Open Geospatial Consortium

Submission Date: 2016-11-08

Approval Date: 2017-03-24

Publication Date: 2018-04-15

External identifier of this OGC® document: http://www.opengis.net/doc/IS/GeoAPI/3.0.1

Internal reference number of this OGC® document: 09-083r4

Version: 3.0.1

Category: OGC® Implementation Standard

Editor: Adrian Custer

OGC GeoAPI 3.0.1 Implementation Standard – with Corrigendum

Document type:	OGC® Standard
Document subtype:	
Document stage:	Approved
Document language:	English

Contents

Erreur ! Cela ne correspond pas à un niveau de titre valide.

i. Abstract

The GeoAPI Implementation Standard defines, through the GeoAPI library, a Java language application programming interface (API) including a set of types and methods which can be used for the manipulation of geographic information structured following the specifications adopted by the Technical Committee 211 of the International Organization for Standardization (ISO) and by the Open Geospatial Consortium (OGC). This standard standardizes the informatics contract between the client code which manipulates normalized data structures of geographic information based on the published API and the library code able both to instantiate and operate on these data structures according to the rules required by the published API and by the ISO and OGC standards.

ii. Preface

This GeoAPI standard evolved from a long effort at the Open Geospatial Consortium (OGC) and in the free software community focused on developing a library of interfaces defining a coherent data model for the manipulation of geospatial data based on the data model defined in the OGC Abstract Specification. The GeoAPI library has been developed to facilitate the creation of interoperable, standards compliant, Java language software.

The GeoAPI interface library originates with the publication in January 2001 of the implementation specification OGC 01-009 *Coordinate Transformation Services* Revision 1.00 (Martin Daly, ed.) which included a set of interfaces written in the Java language and in the `org.opengis` namespace. The GeoAPI project started in 2003 as an effort from several contributors to develop a set of Java language interfaces which could be shared between several projects. The GeoAPI project subsequently considered the interfaces of OGC 01-009 as version 0.1 of the GeoAPI library and started working on GeoAPI 1.0 in collaboration with developers writing the OGC specification *Geographic Objects*. Subsequently, the Open Geospatial Consortium jettisoned its own Abstract Specifications and adopted, as the basis for further work, the standards developed by the Technical Committee 211 of the International Organization for Standardization (ISO) in its ISO 19100 series. The GeoAPI project therefore realigned its library with those standards. In 2003, version 1.0 of the GeoAPI library was released to match the release of the first public draft of the implementation specification OGC 03-064 GO-1 *Application Objects* Version 1.0 (Greg Reynolds, ed.). The standardization effort of GO-1 took a couple of years during which extensive work was made on the GeoAPI library. Release 2.0 of the GeoAPI library was made at the time of the final publication of the GO-1 specification in 2005. This brief historical synopsis explains why this specification adopts the version number 3.0 despite there being no prior OGC specification of the same name.

The GeoAPI library and its reference implementation provide the OGC dual benefits. The reference implementation demonstrates to the standards writers that it is possible to develop a single, coherent implementation of all the ISO/OGC specifications covered by the standardized API. The API provides the OGC community with a new point of interoperability between client code written to use the API and library code written to implement the API, with this layer of interoperability explicitly based on the interfaces defined by the core standards of the OGC.

iii. Submitting organizations

The following organizations submitted this Implementation Standard to the Open Geospatial Consortium:

a) Geomatys, Arles, France.

iv. Submission contact points

All questions regarding this submission should be directed to the editor:

Martin Desruisseaux
Geomatys
24, Rue Pierre-Renaudel
13200 Arles, France

martin.desruisseaux@geomatys.fr

v. Changes to the OGC® Abstract Specification

The OGC® Abstract Specification does not require changes to accommodate this OGC® standard.

vi. Foreword

The GeoAPI interface library is developed by the GeoAPI project (http://www.geoapi.org/). These interfaces have been developed over a number of years with contributors acting as individual volunteers, as government or institutional workers, or as employees in technology companies. The formal list of contributors is maintained in the project documentation at http://www.geoapi.org/geoapi/team-list.html but many others have contributed to the project through discussions at meetings of the Technical Committee of the OGC, on the project mailing lists and elsewhere, by working on implementations or client code of the GeoAPI interfaces, or by helping with other concerns of the project.

This standard complements existing OGC standards by defining a new, language specific layer of normalization. This standard does not replace the core standards developing the ISO/OGC abstract model but complements those documents for developers who use the Java language by documenting the mapping of types and methods from the abstract model into Java and explaining the use of the GeoAPI library. Because this standard differs in design and ambition from earlier OGC specifications which also included Java language interfaces, this document has been proposed as a new standardization effort in its own right.

The GeoAPI Javadoc completed by the annexes A (*Conformance*) and B (*Source Java Archives*) are normative, while the annexes C (*Types and methods*), D (*UML diagram for referencing operation types*), E *(Departures from ISO standards*) and F (*Comparison with legacy OGC specifications*) are informative.

The interfaces described in this standard follow directly, without introducing any new concepts, from the previously published standards of the Open Geospatial Consortium and the International Organization for Standardization. Nonetheless, *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 Inc. 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.

vii. Introduction

The GeoAPI Implementation Standard defines the normalized use of the GeoAPI library.

The GeoAPI library contains a series of interfaces and classes in the Java language defined in several packages which interpret into Java the data model and UML types of the ISO and OGC standards documents. The library includes extensive Javadoc code documentation which complement the injunctions of the ISO/OGC specifications by explaining particularities of the GeoAPI library: interpretations made of the specifications where there was room for choice, constraints due to the library's use of Java, or standard patterns of behavior expected by the library, notably in its handling of return types during exceptional situations.

This document explains the GeoAPI library and defines its use by library code implementing the API and by client code calling the API. Jointly with the library itself, this work aims to provide a carefully considered interpretation of the OGC specifications for the Java language, to provide a base structure to facilitate the creation of software libraries which implementing OGC standards, and to give application developers a well defined, full documented binding reducing the programming effort of using the OGC abstract model and facilitating the portability of application code between different implementations. The interfaces defined in this standard provide one way to structure the use the Java language to implement software which follows the design and intents of the OGC/ISO specifications. The creators of the GeoAPI interfaces consider this approach as an effective compromise between the OGC specifications, the requirements of the Java language, and the tradition of the core Java libraries.

This version of the standard does not yet propose a complete set of interfaces covering the entire abstract standard of the ISO/OGC but focuses on an initial group of interfaces only. This initial group of interfaces covers enough of the abstract model to permit the definition of geospatial coordinate systems and geodetic anchoring points and to enable the conversion of coordinate tuples between different reference systems. The work writing interfaces matching other OGC specifications has already begun in the 'pending' version of the GeoAPI library. It is expected that these other interfaces will be proposed for standardization in subsequent revisions of this specification but the interfaces must first have been implemented, ideally several times, and then tested extensively by use.

GeoAPI Implementation Standard

1. Scope

The GeoAPI Implementation Standard defines, through the GeoAPI library, a Java language application programming interface (API) including a set of types and methods which can be used for the manipulation of geographic information structured following the specifications adopted by the Technical Committee 211 of the International Organization for Standardization (ISO) and by the Open Geospatial Consortium (OGC). This standard standardizes the informatics contract between the client code which manipulates normalized data structures of geographic information based on the published API and the library code able both to instantiate and operate on these data structures according to the rules required by the published API and by the ISO and OGC standards.

The normative publication of the library occurs in a Java Archive (JAR) format binary. That binary is distributed along with a ZIP format bundle of the Javadoc comments as HTML files. An online version of the Javadoc comments, which may contain fixes for errata discovered after publication of this specification, is available at the URL http://www.geoapi.org/3.0/javadoc/index.html.

Version 3.0 of the library covers the base of the OGC Abstract Model for geographic information. GeoAPI 3.0 provides utilities, base types, metadata structures, and geo-referencing data elements which enable the creation of reference systems for spatial coordinates related to the Earth and of mathematical operators to convert coordinates from one coordinate reference system to another. This version of the standard covers the specifications ISO 19103, ISO 19115, ISO 19111, some elements from the closely related OGC™ specification OGC 01-009 and four elements from ISO 19107 necessary to the implementation of ISO 19111. Future versions of this specification are expected to expand this set of interfaces to cover the full model of the OGC Abstract Specification series, including notably Coverage and Feature data structures, with the 'pending' portion of the GeoAPI project already exploring these new areas.

2. Conformance

This specification places no conformance constraints on client code which uses this API backed by some implementation. The Java compiler will both ensure that the client code correctly calls the methods which are invoked and ensure type safety for the objects obtained from the method call. Nonetheless, programmers of client code which uses GeoAPI are urged

to follow the best practices for use of the API which are documented in the Javadoc comments of GeoAPI as well as elsewhere, including herein.

This specification makes certain requirements of libraries implementing this API and defines several conformance classes for implementations covering different packages of the API or providing different levels of complexity in their implementations. These requirements and conformance classes are presented in Annex A (normative).

GeoAPI does not currently have any formal test suite through which to establish conformance of GeoAPI implementations. The construction of such a test suite presents several complex challenges which may be tackled over time. However, GeoAPI does include a validation framework which can be used during unit testing as explained in Annex A.

3. Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this specification, OGC 09-083r4, except for any departures from the listed specifications which are explicitly mentioned in this text. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this specification, OGC 09-083r4, are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies.

- ISO 19103, *Geographic information — Conceptual schema language*, 2005.
- ISO 19115, *Geographic information — Metadata*, 2003.
- ISO 19115, *Geographic information — Metadata / Corrigendum 1*, 2006.
- ISO 19115-2, *Geographic information — Extensions for imagery and gridded data*, 2007.
- ISO 19111, *Geographic information — Spatial referencing by coordinates*, 2007.
- OGC 01-009, *OpenGIS® Implementation Specification: Coordinate Transformation Services*, revision 1.00, 2001 (partially)
- *The Java Language Specification, 3rd Edition*. James Gosling, Bill Joy, Guy Steele, Gilad Bracha, Sun Microsystems, 2005.
- JSR 363: Units of Measurement API, https://jcp.org/en/jsr/detail?id=363.

