.4 InvalidValue

Details of the exception for an InvalidValue are shown in Table 37.

Table 37 — InvalidValue

Purpose	Exception when an incorrect value is assigned to the argument of the structure.
Description	This exception occurs when a null or incorrect value is assigned when such a value is not allowed in a structure element. However the InvalidArgument exception occurs when a structure is specified in the argument of the function. For the method of expressing the exceptions, follow the common specifications which are defined for each process.

8.5.5 UpdateFailure

Details of the exception for an UpdateFailure are shown in Table 38.

Table 38 — UpdateFailure

Purpose	Exception when a database operation (insert, update or delete) fails.
Description	This exception occurs when an insert, update or delete operation on a database fails. For the method of expressing the exceptions, follow the common specifications which are defined for each process.

Annex A
(normative)

Abstract test suite

A.1 Conformance tests for Semantics

A.1.1 PI structure

a) Test Purpose: Verify an adequate PI structure.

b) Test Method: Inspect to ensure that the instance of PI_PlaceIdentifier has the content of RS, value and time.

c) References: 7.2.2

d) Test Type: Capability.

A.1.2 Uniqueness of naming

A.1.2.1 PI_PlaceIdentifier

a) Test Purpose: Verify an adequate value in the PI_PlaceIdentifier instance. Each value shall be identified uniquely in each RS.

b) Test Method: Inspect to ensure that value of PI_PlaceIdentifier instance in the PI_ReferenceSystem is uniquely identified.

c) References: 7.2.2

d) Test Type: Capability.

A.1.2.2 PI_ReferenceSystem

a) Test Purpose: Verify an adequate PI_ReferenceSystem instance. Each PI_ReferenceSystem instance shall be uniquely identified.

b) Test Method: Inspect to ensure that names of PI_ReferenceSystem instances in the PI_RSService are uniquely identified.

c) References: 7.2.3

d) Test Type: Capability.

A.2 Conformance tests for Data

A.2.1 Data in the PI_MatchingService

A.2.1.1 PI_MatchingTable

a) Test Purpose: Verify an adequate PI_MatchingTable instance. Each PI_MatchingTable instance shall consist of PI_MatchedPISet, and shall be stored in the PI matching service based on the requirements specified in this International Standard.

b) Test Method: Inspect to ensure that PI_MatchingTable instance in the PI matching service follows requirements specified in this International Standard.

c) References: 7.2.4 and 7.2.5

d) Test Type: Capability.

A.2.1.2 PI_MatchedPISet

a) Test Purpose: Verify an adequate PI_MatchedPISet instance. Each PI_MatchedPISet instance shall be stored in the PI matching service based on the requirement specified in this International Standard.

b) Test Method: Inspect to ensure that PI_MatchedPISet instance in the PI matching service follows requirements specified in this International Standard.

c) References: 7.2.5

d) Test Type: Capability.

A.2.2 Data in the PI_RSService

A.2.2.1 PI_PlaceIdentifier

a) Test Purpose: Verify an adequate PI_PlaceIdentifier instance. Each PI_PlaceIdentifier instance shall be stored in the PI reference system service based on the requirements specified in this International Standard.

b) Test Method: Inspect to ensure that PI_PlaceIdentifier instance in the PI reference system service follows requirements specified in this International Standard.

c) References: 7.2.2

d) Test Type: Capability.

A.2.2.2 PI_ReferenceSystem

a) Test Purpose: Verify an adequate PI_ReferenceSystem instance. Each PI_ReferenceSystem instance shall be stored in the reference system service based on the requirements specified in this International Standard.

b) Test Method: Inspect to ensure that PI_ReferenceSystem instance in the reference system service follows requirements specified in this International Standard.

c) References: 7.2.3

d) Test Type: Capability.

A.3 Conformance tests for Services

A.3.1 Interface implementation in PI matching service

A.3.1.1 Matched PI insertion interface

a) Test Purpose: Verify an adequate matched PI insertion interface implementation. Each matched PI insertion interface shall be implemented based on the requirements of this International Standard.

b) Test Method: Inspect to ensure that the content of request and response follows the requirement of this International Standard.

c) References: 8.2.2

d) Test Type: Capability.