The normative reference towards the ISO metadata standard, *ISO 19115*, follows the lead of *ISO 19111* in excluding all references to `MD_CRS` and associated types. *ISO 19111* states:

> "Normative reference to ISO 19115 is restricted as follows: in this international standard, normative reference to ISO 19111 excludes the MD_CRS class and its components classes."
>
> *ISO 19111:2007, section 3 "Normative References"*

Despite this statement here, this is documented as a departure from the standard in annex E.

4. Terms and definitions

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

4.1 Application Programming Interface (API)

A formally defined set of types and methods which establish a contract between client code which uses the API and implementation code which provides the API.

4.2 Java

Trademark of Oracle used to refer to an object oriented, single inheritance programming language whose syntax derives from the C programming language and which is defined by the Java Language Specification.

5. Conventions

The conventions in this document follow the model of the ISO 19100 series specifications and standard practice in the fields of geographic information systems and software programming.

5.1 Symbols (and abbreviated terms)

API Application Program Interface
ISO International Organization for Standardization
OGC Open Geospatial Consortium

UML Unified Modeling Language
XML eXtended Markup Language
1D One Dimensional
2D Two Dimensional
3D Three Dimensional
nD Multi-Dimensional

6. A Geographic API in Java

The GeoAPI library formalizes the handling of the types defined in the specification documents for working with geographic information adopted by the International Organization for Standardization (ISO) and the Open Geospatial Consortium (OGC). Whereas the specifications define data types, methods and relationships using the general UML notation, the GeoAPI library implements those standards as Java language interfaces or simple classes. The GeoAPI types jointly form an application programming interface (API) which provides two groups of developers with a common point of exchange. Developers wishing to implement code which fulfills the requirements of the ISO and OGC specifications can adopt GeoAPI as a roadmap for their development. Developers wishing to write code which uses the data types defined by the standards can simply call the methods of the interfaces; they also gain a measure of independence from the particular implementation they are using since another implementation of the API can be swapped without breaking any calls made to the GeoAPI interfaces.

The structure of the GeoAPI library mirrors the packaging and separation of the different ISO and OGC specifications by grouping different types and functionality in separate Java language packages.

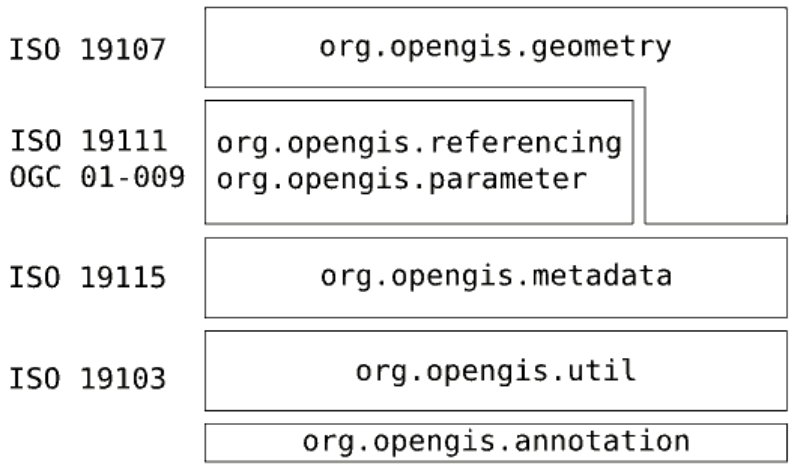


Figure 1: ISO specifications and GeoAPI packages mapping

The library rests on the `org.opengis.annotation` package which provides the annotation system used to document the origin and obligation level of all methods and types in the library. These annotations are available through introspection at runtime for any code which wishes to exploit this information. The base of the library is formed by a formal mapping of the core types used by the ISO and OGC standards to Java equivalents along with extra types

not defined in Java which are provided in the `org.opengis.util` package. The packages in the `org.opengis.metadata` namespace cover the data types defined in the ISO 19115 *Metadata* specification which are data structures holding textual references to elements describing other structures. The packages in the `org.opengis.parameter` and `org.opengis.referencing` namespaces implement the types from the ISO 19111 *Spatial Referencing by Coordinates* specification complemented by the mathematical operator types from the OGC 01-009 Implementation specification *Coordinate Transformation Services*. The packages in the `org.opengis.geometry` namespace cover the data types defined in the ISO 19107 *Spatial Schema* specification, although in version 3.0 of the library only defines the elements from that specification needed by the geo-referencing types defined in the OGC 01-009 specification since these packages are inter-dependent.

7. Annotation package

The GeoAPI annotation package uses the `org.opengis.annotation` namespace and implements Java language annotations and supporting classes which enable GeoAPI to document the origin, original name, and necessity of the various types and methods integrated from the various specification documents.

All classes in GeoAPI, including interfaces and enumeration types, which are based on a published standard should have an annotation label "`@UML`" documenting the standard in which are defined the type or method, the original name of the element and the obligation level of the type if other than the default mandatory level of obligation.

7.1 Use of the annotation types

As an example, the annotation label for the `ProjectedCRS` interface appears in the source code as:

```
@UML(identifier = "SC_ProjectedCRS",
     specification = ISO_19111)
```

which specifies that the type was defined in ISO 19111 standard, in the `SC` "*Coordinate Reference System*" package as the type "`GeographicCRS`" while the method `getCoordinateSystem()` of that class has the annotation:

```
@UML(identifier    = "coordinateSystem",
     obligation    = MANDATORY,
     specification = ISO_19111)
```

which indicates that the method was defined in the same ISO 19111 specification but had the name "`coordinateSystem`" in the standard rather than the "`getCoordinateSystem`" name used by GeoAPI and that a non-null value must be provided by every `ProjectedCRS` instance.

These annotations are available at runtime by Java introspection. This is useful, for example, when code needs to marshall data using the name defined by the ISO standard rather than the GeoAPI name. At runtime, the annotation of a reference to a GeoAPI interface can be obtained as follows, taking as an example the method `getTitle()` in the `Citation` type:

```
Class<?>      type   = Citation.class;
Method        method = type.getMethod("getTitle", (Class<?>[]) null);
UML           annot  = method.getAnnotation(UML.class);
String        ident  = annot.identifier();
Specification spec   = annot.specification();
Obligation    obl    = annot.obligation();
```

Java provides a class instance like the `Citation.class` instance used here for every type, either interface or class, defined in the runtime. The `getMethod(…)` call uses introspection to obtain a reference to the method from which the annotation can then be obtained. The annotation system therefore provides access, at runtime, to the original definition of the element.

8. Utility package

The GeoAPI utility package uses the `org.opengis.util` namespace and implements the types which are defined in the specification from the International Organization for Standardization ISO 19103:2005 *Geographic Information – Conceptual schema language* but are not already present in the Java language itself or in the standard Java library.

The utility package of GeoAPI completes the GeoAPI type mapping from the UML types used by the 19100 series of ISO standards into Java types by providing the elements missing from the Java language or standard library. The ISO 19103 specification defines types and utilities which are used as building blocks by the other standards in the 19100 series. GeoAPI maps these types either to existing types from the Java language and library or, when needed, to types defined in the utility package. For various practical reasons the mapping is not a one-to-one relationship. ISO 19103:2005 defines Primitive types (§8.1.1, of that standard), Collection or Dictionary types (§8.1.2), Enumerated types (§8.1.3), Representational types (§8.1.4), Name types (§8.1.5), and Derived types (§8.1.6). The mapping actually used is explained below. The utility package also includes the extra type `InternationalString` to handle textual sequences which might need to be represented in multiple languages and a basic factory.

The Java types mapped by GeoAPI or provided in the utility package can be used like regular Java language elements. Most of the types can be instantiated directly through public constructors. Enumeration types provide public access to each of their constants. `CodeList` types provide the static `valueOf(…)` method through which instances can be obtained. The `NameFactory` interface provides public methods for the instantiation of the various `GenericName` types. GeoAPI does not specify any extra constraints on the behavior or use of these types.

8.1 Package Mapping

GeoAPI maps the types of ISO 19103 into equivalents from the Java language and library or into types defined in the utility package. However, not all of the types in ISO 19103 have had a mapping defined because the need for these types has not yet appeared since they have not yet appeared in any other specification for which GeoAPI defines interfaces. Such types are listed as 'unimplemented' in the tables below.

8.1.1 Primitive Types

The Primitive types of the ISO/OGC specifications map to single object structures in GeoAPI. Where the mapping can be made directly to a Java primitive type, such as `int` and `double`, the Java primitive is preferred; however, when the value must be able to be set to `null`, the object wrapper of that primitive is used.

The following table shows the mapping used by GeoAPI to represent the types in the ISO 19100 series.

Table 1: Primitive Types Mapping

Type Group	ISO 19103 Type	GeoAPI Type
Numeric	`Integer`	`int` / `java.lang.Integer` `long` / `java.lang.Long`
	`UnlimitedInteger`	*unimplemented*
	`Real`	`double` / `java.lang.Double`
	`Decimal`	`java.math.BigDecimal`
	`Number`	`java.lang.Number`
	`Vector`	*unimplemented*
Text	`CharacterString`	`java.lang.String` `org.opengis.util.InternationalString`
	`Sequence<Character>`	`java.lang.CharSequence`
	`Character`	`char`
	`CharacterSetCode`	`org.opengis.metadata.identification.CharacterSet`
	`LanguageCharacterString`	*unimplemented*
Date and Time	`Date`	`java.util.Date`
	`Time`	`java.util.Date`

	DateTime	java.util.Date
	DatePrecision	*unimplemented*
Truth	Probability	*unimplemented*
	Boolean	boolean / java.lang.Boolean
	Logical	*unimplemented*
	Truth	*unimplemented*
	DiscreteTruth	*unimplemented*
	ContinuousTruth	*unimplemented*
Multiplicities	Multiplicity	*unimplemented*
	MultiplicityRange	*unimplemented*
Enumerations	Sign	*unimplemented*
	Digit	*unimplemented*
	Bit	*unimplemented*

Several of the objects in ISO 19103 have not been implemented since they have not yet been needed during the development of the rest of the interfaces. GeoAPI will consider implementing these types when they become necessary for the implementation of other elements in the ISO and OGC standards.

The interface `InternationalString` is an extension used by GeoAPI to handle Java `String` objects which may potentially need to be translated for users of different locales. Conceptually this acts as a String but may, depending on the implementation, provide access to locale specific representations of that String. This is useful, for example, when an implementation is operating on a server that serves multiple languages simultaneously, to allow sending String representations in the locale of the client rather than the locale of the server running the GeoAPI implementation.