A.3.1.2 Matched PI retrieval interface

a) Test Purpose: Verify an adequate matched PI retrieval interface implementation. Each matched PI retrieval interface shall be implemented based on the requirements of this International Standard.

b) Test Method: Inspect to ensure that the content of request and response follows the requirement of this International Standard.

c) References: 8.2.3

d) Test Type: Capability.

A.3.1.3 Matched PI deletion interface

a) Test Purpose: Verify an adequate matched PI deletion interface implementation. Each Matched PI deletion interface shall be implemented based on the requirements of this International Standard.

b) Test Method: Inspect to ensure that the content of request and response follows the requirement of this International Standard.

c) References: 8.2.4

d) Test Type: Capability.

A.3.1.4 Service metadata acquisition interface

a) Test Purpose: Verify an adequate service metadata acquisition interface implementation. Each service metadata acquisition interface shall be implemented based on the requirements of this International Standard.

b) Test Method: Inspect to ensure that the content of request and response follows the requirement of this International Standard.

c) References: 8.2.5

d) Test Type: Capability.

A.3.2 Interface implementation in reference system service

A.3.2.1 RS creation interface

a) Test purpose: to verify an adequate RS creation interface implementation. Each RS creation interface shall be implemented based on the requirements of this International Standard.

b) Test method: Inspect to ensure that the content of request and response follows the requirement of this International Standard.

c) References: 8.3.2

d) Test Type: Capability.

A.3.2.2 RS retrieval interface

a) Test purpose: to verify an adequate RS retrieval interface implementation. Each RS retrieval interface shall be implemented based on the requirements of this International Standard.

b) Test method: Inspect to ensure that the content of request and response follows the requirement of this International Standard.

c) References: 8.3.3

d) Test Type: Capability.

A.3.2.3 RS update interface

a) Test purpose: to verify an adequate RS update interface implementation. Each RS update interface shall be implemented based on the requirements of this International Standard.

b) Test method: Inspect to ensure that the content of request and response follows the requirement of this International Standard.

c) References: 8.3.4

d) Test Type: Capability.

A.3.2.4 RS deletion interface

a) Test purpose: to verify an adequate RS deletion interface implementation. Each RS deletion interface shall be implemented based on the requirements of this International Standard.

b) Test method: Inspect to ensure that the content of request and response follows the requirement of this International Standard.

c) References: 8.3.5

d) Test Type: Capability.

A.3.2.5 PI insertion interface

a) Test purpose: to verify an adequate PI insertion interface implementation. Each PI insertion interface shall be implemented based on the requirements of this International Standard.

b) Test method: Inspect to ensure that the content of request and response follows the requirement of this International Standard.

c) References: 8.3.6

d) Test Type: Capability.

A.3.2.6 PI retrieval interface

a) Test purpose: to verify an adequate PI retrieval interface implementation. Each PI retrieval interface shall be implemented based on the requirements of this International Standard.

b) Test method: Inspect to ensure that the content of request and response follows the requirement of this International Standard.

c) References: 8.3.7

d) Test Type: Capability.

A.3.2.7 PI deletion interface

a) Test purpose: to verify an adequate PI deletion interface implementation. Each PI deletion interface shall be implemented based on the requirements of this International Standard.

b) Test method: Inspect to ensure that the content of request and response follows the requirement of this International Standard.

c) References: 8.3.8

d) Test Type: Capability.

A.3.2.8 Service metadata interface

a) Test purpose: to verify an adequate service metadata Interface implementation. Each service metadata Interface shall be implemented based on the requirements of this International Standard.

b) Test method: Inspect to ensure that the content of request and response follows the requirement of this International Standard.

c) References: 8.3.9

d) Test Type: Capability.

A.4 Conformance tests for PI encoding

A.4.1 PI encoding using GML

a) Test Purpose: Verify an adequate PI Encoding using GML. Each PI Encoding using GML shall be implemented based on the requirements of this International Standard.

b) Test Method: Inspect to ensure that the content of request and response follows the requirement of this International Standard.

c) References: B.2

d) Test Type: Capability.

Annex B
(normative)

PI encoding using GML

B.1 Encoding Introduction

The following specifies the encoding of a PI_PlaceIdentifier in 7.2.2 based on GML (ISO 19136:2007).