> **Note:** `InternationalString` is inspired by JSR-150 (*Internationalization Service for J2EE*) with support for different timezones omitted.

8.1.2 Collection and dictionary types

GeoAPI implements ISO 19103 collection types using the standard Java Collections Framework. The one major difference is that GeoAPI collections do not implement the `TransfiniteSet` interface.

Table 2: Collection and Dictionary Types Mapping

ISO 19103 Type	GeoAPI Type
Transfinite Set	*unimplemented*
Collection	java.util.Collection
Set	java.util.Set
Bag	java.util.Collection
Sequence	java.util.List
CircularSequence	*unimplemented*

Dictionary	java.util.Map
KeyValuePair	java.util.Map.Entry

These collection types are used within GeoAPI qualified with a parametric type, which does not quite follow strictly the template notion which these types have in the ISO standards but is the closest one can conveniently do in the Java language.

8.1.3 Enumerated types

GeoAPI distinguishes between two enumerated types depending on whether the complete set of literal types is known when the code is originally created or if the list may be extended at run time or when the code is extended. The Java language provides the `Enum` language construct for the former case and GeoAPI defines the `CodeList` interface for the latter case.

Table 3: Enumerated Types Mapping

ISO 19103 Type	GeoAPI Type
Enumeration	java.lang.Enum
CodeList	org.opengis.util.CodeList

8.1.4 Representation types

GeoAPI currently defines only a strict minimum of the representation types in order to cover those necessary for the coverage package implementing the types in ISO 19123.

Table 4: Representation Types Mapping

ISO 19103 Type	GeoAPI Type
Schema	*Unimplemented*
Any	java.lang.Object
Type	org.opengis.util.Type
RecordSchema	org.opengis.util.RecordSchema
RecordType	org.opengis.util.RecordType
Record	org.opengis.util.Record

8.1.5 Name types

The name types in ISO 19103 have little documentation. The current explanation for how we interpret this Name system is in the Javadoc for `GenericName`:

http://www.geoapi.org/snapshot/javadoc/org/opengis/util/GenericName.html

which explains our current interpretation of scopes and namespaces.

Table 5: Name Types Mapping

ISO 19103 Type	GeoAPI Type
(constructors)	org.opengis.util.NameFactory
NameSpace	org.opengis.util.NameSpace
GenericName	org.opengis.util.GenericName
ScopedName	org.opengis.util.ScopedName
LocalName	org.opengis.util.LocalName
TypeName	org.opengis.util.TypeName
MemberName	org.opengis.util.MemberName

The `NameFactory` is an extension of the GeoAPI project designed to allow the construction of instances of these Name types.

8.1.6 Derived types

The derived types from ISO 19103 are almost all related to units and measurements. GeoAPI relies for these types on the interfaces defined by the external standard JSR-363.

The UOMo interfaces rely extensively on parametrized types to qualify the type of Unit or Measure being used.

Table 6: Derived Types Mapping

ISO 19103 Type	GeoAPI Type
Measure	javax.measure.Quantity
UnitOfMeasure	javax.measure.Unit<? extends Quantity>
Area	javax.measure.quantity.Area
UomArea	javax.measure.Unit<Area>
Length	javax.measure.quantity.Length
Distance	javax.measure.quantity.Length
UomLength	javax.measure.Unit<Length>
Angle	javax.measure.quantity.Angle
UomAngle	javax.measure.Unit<Angle>
Scale	javax.measure.quantity.Dimensionless
UomScale	javax.measure.Unit<Dimensionless>
Time	javax.measure.quantity.Time
UomTime	javax.measure.Unit<Time>
Volume	javax.measure.quantity.Volume
UomVolume	javax.measure.Unit<Volume>
Velocity	javax.measure.quantity.Speed
UomVelocity	javax.measure.Unit<Speed>

AngularVelocity	*Unimplemented*
UomAngularVelocity	*Unimplemented*
NULL	null
EMPTY	java.util.Collections.EMPTY_SET

GeoAPI uses the Java language keyword `null` to represent the ISO NULL value and the empty set from the Java Collections Framework for the ISO EMPTY. Note that programmers, for type safety when using Java Generics, should call the method `java.util.Collections.emptySet()` rather than refer directly to the constant, since the former will have the parametric type at compile time.

8.2 Use of the utility types

Use of the types in the GeoAPI utility package follows directly standard practice in Java.

The `org.opengis.util.InternationalString` interface provides a container for multiple versions of the same text, each for a specific `Locale` – the identifier used in Java for a specific language, possibly in a named territory.

```
NameFactory factory =  ...{Implementation dependent}
Map<Locale,String> names = new HashMap<Locale,String>();
names.put(Locale.ENGLISH, "My documents");
names.put(Locale.FRENCH,  "Mes documents");
InternationalString localized = factory.createInternationalString(names);
System.out.println(localized);
System.out.println(localized.toString(Locale.FRENCH));
```

The method to obtain factories is not specified by this standard and therefore depends on the design of the library implementation. Also, the locale used by default depends on the choice of the implementation so the result of the call `toString()` without parameters will depend on the implementation.

The use of `org.opengis.util.CodeList` constructs includes accessing statically defined elements, defining new elements and retrieving any element defined for the code list. Considering, for example, `org.opengis.metadata.distribution.MediumName` used to specify the kinds of physical media on which a data set could be distributed, the following code could be used

```
MediumName cd     = MediumName.CD_ROM;
MediumName usbkey = MediumName.valueOf("USB_KEY");
```

where the second locution will create a new value if it does not exist. Special care should be taken to keep such calls consistent throughout the code since the `CodeList` will create a new element if there are any differences between the `String` parameters: for example, the call

```
MediumName med = MediumName.valueOf("CDROM");
```

would return a new value rather than the static `CD_ROM`.

The use of `javax.measure.Unit` and associated types is explained at length in the specification document *Units and Measures*. Here, only a trivial example is presented (the `Units` class must be provided by a JSR-363 implementation):

```
Unit<Length> sourceUnit = Units.MILE;
Unit<Length> targetUnit = Units.KILOMETRE;
UnitConverter converter = source.getConverterTo(target);
double source = 123.2;
double target = converter.convert(source);
```

where the initial calls define units of length and then a converter is used to obtain the equivalent length in a new unit.

8.3 Departure from ISO 19103

GeoAPI differs from ISO 19103 in not providing all of the types defined in the standard. The elements that have not been defined have not yet been encountered in subsequent standards implemented by GeoAPI.

The `InternationalString` type provided by the utility package extends the basic `CharSequence` type provided by Java for internationalization by enabling the object to hold a separate `String` for every locale it wishes to handle.

The `NameFactory` type provided by the utility package complements the Name types defined by ISO 19103 by providing a formalized approach to instantiating the objects.

The Collections provided by GeoAPI are the standard Java collections and therefore do not extend `TransfiniteSet` as required by the ISO 19103 specification. However, the concept of `TransfiniteSet` applies most naturally to geometric constructs rather than to sets more generally.

8.4 Future improvements

There are several improvements related to the GeoAPI utility package that are to be expected in future revisions of this standard. The `GenericName` system may need another revision since it has proved to be a very difficult system to interpret correctly. Similarly, the `Record` system remains unclear and may need revision. The mapping of elements to `Date` might eventually evolve since the Java standard library is gaining its third implementation of data types designed to hold calendar based temporal references; if the new constructs replace the old with much more convenient functionality it might be worth moving to the new constructs in some future revision.

9. Metadata packages

The GeoAPI metadata packages use the `org.opengis.metadata` namespace and implement the types defined in the specification from the International Organization for Standardization ISO 19115:2003 *Geographic Information – Metadata* along with the modifications of *Technical Corrigendum 1* from 2006. They are completed or merged with the types defined in ISO 19115-2:2007 *Geographic Information – Extensions for imagery and gridded data*.

The metadata packages of GeoAPI provide container types for descriptive elements which may be related to data sets or components. All of these data structures are essentially containers for strings, and the interfaces consist almost exclusively of methods which provide access to the strings or a container. The API defines no methods which manipulate or modify the data structures.

The metadata packages of GeoAPI have been built primarily in support of the geodetic types defined in the referencing packages and therefore consider primarily read access to the data structure contents. The GeoAPI metadata interfaces provide no methods to set the values of the types. Furthermore, because the way that wild-cards for Java Generics have been used in the interfaces, the collection instances are constrained to be read only. Implementors are free to provide a fully mutable implementation of GeoAPI interfaces, but users may need to cast to the implementation classes in order to modify a metadata.

The GeoAPI rules of method return values have been changed for the metadata packages. Elsewhere in GeoAPI, methods which have a mandatory obligation in the specification must return an instance of the return type and cannot return the Java `null` reference. However, in the metadata package this rule is relaxed because data sets are encountered so frequently which have not correctly followed the requirements of the specification. In the GeoAPI metadata packages, all methods are considered to have an optional obligation and must follow the rules for that obligation level. This means that metadata methods shall return the object if present or otherwise either return `null` or return the empty collection, if the method return type is a Java Collection. This modification has been adopted to allow implementations sufficient latitude to handle metadata records which do not correctly conform to the specification. Nonetheless, sophisticated implementations can determine if a metadata record conforms with the specification by inspecting the annotation at runtime.

9.1 Package mapping

The mapping of ISO 19115 packages to GeoAPI packages follows an almost perfectly parallel naming scheme.

Table 7: Metadata Package Mapping

ISO 19115 Package	GeoAPI Package
Metadata entity set information	org.opengis.metadata
Identification information	org.opengis.metadata.identification
Constraint information	org.opengis.metadata.constraint
Data quality information	org.opengis.metadata.quality org.opengis.metadata.lineage
Maintenance information	org.opengis.metadata.maintenance
Spatial representation information	org.opengis.metadata.spatial
Reference system information	org.opengis.referencing.* org.opengis.parameter *(see below)*
Content information	org.opengis.metadata.content
Portrayal catalogue reference	org.opengis.metadata
Distribution information	org.opengis.metadata.distribution
Metadata extension information	org.opengis.metadata
Application schema information	org.opengis.metadata
Extent information	org.opengis.metadata.extent
Citation and responsible party information	org.opengis.metadata.citation

Several minor packages have been aggregated into the top level package. The *Data quality information* package has been split into two packages to separate the `DQ_*` types from the `LI_*` types. As explained next, the *Reference system information* has been replaced by the types from the referencing package.