B.2 PI Schema in GML

The element `pi:PI_PlaceIdentifier` is declared as follows:

```
<element name="PI_PlaceIdentifier" type="pi:PI_PlaceIdentifierType" substitutionGroup="gml:AbstractGML"/>
```

`pi:PI_PlaceIdentifierType` is derived from `gml:AbstractGMLType`, following the rule that GML requires all identifiable objects shall be derived from this type (see ISO 19136:2007, 7.2.2.2).

```
<complexType name="PI_PlaceIdentifierType">
    <complexContent>
        <extension base="gml:AbstractGMLType">
            <sequence>
                <element name="rs" type="xs:anyURI" minOccurs="0"/>
                <element name="value" type="xs:string"/>
                <element name="validPeriod" type="gml:TimePeriodType" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

B.3 PI instances

An instance of `pi:PI_PlaceIdentifier` may be described in an XML document as shown in the following example:

```
<PI_PlaceIdentifier gml:id="pi001">
    <rs>http://www.post.japanpost.jp/zipcode/</rs>
    <value>113-0001</value>
</PI_PlaceIdentifier>
```

The element `rs` points to the reference system of this PI.

If the reference system of the Place Identifiers is well known in the application domain, the element `rs` can be omitted:

```
<PI_PlaceIdentifier gml:id="pi001">
    <value>113-0001</ value>
</PI_PlaceIdentifier>
```

The client may retrieve PI instances (different Place Identifiers that identify the same place) through the PI matching service if provided. In this case, PI instances may be encoded as the reply of the request message as shown below:

```
<PI_PlaceIdentifier gml:id="pi001">
    <value>http://www.post.japanpost.jp/cgi-post/zipcode.php?post=113-0001</value>
</PI_PlaceIdentifier>
```

A PI can identify not only places in the real world but also those in a virtual world, as shown below:

```
<PI_PlaceIdentifier gml:id=”pi001”>
    <value srsName = ”URL”>http://www.isotc211.org/plenary29.xml</value>
    <pi:validPeriod gml:id=”time1”>
            <gml:beginPosition>2009-11-05</gml:beginPosition>
            <gml:endPosition>2009-11-06</gml:endPosition>
    </pi:validPeriod>
</PI_PlaceIdentifier>
```

In this example, `gml:duration` is used to describe the period of existence of a built structure in the real world:

```
<PI_PlaceIdentifier gml:id=”pi003”>
    <rs>bldgmgt.tokyo.go.jp</rs>
    <value>13103-2454543-A1-6893394-12B</value>
    <validPeriod gml:id=”time2”>
            <gml:duration>1960-04-01T00:00:00/2001-06-30T00:00:00</gml: duration>
    </validPeriod>
</PI_PlaceIdentifier>
```

Annex C
(informative)

PI encoding using 'tag' URI Scheme

C.1 Encoding Introduction

This Annex describes the encoding of a PI_PlaceIdentifier, as defined in 7.2.2, based on the 'tag' URI Scheme specification (IETF RFC 4151, 2005).

C.2 Expression of PI based on RFC 4151 ('tag' URI scheme)

If the PI is represented based on the 'tag' URI scheme specification, the structure is shown below:

tag:bldgmgt.tokyo.go.jp,2008-03-18:pi/13103-2454543-A1-6893394-12B

The following rules shall be observed if the PI is represented using the 'tag' URI scheme specification.

A Fully Qualified Domain Name (FQDN) after the Schema Name "tag": should be used.

In this example, *"bldgmgt.tokyo.go.jp"* is the reference system (RS). Based on the requirements of 'tag' URI scheme the RS is represented by a FQDN, which indirectly guarantees the uniqueness of the RS.

In this example *"2008-03-18"* refers to the *"validPeriod"* attribute, yet follows the structure mandated by the "date" definition of the 'tag' URI Scheme specification.

The value *"13103-2454543-A1-6893394-12B"* describes the *"value"* attribute and is preceded by "pi/" to identify the value as a PI.

Annex D
(informative)

PI encoding using Well Known Text (WKT)

D.1 Encoding introduction

This Annex describes the encoding of a PI_PlaceIdentifier, as defined in 7.2.2, based on the Well Known Text (WKT) format (ISO 19125-1:2004).

D.2 Expression of a PI based on WKT

A simple encoding of a PI instance in WKT is shown below:

PI[RS[“bldgmgt.tokyo.go.jp”],VALUE[“13103-2454543-A1-6893394-12B”],PERIOD[“2008-03-18T14:07+09”]]

The following rules should be observed if the PI is encoded using WKT.

The *RS* keyword describes the name of the reference system. In this example, the RS is represented by a Fully Qualified Domain Name (FQDN), which indirectly guarantees the uniqueness of the RS.

The *VALUE* keyword describes the *“value”* attribute.

The *PERIOD* keyword refers to the *“validPeriod”* attribute and describes the temporality of the *“value”* conforming to the structure defined by ISO 19108:2002.

Annex E
(informative)

Use case examples

E.1 Introduction

To more easily understand the concepts and merits of the PI Architecture, a selection of use case examples has been compiled. Varying levels of detail are provided in each example. While the aim of each example is not to provide a detailed implementation design, these use case examples help to illustrate the concepts of the PI Architecture and provide ideas suitable for implementation.

The main use of the PI Architecture is to extend existing systems, where data is not easily shared. The flexible design of the PI allows data from existing systems to be structured and shared through an implementation of the PI Architecture.

The example presented in E.2 is a simple, self contained spatial bookmark system. It provides an easy to understand scenario and describes the merits of Place Identifiers. The example presented in E.3 is a more complex scenario which describes multiple, distributed databases or registries of Place Identifiers. A Conceptual Description of PI Data Storage is presented in E.4. The extraction of PI data instances from XML documents is described in E.5. Finally, E.6 describes “localized or personal reference systems” within the aspect of user communities.

E.2 Spatial Bookmarks