9.2 Use of the GeoAPI metadata packages

The types in the GeoAPI metadata packages are primarily containers of Java String types, primitive types and other metadata types, and have been designed around providing read access to those elements. Metadata elements will be encountered in the data types from the referencing packages and the interfaces enable users to obtain the elements of the data type.

As an example, we want to print a list of all the authors for a document starting with an `org.opengis.metadata.citation.Citation` element.

```
Citation citation = ...; // We assume this instance is already available

for (ResponsibleParty rp : citation.getCitedResponsibleParties()) {
    if (rp.getRole() == Role.AUTHOR) {
        String author = rp.getIndividualName();
        System.out.println(author);
    }
}
```


The remainder of the metadata packages work in similar ways, where client code must disaggregate an instance to obtain the elements needed.

9.3 Departures from standard

The major departure in the GeoAPI metadata packages from the published ISO 191115 standard come from GeoAPI following the ISO 19111 standard and replacing the `MD_CRS` type from ISO 19115 with the types in ISO 19111. The types from ISO 19111 duplicate the classes present in the metadata specification but with richer, more complete semantics. GeoAPI does not implement the following classes but substitutes a suitable replacement from the referencing packages.

Table 8: Mapping of types from the reference system information package

ISO 19115 type	GeoAPI replacement
MD_ReferenceSystem	org.opengis.referencing.ReferenceSystem
MD_CRS	org.opengis.referencing.crs.CoordinateReferenceSystem
MD_EllipsoidParameters	org.opengis.referencing.datum.Ellipsoid
MD_ProjectionParameters	org.opengis.parameter.ParameterValueGroup
MD_ObliqueLineAzimuth	org.opengis.parameter.ParameterValue
MD_ObliqueLinePoint	org.opengis.parameter.ParameterValue

Note however, that the parameter package of GeoAPI and ISO 19111 is more generic than the explicit types defined in ISO 19115, handling referencing constructs in a map like structure rather than as individual, named data types.

Another departure is in the way GeoAPI metadata package added the types and methods defined in the specification ISO 19115-2 *Geographic Information – Metadata – Part 2: Extensions for imagery and gridded data*. The latter was forced to create a number of types to hold elements which naturally could occur directly in the types defined by ISO 19115. We integrated such types directly into the existing types rather than adding complexity to the API which exists by historical accident.

9.4 Future work

Future revisions of these packages may add factory interfaces through which these types could be instantiated. However, the actual design for such a factory system has not yet been agreed upon by the contributors to GeoAPI.

10. Geometry packages

The GeoAPI geometry packages use the `org.opengis.geometry` namespace and implement the types defined in the specification from the International Organization for Standardization ISO 19107:2003 *Geographic Information - Spatial schema*.

The geometry packages of GeoAPI provide spatial types combining coordinates with the reference system used for those coordinates. These types implement a vector based spatial representation of elements. The geometry packages also include a sophisticated container-ship hierarchy, objects which know of their boundary, and topological data structures.

The geometry types defined in this standard include only the two simplest types in the specification along with their abstract parent interface. It is expected that the two concrete types will be instantiated through public constructors.

10.1 Defined types

GeoAPI defines a minimal set of four types from the ISO 19107 *Geographic Information - Spatial schema* specification, `DirectPosition`, `Position`, `Envelope`, and `MismatchedDimensionException`, because these types are needed by the referencing package.

Table 9: Mapping of types from the Coordinate geometry package

ISO 19107 type	GeoAPI type
GM_Position	org.opengis.geometry.coordinate.Position
DirectPosition	org.opengis.geometry.DirectPostion
GM_Envelope	org.opengis.geometry.Envelope

The `DirectPosition` type represents a single location in the anchored coordinate space defined by a `CoordinateReferenceSystem`. Since `DirectPosition` extends the `Position` type that interface was needed as well.

The `Envelope` type represents the lower and upper extreme values along each axis. The type is frequently conflated with a bounding rectilinear box but the two elements differ conceptually in subtle ways. For example, the bounding box of Siberia crosses the anti-meridian and runs from around 60 degrees east of Greenwich to 170 degrees west whereas the Envelope for Siberia goes from -180 degrees longitude to 180 degrees longitude. A further possible confusion arises because the Envelope type in ISO 19107 provides methods to obtain the 'corners' of the Envelope as `DirectPositions`. However, users should note that these `DirectPositions` might not have any meaning in physical space. For example the

corners could be outside the CRS domain of validity even if the feature itself is fully inside that domain. The corner `DirectPositions` are acting, for convenience, as data containers for a tuple of ordinates but not as representations of an actual Position so the ordinates of the tuple must be considered independent.

GeoAPI also defines a `MismatchedDimensionException` Java exception. This type can be used for method calls whose parameters might be nonsensical if they do not share the same, or have the correct, dimension.

10.2 Use of the geometry packages

The usage of the data types in the geometry package of GeoAPI follow the standard rules of Java and do not warrant extended explanation here.

10.3 Departure from Standards

GeoAPI has moved the `DirectPosition` and `Envelope` types from the `coordinate` sub-package where they are defined in the ISO 19107 specification up to the `org.opengis.geometry` package due to their importance and frequency of use. Conceptually, the ISO 19107 standard considers geometric objects to be collections of `DirectPositions` so that data structure is used throughout the API.

10.4 Future work

Future versions of this specification are expected to present a much larger set of interfaces for the types from ISO 19107. For now, the interfaces defined by the GeoAPI project remain experimental with no functional reference implementation.

11. Referencing and Parameter packages

The GeoAPI referencing and parameter packages use the `org.opengis.referencing` and `org.opengis.parameter` namespaces respectively and implement the types defined in the standard from the International Organization for Standardization ISO 19111:2007 *Geographic Information - Spatial referencing by coordinates*. The referencing package also includes the types describing object factories and mathematical transformation operators between reference frames defined in the standard from the Open Geospatial Consortium OGC 01-009 *OpenGIS Implementation Specification: Coordinate Transformation Services* from 2003.

The referencing and parameter packages of GeoAPI provide data constructs and operations for geospatial referencing and coordinate operations.

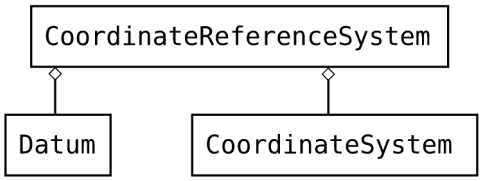


Figure 2: Components of a CRS
(after Fig.2, ISO 19111:2007)

The referencing package types can be used to define geospatial referencing constructs based on the ISO 19111 specification which can be used to define various engineering and geodetic datums, define various coordinate systems, and combine those to define all the coordinate referencing systems (CRS) generally encountered in geospatial science.

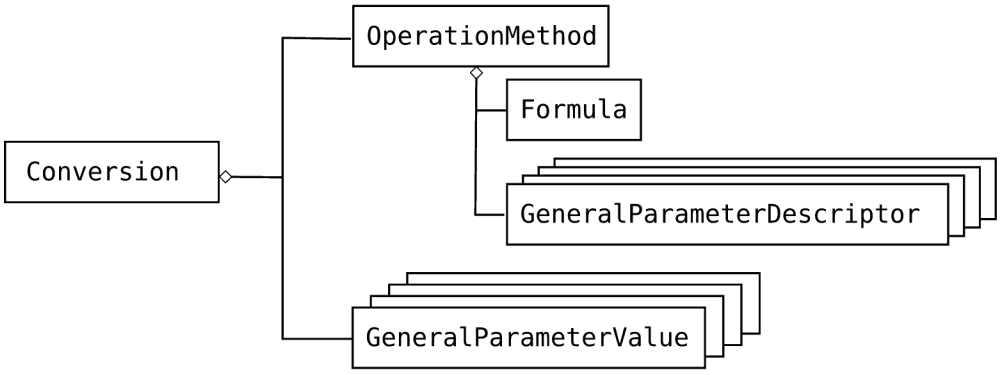


Figure 3: Components of a Mercator projection

Finally, the referencing packages include factory types also defined originally in the OGC 01-009 specification. These object factories define a normalized approach to object instantiation and come in two forms, the `ObjectFactories` which instantiate objects by assembling types passed as arguments and the `AuthorityFactories` which instantiate objects based on the values of some third party database, notably those in the EPSG SQL database of referencing objects assembled by the Surveying & Positioning Committee of the International Association of Oil & Gas producers.