In this use case example an Internet Service Provider (ISP) has created a “Spatial Bookmark” application for users. The premise of this application allows a user to create a bookmark (record) for a shop, restaurant or general “place” they like to visit. Use of the PI concept enhances this application by enabling the bookmark to have a spatial reference to the place, in addition to allowing the contents of the bookmark to be discovered and shared between other systems.

To build the “Spatial Bookmark” application, the ISP has contracted access to both spatial data and geocoding functions as “services” from an Application Service Provider (ASP).

Considering the example of a newly opened café, a user of the “Spatial Bookmark” application would create a bookmark and add content about their visit to the café. In addition to that content, the application would assist the user to input a Place Identifier in the form of an address, an identified position on a background map, or the URL to the website of the café. The application would store the content and place identifier as a matched PI set. Additional functions of the application allow other users to discover the content about the new café. When building the “Spatial Bookmark” application the ISP made use of International Standards, therefore the encoded PI instances are available for sharing with other users and systems.

E.3 Multiple distributed registries

In this use case example a building management company has chosen to make some of the information about the buildings they manage available as Place Identifiers. These well formed, matched PI instances contain the name of the building and a street address. Prior to creation of the PI instances by the building management company, other well formed PI instances consisting of the contents of a corporate database were created by a business listing company. Another set of matched PI instances were also created, this time by the telephone company, containing all non-private listings and their phone numbers. Finally, another set of matched PI instances were created with restricted permissions by a private user. These PI instances consisted of marketing leads and were defined with a company name and possible sales volumes.

The service concepts defined by the PI Architecture, when implemented, may be used to examine the PI instances and create matched linkages between them. Various services may additionally examine the well formed matched PI instances and create new Place Identifiers, such as extracting latitude and longitude values based on the street address instances, or making linkages between the telephone instances and other instances where possible. Other services or users may further make use of the matched PI sets to facilitate data discovery and examine new linkages which were not possible to examine before.

Given the set of matched PI instances described above, a salesperson may want to obtain addresses based on company names they have listed in their sales leads. These linkages may be examined to reveal even more associated information, such as the phone number and additional background about each of the businesses that match with the definitions in the marketing leads. In addition to the linkages that are created between the matched Place Identifiers, the spatially enabled linkages can now be examined from a visual perspective, using that data inside of a GIS or online mapping system. Figure E.1 shows these possible service scenarios.

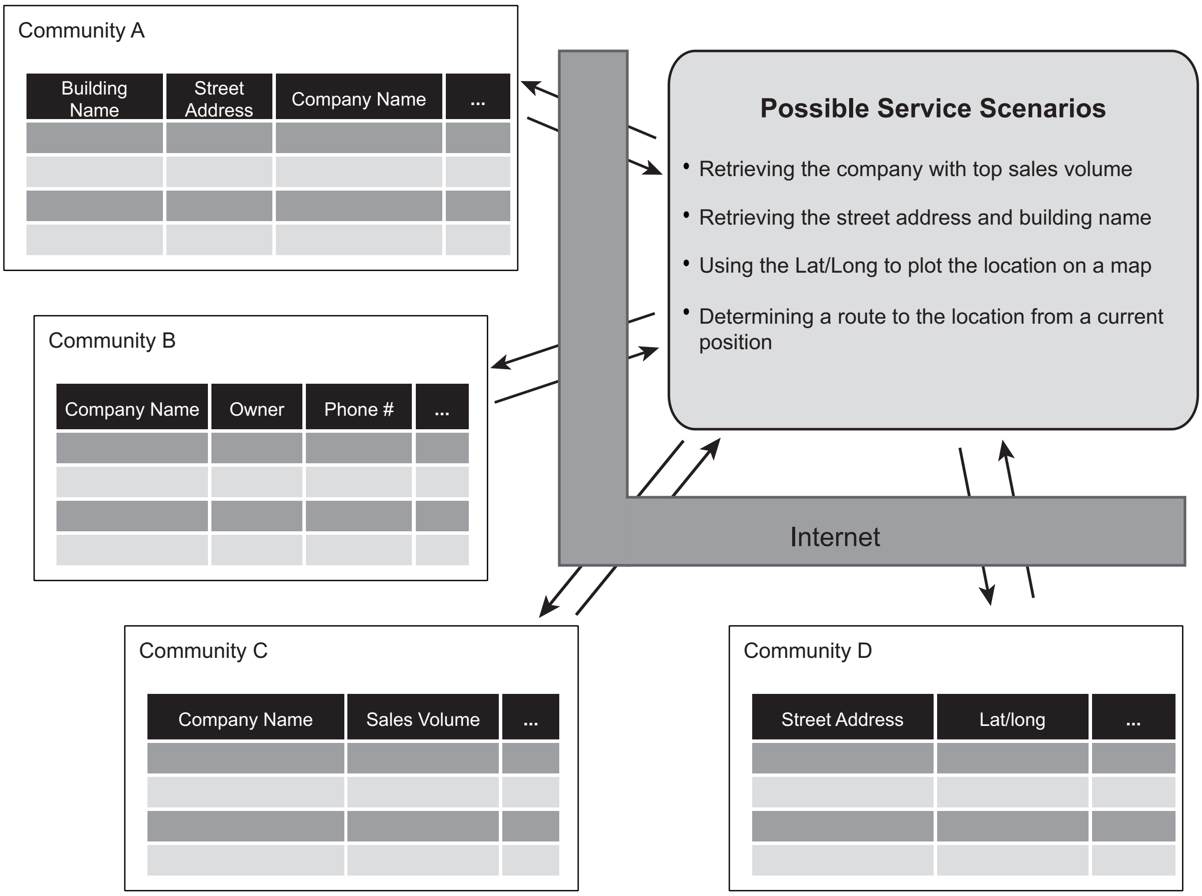

Figure E.1 — Conceptual view of multiple registries

E.4 Conceptual description of PI data storage

There are various implementation methods which can be used for the aggregate storage of PI data instances. One example, which helps to clearly illustrate the relationship between the PI instances and RS definitions is a type of table. Here, two or more values in a row (horizontally grouped) can be referred to as a matched PI set. The fields or columns (vertically grouped) comprise values of the same reference system (RS). It is not necessary for each cell to have a value, however at least one cell will be populated due to the basic structure. Figure E.2 shows the conceptual example of PI data storage.

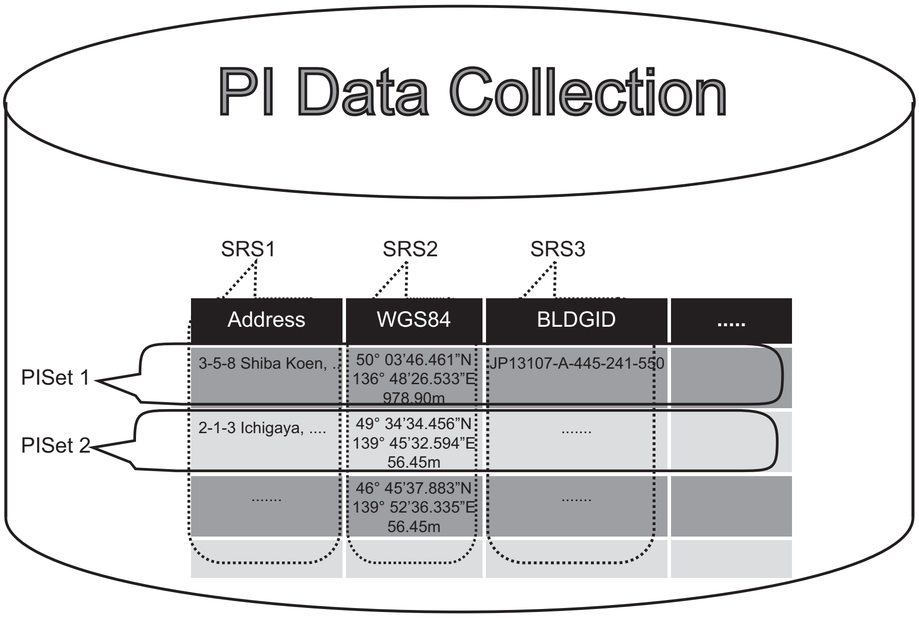

Figure E.2 — Conceptual view of PI data storage

E.5 Extraction of PI data instances from XML documents

In the previous examples various PI matching services that would be used to prepare linkages between other matched PI sets have been described. However, there are many cases where data already exists in matched form stored in pre-existing databases and geographic information systems. As this data is already matched, re-matching using the PI Architecture is not necessary.

E.6 Localized or personal reference systems

The PI Architecture does not designate the use of any specific reference systems (RSs). Instead, the flexible and extensible design of the PI Architecture enables the registration of any RS. While it is envisioned that most RSs will be globally recognized and possibly "approved" by the various organizations which "manage" them, it is also possible to register a type of "localized or personal reference system." When there is no necessity of using an "approved" RS to express a place, in most cases, people would then express places according to a personal reference system. Referred here as MyPersonalRS, this RS may be registered by a highly localized community or even a single person. Place Identifiers stored within MyPersonalRS can be envisioned as having less security or management requirements and are more open thus more readily available for sharing and PI matching with other Place Identifiers. This openness creates a sense of highly dynamic content, built from the sharing of Place Identifiers from many other MyPersonalRS.

For example a user stores place information in their MyCafe RS for the cafes they have visited. However, if this RS is openly shared, the usability and value of their MyCafe RS will readily grow with the inclusion of matched Place Identifiers from the other similar personal MyCafe RS from other users. In turn, the contributions from MyCafe RS will also add value to other users personal RS. Thus, it is expected that PI architecture promotes circulation of the place information and plays a large role in creation of a new information retrieval service. This flexibility is a key concept within the PI Architecture and will be a merit for developers and users alike.

Bibliography

[1] ISO 19108:2002, *Geographic information — Temporal schema*

[2] ISO 19109:2005, *Geographic information — Rules for application schema*

[3] ISO 19119:2005, *Geographic information — Services*

[4] ISO 19125-1:2004, *Geographic information — Simple feature access — Part 1: Common architecture*

[5] ISO 19128:2005, *Geographic information — Web map server interface*

[6] ISO 19132:2007, *Geographic information — Location-based services — Reference model*

[7] ISO 19135:2005, *Geographic information — Procedures for item registration*

[8] ISO/IEC 19501:2005, *Information technology — Open Distributed Processing — Unified Modeling Language (UML) Version 1.4.2*

[9] The Concise Oxford Dictionary of Current English. Thompson, D. (Editor), ISBN 0198613202, Oxford University Press, USA, 9 ed. 1995

[10] Internet Engineering Task Force (IETF). 2005, *'tag' URI Scheme RFC 4151*, The Internet Society, Washington D.C. Available at http://www.ietf.org/rfc/rfc4151.txt

ICS 35.240.70

Price based on 41 pages