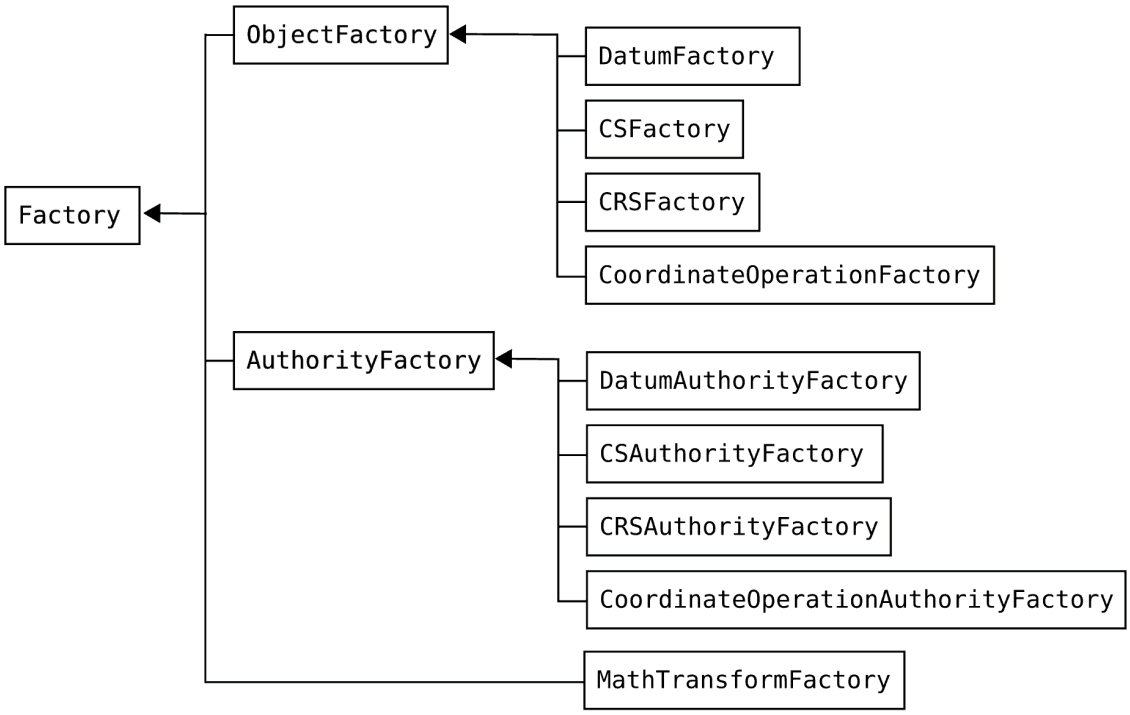


Figure 4: Referencing factories

The use of the types defined in the GeoAPI referencing and parameter packages follows the general usage pattern of the library. Since these packages provide factories, so code that needs to instantiate one of the objects defined in these packages should first obtain a reference to the factory in some implementation dependent manner and then use the factory methods to instantiate the desired object instances. These instances can then be used through the interface defined in the GeoAPI library. The only unusual pattern in these packages arises because the `ParameterValue` types provide methods to set the value of the type. In the general use pattern for these types, a `ParameterValueGroup` containing all the named parameters for a method of an operation is first obtained from the `MathTransformFactory` and then each `ParameterValue` type is obtained in turn and its value set. This use pattern ensures that all the needed parameters for an operation method can be obtained as a single block.

11.1 Package Mapping

The mapping of ISO 19111 packages to GeoAPI packages follows an almost perfectly parallel naming scheme while the OGC 01-009 packages map to GeoAPI less linearly because the factory system of the OGC standard provides factory types in each GeoAPI package.

Table 10: Referencing and Parameter Package Mapping

ISO 19111 (OGC 01-009) Package	GeoAPI Package
IO Identified Object	org.opengis.referencing
RS Reference System	org.opengis.referencing
SC Coordinate Reference System	org.opengis.referencing.crs
CS Coordinate System	org.opengis.referencing.cs
CD Datum	org.opengis.referencing.datum
CC Coordinate Operation	org.opengis.referencing.operation org.opengis.parameter
CS Coordinate Systems (OGC 01-009)	org.opengis.referencing org.opengis.referencing.crs org.opengis.referencing.datum
CT Coordinate Transformations (OGC 01-009)	org.opengis.referencing.operation
PT Positioning (OGC 01-009)	org.opengis.referencing.operation

Nonetheless, the mapping is fairly straightforward. It should be noted, as was discussed in the section on Metadata, that several types from the ISO 19115 specification also map into the GeoAPI referencing packages.

11.2 Use of the referencing and parameter types

The following examples illustrate the use of the referencing and parameter packages of GeoAPI.

11.2.1 Creating a Projected Coordinate Reference System

A Coordinate Reference System can be constructed on its own or can be derived from other systems. This example shows how to build a `ProjectedCRS` based on the Mercator projection. Here we use an Authority which has already defined the method for this projection and then set the parameters to desired values before creating the CRS.

```
// Obtaining factory instances is implementation dependent
CRSFactory crsFactory = ...;
CoordinateOperationFactory opFactory = ...;
CoordinateOperationAuthorityFactory af = ...;
```



```
// We assume these instances are already available (used at end)
GeographicCRS baseGeograhpicCRS = ...;
CartesianCS cartesianCS = ...;

// Get the parameters initialized to their default values
OperationMethod method = af.createOperationMethod("Mercator (1SP)");
ParameterValueGroup pg = method.getParameters().createValue();

// Set the parameter values
pg.parameter("semi-major axis").setValue(6377397.155);
pg.parameter("semi-minor axis").setValue(6377397.155 * (1 - 1/299.15281));
pg.parameter("Latitude of natural origin").setValue(0.0);
pg.parameter("Longitude of natural origin").setValue(110.0);
pg.parameter("Scale factor at natural origin").setValue(0.997);
pg.parameter("False easting").setValue(3900000.0);
pg.parameter("False northing").setValue(900000.0);

// Create the defining conversion
Map<String,Object> properties = new HashMap<String,Object>();
properties.put(Conversion.NAME_KEY, "Makassar / NEIEZ");
Conversion def = opFactory.createDefiningConversion(properties, method, pg);

// Create the projected CRS
properties.clear();
properties.put(Conversion.NAME_KEY, "Makassar / NEIEZ");
ProjectedCRS projectedCRS = crsFactory.createProjectedCRS(
        properties, baseGeographicCRS, def, cartesianCS);
```

This gives us a `ProjectedCRS` with the appropriate parameters for our needs.

11.2.2 Build a Coordinate Operation

In this usage example we build an operation using a sophisticated factory.

```
// Obtaining factory instances is implementation dependent
CoordinateOperationFactory opFactory = ...;

// We assume these instances are already available (taken from above)
CoordinateReferenceSystem sourceCRS = baseGeographicCRS;
CoordinateReferenceSystem targetCRS = projectedCRS;

CoordinateOperation op = opFactory.createOperation(sourceCRS, targetCRS);
```

The factory has done all the work of establishing which parameters should be used and correctly instantiating the operation.

11.2.3 Transform a coordinate between coordinate reference systems.

In this example, we use the operation we just created to calculate the coordinates in a destination coordinate reference system equivalent to the coordinates in a source coordinate reference system.

```
// We assume these instances are already available
CoordinateOperation op = ...;
double[] sourceOrdinates = ...;
```

```
// Create the destination array
double[] targetOrdinates = new double[sourceOrdinates.length];

MathTransform mt = op.getMathTransform();
mt.transform(sourceOrdinates, 0, targetOrdinates, 0,
        sourceOrdinates.length / mt.getSourceDimensions());
```

with the user needing to guarantee that the length of the ordinate arrays are the same integer multiple of the number of dimensions in their respective coordinate reference systems.

11.3 Departure from Standards

The major departure of GeoAPI from the ISO 19111 standard comes from the inclusion, directly in the `CoordinateOperation` type, of a method providing access to the `MathTransform` construct from the older OGC specification. This departure fundamentally alters the function of these packages: under the ISO 19111 standard the classes only describe coordinate reference systems and the operations which convert between them, under GeoAPI the classes also provide an object which can actually calculate the coordinates in a destination CRS equivalent to given coordinates in a source CRS. For reasons of consistency with the OGC 01-009 approach, the method providing access to the `MathTransform` has been directly integrated into the `CoordinateOperation` interface so that users can obtain the mathematical object directly from the object that defines the operation. GeoAPI further departs in defining its own 1D and 2D `MathTransforms`, for speed, convenience and interoperability with the Java2D graphics library.

The second major departure of GeoAPI from the ISO 19111 standard comes from the addition of the factory system defined in the OGC 01-009 standard. This departure adds two factory hierarchies, a default factory hierarchy in which new instances are obtained by providing the content as parameters to the method calls and an 'authority' factory hierarchy in which instances are obtained based on some code identifier of the object desired specific to the particular authority supported by the factory instance. The factories provide a common basis for object instantiation and, if used exclusively, simplify the work of switching between implementations. The interfaces describe two type hierarchies for factory types: the hierarchy rooted in the `ObjectFactory` type all instantiate objects by given the necessary content elements whereas the factories rooted in `AuthorityFactory` instantiate objects based on some identification code and some data source mapping the code to object contents. GeoAPI focuses especially on the few authority codes provided by the OGC in the CRS and AUTO namespaces and the authority codes provided by the EPSG database of the Surveying & Positioning Committee of the International Association of Oil & Gas producers (OGP).

One minor departure from the ISO 19111 specification comes from GeoAPI defining an Ellipsoidal `VerticalDatumType`. The ISO specification does not allow distances above an ellipsoid independent of the longitude and latitude coordinates in order to prevent users from misusing the vertical ordinate during conversion. However, this separation is not inherently incorrect, but merely dangerous, and is necessary to handle older constructs such as the

coordinate reference systems defined in the Well-Known Text textual format. GeoAPI has therefore elected to integrate this vertical datum type.

11.4 Future work

The referencing and parameter packages are not expected to change fundamentally in subsequent revisions of this standard. The only changes which might arise would come from unforeseen conflicts during the integration of the temporal types from the ISO 19108 *Geographic Information - Temporal Schema* standard which defines its own `TemporalCRS` and `TemporalCS` which are expected to be dropped in favor of the types already defined in the referencing packages.

Annex A

(normative)

Conformance

Libraries implementing GeoAPI are enjoined to follow certain requirements to claim conformance with this standard. The standard does permit implementations with different levels of coverage of the library by providing, below, a number of conformance classes for implementation libraries.

A.1 Fundamental requirements

All implementing libraries must follow the requirements made in this clause.

Implementing libraries must satisfy all paragraphs in this standard and in the library Javadoc that use the keywords "required", "shall", "shall not", or "must".

Java libraries which provide code implementations of the GeoAPI interfaces and which wish to claim conformance with this standard shall follow the dictates both of the Javadoc comments in the API and of the language of the OGC specifications which define each Java method.

Conformant libraries shall respect the following general pattern for method return values unless countermanded by the Javadoc code documentation for a particular method. Methods which generate new instances, such as `Factory` methods, are expected to return the desired value or to throw a checked exception such as a `FactoryException`. 'Setter' methods, methods which set the value of an object, are expected either to succeed or to throw an `UnsupportedOperationException` if the method is either not implemented or illegal in that implementation. 'Getter' methods, methods which obtain a value from an object, are documented through annotations to the Javadoc as mandatory or optional. Mandatory 'getter' methods are expected to return the requested value unless the value is missing in which case they shall throw the runtime exception, `IllegalStateException`. (An exception is made to this rule in the metadata packages because of the extensive existence of incomplete metadata. In those packages, all methods are treated as optional.) Optional 'getter' methods are expected to return the requested value unless the value is missing or the method is not implemented in which case they shall return `null`. Exceptions to these general rules occur occasionally but are documented in the Javadoc comments.

All the instances of GeoAPI interfaces which are generated by a conformant library shall be valid according to the test validator, whenever a validator exists for the instance type. This does not require that all instances be tested but merely that if the instances were tested, they would validate.

A.2 Conformance levels

This standard provides several levels of conformance for libraries that wish to claim conformance with this standard.

All implementations must necessarily provide a fully functional implementation of the base types required by the library. This means that all implementing libraries must provide a fully working implementation of the JSR-363 standard, possibly by including the reference implementation directly. All implementing libraries must also provide functional implementations of the types defined in the `org.opengis.util` package.

A.2.1 Conformance Level M – Metadata

The first level of conformance, **M1**, requires the implementing library to provide a functional implementation of methods annotated with `@Profile(level=CORE)` in the `org.opengis.metadata` packages.

The second level of conformance, **M2**, requires the implementing library to provide a functional implementation of all the types defined in the `org.opengis.metadata` packages.

A.2.2 Conformance Level R-A – Referencing Base

Libraries implementing the types defined in the `org.opengis.referencing` and `org.opengis.parameter` packages can reach several different levels of conformance depending on the coverage and complexity of their implementation.

The simplest conformant status for the Referencing level, Status **R-A1** provides code, including the `ObjectFactory` types, which can instantiate all the objects in the `org.opengis.referencing.datum`, `cs`, and `crs` packages but may be limited to the creation of coordinate referencing systems which are not compound.

The next status for this level, Status **R-A2** provides the types in level **R-A1** but includes all the types necessary for compound coordinate reference systems. At this conformance level, the implementation must be able to construct any `CoordinateReferenceSystem` which is legal under the ISO 19111 standard, including all of the projected systems.

A.2.3 Conformance Level R-B – Referencing Authority Factories

This conformance level requires implementations to be able to instantiate types from the Authority factories.

The simplest conformance status for this level, Status **R-B1** requires being able to instantiate the most common objects from the OGC authority. The factory must be able to handle the following identifiers:

- CRS:1 (computer display)
- CRS:84 (geographic, WGS 84)
- CRS:83 (geographic, NAD83)
- CRS:27 (geographic, NAD27)
- CRS:88 (NAD vertical datum)
- AUTO2:42001 (Universal Transverse Mercator)
- AUTO2:42002 (Transverse Mercator)
- AUTO2:42003 (Orthographic)
- AUTO2:42004 (Equirectangular)
- AUTO2:42005 (Mollweide)

which are defined by the OGC for other implementation specifications. The factory should also be able to handle the URN form of these identifiers, such as urn:ogc:def:crs:epsg:4326, and the URL form, such as http://www.opengis.net/gml/srs/epsg.xml#4326.

The next conformance status for this level, Status **R-B2** requires being able to instantiate valid instances from any Well-Known Text (WKT) string. WKT is defined in OGC 01-009.

The final conformance status for this level, Status **R-B3** requires being able to instantiate a valid instances of the `Datum`, `CoordinateSystem`, or `CoordinateReferenceSystem` interfaces based on the codes and values in the EPSG database. The database is maintained by the Surveying and Positioning Committee of the International Association of Oil and Gas Producers and can be found at the URL http://www.epsg.org/.

A.2.4 Conformance Level R-C – Referencing Operations

This conformance level requires implementations to be able to create the types in the `org.opengis.referencing.operation` and `org.opengis.parameter` packages.

The simplest conformance status for this level, Status **R-C1** requires implementations to provide the `CoordinateOperationFactory` type and be able to instantiate any of the types in the two packages.

The second conformance status for this level, Status **R-C2** requires a `CoordinateOperationAuthorityFactory` able to instantiate the `CoordinateOperation` instances based on the codes and values in the EPSG database.

A.2.5 Conformance Level R-M – Math Transforms

This conformance level requires that the `CoordinateOperations` provided by the implementations be able to create the appropriate `MathTransform` instance for the `OperationMethod` of the `CoordinateOperation`. The `MathTransform` will then permit the calculation of coordinates in a target coordinate reference system from the values of a coordinate in a source coordinate reference system. The different status categories for this level are distinguished by the mathematical complexity of the `OperationMethod` which are supported.

The first conformance status for this level, Status **R-M1** requires that conformant implementations be able to instantiate the appropriate `MathTransform` instance for any `CoordinateOperation` which uses one of the `OperationMethod` types identified below:

- Affine general parametric transformation (EPSG:9624)
- Longitude rotation (EPSG:9601)
- Equidistant Cylindrical (EPSG:9842, 9823)
- Mercator (1SP) (EPSG:9804)
- Mercator (2SP) (EPSG:9805)

These `MathTransform` instances involve no shift in Datum and the most basic mathematical treatment.

The next conformance status for this level, Status **R-M2**, requires that conformant implementations be able to instantiate the appropriate `MathTransform` instance for any `CoordinateOperation` which uses one of the `OperationMethod` types identified below:

- Transverse Mercator (EPSG:9807)
- Transverse mercator (South Orientated) (EPSG:9808)
- Lambert Conic Conformal (1SP) (EPSG:9801)
- Lambert Conic Conformal (2SP) (EPSG:9802)
- Lambert Conic Conformal (2SP Belgium) (EPSG:9803)

These operations involve no shift in Datum but require more advanced mathematics.

The third conformance status for this level, Status **R-M3**, requires that conformant implementations be able to instantiate the appropriate `MathTransform` instance for any `CoordinateOperation` which uses one of the `OperationMethod` types identified below:

- Molodensky transformation (EPSG:9604)
- Abridged Molodensky transformation (EPSG:9605)
- Geographic/geocentric conversions (EPSG:9602)
- Geocentric translation (EPSG:9603)
- Position Vector 7-parameters (EPSG:9606)
- Coordinate Frame rotation (EPSG:9607)

These operations perform a shift in Datum but the shifts require only a small number of parameters.

The final conformance status for this level, Status **R-M4** requires that conformant implementations be able to instantiate the appropriate `MathTransform` instance for any `CoordinateOperation` which uses one of the `OperationMethod` types identified below:

- Ellipsoid to Geoid
- North American Datum Conversion (EPSG:9613)

These operations require a shift in Datum based on an extensive set of parameters using a numerical Grid or a set of spherical harmonic parameters.

A.3 Validation

The GeoAPI source bundle, in the test packages of the conformance modules, contains a number of validator which can be used in JUnit test cases to test compliance of the objects created in an implementation. This is not as sophisticated as a full conformance test suite. Nonetheless, the GeoAPI validators can establish that certain instances are invalid and therefore can readily be integrated into the test suite of any implementation library.

A.3.1 Example of a validation test

The following code demonstrates an example which uses the validators contained in the GeoAPI binary distribution to evaluate an instance object created by the implementation within a unit test. This test would require the JUnit library, version 4 or later, on the Java Classpath.

```
import static org.opengis.test.Validators.*;

public class ValidationTests {

    @Test
    public void testCRS() {
        // The implementation would build this CRS
        CoordinateReferenceSystem crs = ...;
        validate(crs);
    }
}
```

If the validation fails, the JUnit library would throw an `AssertionError`. Also, the GeoAPI binary JAR archive must be on the Java CLASSPATH for the library to be linkable at runtime.

Annex B

(normative)

GeoAPI Source Java Archive

In addition to this document, this specification includes the normative GeoAPI Java archive file:

```
geoapi-3.0.1-sources.jar
```

That archive contains the authoritative Javadoc code documentation for the types and methods.

The Java archive file contains the following elements:

```
META-INF/MANIFEST.MF
org.opengis.annotation/
org.opengis.geometry/
org.opengis.geometry.coordinate/
org.opengis.metadata/
org.opengis.metadata.citation/
org.opengis.metadata.constraint/
org.opengis.metadata.content/
org.opengis.metadata.distribution/
org.opengis.metadata.extent/
org.opengis.metadata.identification/
org.opengis.metadata.lineage/
org.opengis.metadata.maintenance/
org.opengis.metadata.quality/
org.opengis.metadata.spatial/
org.opengis.parameter/
org.opengis.referencing/
org.opengis.referencing.crs/
org.opengis.referencing.cs/
org.opengis.referencing.datum/
org.opengis.referencing.operation/
org.opengis.util/
```

with each directory holding Java source files (.java extension) and some directories having documentation directories holding text or image files.

Annex C

(informative)

GeoAPI Types and Methods

This annex lists the GeoAPI identifiers (first column) together with the OGC/ISO identifiers and their originating specifications. This list includes every types and members present in the Javadoc, but without their method signature. Implementors should refer to the Javadoc for the detailed API description.

Package org.opengis.geometry		
Interface DirectPosition	DirectPosition	ISO 19107
getCoordinateReferenceSystem	coordinateReferenceSystem	ISO 19107
getDimension	dimension	ISO 19107
getCoordinate	coordinate	ISO 19107
getOrdinate		
setOrdinate		
equals		Java
hashCode		Java
Interface Envelope	GM_Envelope	ISO 19107
getCoordinateReferenceSystem		
getDimension		
getLowerCorner	lowerCorner	ISO 19107
getUpperCorner	upperCorner	ISO 19107
getMinimum		
getMaximum		
getMedian		
getSpan		
Class MismatchedDimensionException		

Package org.opengis.geometry.coordinate		
Interface Position	GM_Position	ISO 19107
getDirectPosition	direct	ISO 19107

Package org.opengis.metadata

Interface ApplicationSchemaInformation	MD_ApplicationSchemaInformation	ISO 19115
getName	name	ISO 19115
getSchemaLanguage	schemaLanguage	ISO 19115
getConstraintLanguage	constraintLanguage	ISO 19115
getSchemaAscii	schemaAscii	ISO 19115
getGraphicsFile	graphicsFile	ISO 19115
getSoftwareDevelopmentFile	softwareDevelopmentFile	ISO 19115
getSoftwareDevelopmentFileFormat	softwareDevelopmentFileFormat	ISO 19115
Code list Datatype	MD_DatatypeCode	ISO 19115
CLASS	class	ISO 19115
CODE_LIST	codelist	ISO 19115
ENUMERATION	enumeration	ISO 19115
CODE_LIST_ELEMENT	codelistElement	ISO 19115
ABSTRACT_CLASS	abstractClass	ISO 19115
AGGREGATE_CLASS	aggregateClass	ISO 19115
SPECIFIED_CLASS	specifiedClass	ISO 19115
DATATYPE_CLASS	datatypeClass	ISO 19115
INTERFACE_CLASS	interfaceClass	ISO 19115
UNION_CLASS	unionClass	ISO 19115
META_CLASS	metaClass	ISO 19115
TYPE_CLASS	typeClass	ISO 19115
CHARACTER_STRING	characterString	ISO 19115
INTEGER	integer	ISO 19115
ASSOCIATION	association	ISO 19115
Interface ExtendedElementInformation	MD_ExtendedElementInformation	ISO 19115
getName	name	ISO 19115
getShortName	shortName	ISO 19115
getDomainCode	domainCode	ISO 19115
getDefinition	definition	ISO 19115
getObligation	obligation	ISO 19115
getCondition	condition	ISO 19115
getDataType	dataType	ISO 19115
getMaximumOccurrence	maximumOccurrence	ISO 19115
getDomainValue	domainValue	ISO 19115
getParentEntity	parentEntity	ISO 19115
getRule	rule	ISO 19115
getRationales	rationale	ISO 19115
getSources	source	ISO 19115

Interface FeatureTypeList	MD_FeatureTypeList	ISO 19115
getSpatialObject	spatialObject	ISO 19115
getSpatialSchemaName	spatialSchemaName	ISO 19115
Interface Identifier	MD_Identifier	ISO 19115
getCode	code	ISO 19115
getAuthority	authority	ISO 19115
Interface Metadata	MD_Metadata	ISO 19115
getFileIdentifier	fileIdentifier	ISO 19115
getLanguage	language	ISO 19115
getCharacterSet	characterSet	ISO 19115
getParentIdentifier	parentIdentifier	ISO 19115
getHierarchyLevels	hierarchyLevel	ISO 19115
getHierarchyLevelNames	hierarchyLevelName	ISO 19115
getContacts	contact	ISO 19115
getDateStamp	dateStamp	ISO 19115
getMetadataStandardName	metadataStandardName	ISO 19115
getMetadataStandardVersion	metadataStandardVersion	ISO 19115
getDataSetUri	dataSetURI	ISO 19115
getLocales	locale	ISO 19115
getSpatialRepresentationInfo	spatialRepresentationInfo	ISO 19115
getReferenceSystemInfo	referenceSystemInfo	ISO 19115
getMetadataExtensionInfo	metadataExtensionInfo	ISO 19115
getIdentificationInfo	identificationInfo	ISO 19115
getContentInfo	contentInfo	ISO 19115
getDistributionInfo	distributionInfo	ISO 19115
getDataQualityInfo	dataQualityInfo	ISO 19115
getPortrayalCatalogueInfo	portrayalCatalogueInfo	ISO 19115
getMetadataConstraints	metadataConstraints	ISO 19115
getApplicationSchemaInfo	applicationSchemaInfo	ISO 19115
getMetadataMaintenance	metadataMaintenance	ISO 19115
getAcquisitionInformation	acquisitionInformation	ISO 19115-2
Interface MetadataExtensionInformation	MD_MetadataExtensionInformation	ISO 19115
getExtensionOnLineResource	extensionOnLineResource	ISO 19115
getExtendedElementInformation	extendedElementInformation	ISO 19115
Code list Obligation	MD_ObligationCode	ISO 19115
MANDATORY	mandatory	ISO 19115
OPTIONAL	optional	ISO 19115
CONDITIONAL	conditional	ISO 19115

Interface PortrayalCatalogueReference	MD_PortrayalCatalogueReference	ISO 19115
getPortrayalCatalogueCitations	portrayalCatalogueCitation	ISO 19115

Package org.opengis.metadata.acquisition

Interface AcquisitionInformation	MI_AcquisitionInformation	ISO 19115-2
getAcquisitionPlans	acquisitionPlan	ISO 19115-2
getAcquisitionRequirements	acquisitionRequirement	ISO 19115-2
getEnvironmentalConditions	environmentalConditions	ISO 19115-2
getInstruments	instrument	ISO 19115-2
getObjectives	objective	ISO 19115-2
getOperations	operation	ISO 19115-2
getPlatforms	platform	ISO 19115-2
Code list Context	MI_ContextCode	ISO 19115-2
ACQUISITION	acquisition	ISO 19115-2
PASS	pass	ISO 19115-2
WAY_POINT	wayPoint	ISO 19115-2
Interface EnvironmentalRecord	MI_EnvironmentalRecord	ISO 19115-2
getAverageAirTemperature	averageAirTemperature	ISO 19115-2
getMaxRelativeHumidity	maxRelativeHumidity	ISO 19115-2
getMaxAltitude	maxAltitude	ISO 19115-2
getMeteorologicalConditions	meteorologicalConditions	ISO 19115-2
Interface Event	MI_Event	ISO 19115-2
getIdentifier	identifier	ISO 19115-2
getTrigger	trigger	ISO 19115-2
getContext	context	ISO 19115-2
getSequence	sequence	ISO 19115-2
getTime	time	ISO 19115-2
getExpectedObjectives	expectedObjective	ISO 19115-2
getRelatedPass	relatedPass	ISO 19115-2
getRelatedSensors	relatedSensor	ISO 19115-2
Code list GeometryType	MI_GeometryTypeCode	ISO 19115-2
POINT	point	ISO 19115-2
LINEAR	linear	ISO 19115-2
AREAL	areal	ISO 19115-2
STRIP	strip	ISO 19115-2

Interface Instrument	MI_Instrument	ISO 19115-2
getCitations	citation	ISO 19115-2
getIdentifier	identifier	ISO 19115-2
getType	type	ISO 19115-2
getDescription	description	ISO 19115-2
getMountedOn	mountedOn	ISO 19115-2
Interface Objective	MI_Objective	ISO 19115-2
getIdentifiers	identifier	ISO 19115-2
getPriority	priority	ISO 19115-2
getTypes	type	ISO 19115-2
getFunctions	function	ISO 19115-2
getExtents	extent	ISO 19115-2
getObjectiveOccurences	objectiveOccurence	ISO 19115-2
getPass	pass	ISO 19115-2
getSensingInstruments	sensingInstrument	ISO 19115-2
Code list ObjectiveType	MI_ObjectiveTypeCode	ISO 19115-2
INSTANTANEOUS_COLLECTION	instantaneousCollection	ISO 19115-2
PERSISTENT_VIEW	persistentView	ISO 19115-2
SURVEY	survey	ISO 19115-2
Interface Operation	MI_Operation	ISO 19115-2
getDescription	description	ISO 19115-2
getCitation	citation	ISO 19115-2
getIdentifier	identifier	ISO 19115-2
getStatus	status	ISO 19115-2
getType	type	ISO 19115-2
getChildOperations	childOperation	ISO 19115-2
getObjectives	objective	ISO 19115-2
getParentOperation	parentOperation	ISO 19115-2
getPlan	plan	ISO 19115-2
getPlatforms	platform	ISO 19115-2
getSignificantEvents	significantEvent	ISO 19115-2
Code list OperationType	MI_OperationTypeCode	ISO 19115-2
REAL	real	ISO 19115-2
SIMULATED	simulated	ISO 19115-2
SYNTHESIZED	synthesized	ISO 19115-2
Interface Plan	MI_Plan	ISO 19115-2
getType	type	ISO 19115-2
getStatus	status	ISO 19115-2

getCitation	citation	ISO 19115-2
getOperations	operation	ISO 19115-2
getSatisfiedRequirements	satisfiedRequirement	ISO 19115-2
Interface Platform	MI_Platform	ISO 19115-2
getCitation	citation	ISO 19115-2
getIdentifier	identifier	ISO 19115-2
getDescription	description	ISO 19115-2
getSponsors	sponsor	ISO 19115-2
getInstruments	instrument	ISO 19115-2
Interface PlatformPass	MI_PlatformPass	ISO 19115-2
getIdentifier	identifier	ISO 19115-2
getExtent	extent	ISO 19115-2
getRelatedEvents	relatedEvent	ISO 19115-2
Code list Priority	MI_PriorityCode	ISO 19115-2
CRITICAL	critical	ISO 19115-2
HIGH_IMPORTANCE	highImportance	ISO 19115-2
MEDIUM_IMPORTANCE	mediumImportance	ISO 19115-2
LOW_IMPORTANCE	lowImportance	ISO 19115-2
Interface RequestedDate	MI_RequestedDate	ISO 19115-2
getRequestedDateOfCollection	requestedDateOfCollection	ISO 19115-2
getLatestAcceptableDate	latestAcceptableDate	ISO 19115-2
Interface Requirement	MI_Requirement	ISO 19115-2
getCitation	citation	ISO 19115-2
getIdentifier	identifier	ISO 19115-2
getRequestors	requestor	ISO 19115-2
getRecipients	recipient	ISO 19115-2
getPriority	priority	ISO 19115-2
getRequestedDate	requestedDate	ISO 19115-2
getExpiryDate	expiryDate	ISO 19115-2
getSatisfiedPlans	satisfiedPlan	ISO 19115-2
Code list Sequence	MI_SequenceCode	ISO 19115-2
START	start	ISO 19115-2
END	end	ISO 19115-2
INSTANTANEOUS	instantaneous	ISO 19115-2
Code list Trigger	MI_TriggerCode	ISO 19115-2
AUTOMATIC	automatic	ISO 19115-2

MANUAL	manual	ISO 19115-2
PRE_PROGRAMMED	preProgrammed	ISO 19115-2

Package org.opengis.metadata.citation

Interface Address	CI_Address	ISO 19115
getDeliveryPoints	deliveryPoint	ISO 19115
getCity	city	ISO 19115
getAdministrativeArea	administrativeArea	ISO 19115
getPostalCode	postalCode	ISO 19115
getCountry	country	ISO 19115
getElectronicMailAddresses	electronicMailAddress	ISO 19115
Interface Citation	CI_Citation	ISO 19115
getTitle	title	ISO 19115
getAlternateTitles	alternateTitle	ISO 19115
getDates	date	ISO 19115
getEdition	edition	ISO 19115
getEditionDate	editionDate	ISO 19115
getIdentifiers	identifier	ISO 19115
getCitedResponsibleParties	citedResponsibleParty	ISO 19115
getPresentationForms	presentationForm	ISO 19115
getSeries	series	ISO 19115
getOtherCitationDetails	otherCitationDetails	ISO 19115
getCollectiveTitle	collectiveTitle	ISO 19115
getISBN	ISBN	ISO 19115
getISSN	ISSN	ISO 19115
Interface CitationDate	CI_Date	ISO 19115
getDate	date	ISO 19115
getDateType	dateType	ISO 19115
Interface Contact	CI_Contact	ISO 19115
getPhone	phone	ISO 19115
getAddress	address	ISO 19115
getOnlineResource	onlineResource	ISO 19115
getHoursOfService	hoursOfService	ISO 19115
getContactInstructions	contactInstructions	ISO 19115
Code list DateType	CI_DateTypeCode	ISO 19115
CREATION	creation	ISO 19115
PUBLICATION	publication	ISO 19115

REVISION	revision	ISO 19115
Code list OnLineFunction	CI_OnLineFunctionCode	ISO 19115
DOWNLOAD	download	ISO 19115
INFORMATION	information	ISO 19115
OFFLINE_ACCESS	offlineAccess	ISO 19115
ORDER	order	ISO 19115
SEARCH	search	ISO 19115
Interface OnlineResource	CI_OnlineResource	ISO 19115
getLinkage	linkage	ISO 19115
getProtocol	protocol	ISO 19115
getApplicationProfile	applicationProfile	ISO 19115
getName	name	ISO 19115
getDescription	description	ISO 19115
getFunction	function	ISO 19115
Code list PresentationForm	CI_PresentationFormCode	ISO 19115
DOCUMENT_DIGITAL	documentDigital	ISO 19115
DOCUMENT_HARDCOPY	documentHardcopy	ISO 19115
IMAGE_DIGITAL	imageDigital	ISO 19115
IMAGE_HARDCOPY	imageHardcopy	ISO 19115
MAP_DIGITAL	mapDigital	ISO 19115
MAP_HARDCOPY	mapHardcopy	ISO 19115
MODEL_DIGITAL	modelDigital	ISO 19115
MODEL_HARDCOPY	modelHardcopy	ISO 19115
PROFILE_DIGITAL	profileDigital	ISO 19115
PROFILE_HARDCOPY	profileHardcopy	ISO 19115
TABLE_DIGITAL	tableDigital	ISO 19115
TABLE_HARDCOPY	tableHardcopy	ISO 19115
VIDEO_DIGITAL	videoDigital	ISO 19115
VIDEO_HARDCOPY	videoHardcopy	ISO 19115
Interface ResponsibleParty	CI_ResponsibleParty	ISO 19115
getIndividualName	individualName	ISO 19115
getOrganisationName	organisationName	ISO 19115
getPositionName	positionName	ISO 19115
getContactInfo	contactInfo	ISO 19115
getRole	role	ISO 19115
Code list Role	CI_RoleCode	ISO 19115
RESOURCE_PROVIDER	resourceProvider	ISO 19115
CUSTODIAN	custodian	ISO 19115
OWNER	owner	ISO 19115

USER	user	ISO 19115
DISTRIBUTOR	distributor	ISO 19115
ORIGINATOR	originator	ISO 19115
POINT_OF_CONTACT	pointOfContact	ISO 19115
PRINCIPAL_INVESTIGATOR	principalInvestigator	ISO 19115
PROCESSOR	processor	ISO 19115
PUBLISHER	publisher	ISO 19115
AUTHOR	author	ISO 19115
Interface Series	CI_Series	ISO 19115
getName	name	ISO 19115
getIssueIdentification	issueIdentification	ISO 19115
getPage	page	ISO 19115
Interface Telephone	CI_Telephone	ISO 19115
getVoices	voice	ISO 19115
getFacsimiles	facsimile	ISO 19115

Package org.opengis.metadata.constraint

Code list Classification	MD_ClassificationCode	ISO 19115
UNCLASSIFIED	unclassified	ISO 19115
RESTRICTED	restricted	ISO 19115
CONFIDENTIAL	confidential	ISO 19115
SECRET	secret	ISO 19115
TOP_SECRET	topSecret	ISO 19115
Interface Constraints	MD_Constraints	ISO 19115
getUseLimitations	useLimitation	ISO 19115
Interface LegalConstraints	MD_LegalConstraints	ISO 19115
getAccessConstraints	accessConstraints	ISO 19115
getUseConstraints	useConstraints	ISO 19115
getOtherConstraints	otherConstraints	ISO 19115
Code list Restriction	MD_RestrictionCode	ISO 19115
COPYRIGHT	copyright	ISO 19115
PATENT	patent	ISO 19115
PATENT_PENDING	patentPending	ISO 19115
TRADEMARK	trademark	ISO 19115
LICENSE	license	ISO 19115
INTELLECTUAL_PROPERTY_RIGHTS		

intellectualPropertyRights	ISO 19115	
RESTRICTED	restricted	ISO 19115
OTHER_RESTRICTIONS	otherRestrictions	ISO 19115
Interface SecurityConstraints	MD_SecurityConstraints	ISO 19115
getClassification	classification	ISO 19115
getUserNote	userNote	ISO 19115
getClassificationSystem	classificationSystem	ISO 19115
getHandlingDescription	handlingDescription	ISO 19115

Package org.opengis.metadata.content

Interface Band	MD_Band	ISO 19115
getMaxValue	maxValue	ISO 19115
getMinValue	minValue	ISO 19115
getUnits	units	ISO 19115
getPeakResponse	peakResponse	ISO 19115
getBitsPerValue	bitsPerValue	ISO 19115
getToneGradation	toneGradation	ISO 19115
getScaleFactor	scaleFactor	ISO 19115
getOffset	offset	ISO 19115
getBandBoundaryDefinition	bandBoundaryDefinition	ISO 19115-2
getNominalSpatialResolution	nominalSpatialResolution	ISO 19115-2
getTransferFunctionType	transferFunctionType	ISO 19115-2
getTransmittedPolarization	transmittedPolarization	ISO 19115-2
getDetectedPolarization	detectedPolarization	ISO 19115-2
Code list BandDefinition	MI_BandDefinition	ISO 19115-2
THREE_DB	3dB	ISO 19115-2
HALF_MAXIMUM	halfMaximum	ISO 19115-2
FIFTY_PERCENT	fiftyPercent	ISO 19115-2
ONE_OVER_E	oneOverE	ISO 19115-2
EQUIVALENT_WIDTH	equivalentWidth	ISO 19115-2
Interface ContentInformation	MD_ContentInformation	ISO 19115
Code list CoverageContentType	MD_CoverageContentTypeCode	ISO 19115
IMAGE	image	ISO 19115
THEMATIC_CLASSIFICATION	thematicClassification	ISO 19115
PHYSICAL_MEASUREMENT	physicalMeasurement	ISO 19115

Interface CoverageDescription	MD_CoverageDescription	ISO 19115
getAttributeDescription	attributeDescription	ISO 19115
getContentType	contentType	ISO 19115
getDimensions	dimension	ISO 19115
getRangeElementDescriptions	rangeElementDescription	ISO 19115-2
Interface FeatureCatalogueDescription	MD_FeatureCatalogueDescription	ISO 19115
isCompliant	complianceCode	ISO 19115
getLanguages	language	ISO 19115
isIncludedWithDataset	includedWithDataset	ISO 19115
getFeatureTypes	featureTypes	ISO 19115
getFeatureCatalogueCitations	featureCatalogueCitation	ISO 19115
Interface ImageDescription	MD_ImageDescription	ISO 19115
getIlluminationElevationAngle	illuminationElevationAngle	ISO 19115
getIlluminationAzimuthAngle	illuminationAzimuthAngle	ISO 19115
getImagingCondition	imagingCondition	ISO 19115
getImageQualityCode	imageQualityCode	ISO 19115
getCloudCoverPercentage	cloudCoverPercentage	ISO 19115
getProcessingLevelCode	processingLevelCode	ISO 19115
getCompressionGenerationQuantity	compressionGenerationQuantity	ISO 19115
getTriangulationIndicator	triangulationIndicator	ISO 19115
isRadiometricCalibrationDataAvailable	radiometricCalibrationDataAvailability	ISO 19115
isCameraCalibrationInformationAvailable	cameraCalibrationInformationAvailability	ISO 19115
isFilmDistortionInformationAvailable	filmDistortionInformationAvailability	ISO 19115
isLensDistortionInformationAvailable	lensDistortionInformationAvailability	ISO 19115
Code list ImagingCondition	MD_ImagingConditionCode	ISO 19115
BLURRED_IMAGE	blurredImage	ISO 19115
CLOUD	cloud	ISO 19115
DEGRADING_OBLIQUITY	degradingObliquity	ISO 19115
FOG	fog	ISO 19115
HEAVY_SMOKE_OR_DUST	heavySmokeOrDust	ISO 19115
NIGHT	night	ISO 19115
RAIN	rain	ISO 19115
SEMI_DARKNESS	semiDarkness	ISO 19115
SHADOW	shadow	ISO 19115
SNOW	snow	ISO 19115
TERRAIN_MASKING	terrainMasking	ISO 19115

Code list PolarizationOrientation	MI_PolarizationOrientationCode	ISO 19115-2
HORIZONTAL	horizontal	ISO 19115-2
VERTICAL	vertical	ISO 19115-2
LEFT_CIRCULAR	leftCircular	ISO 19115-2
RIGHT_CIRCULAR	rightCircular	ISO 19115-2
THETA	theta	ISO 19115-2
PHI	phi	ISO 19115-2
Interface RangeDimension	MD_RangeDimension	ISO 19115
getSequenceIdentifier	sequenceIdentifier	ISO 19115
getDescriptor	descriptor	ISO 19115
Interface RangeElementDescription	MI_RangeElementDescription	ISO 19115-2
getName	name	ISO 19115-2
getDefinition	definition	ISO 19115-2
getRangeElements	rangeElement	ISO 19115-2
Code list TransferFunctionType	MI_TransferFunctionTypeCode	ISO 19115-2
LINEAR	linear	ISO 19115-2
LOGARITHMIC	logarithmic	ISO 19115-2
EXPONENTIAL	exponential	ISO 19115-2

Package org.opengis.metadata.distribution

Interface DataFile	MX_DataFile	ISO 19139
getFeatureTypes	featureType	ISO 19139
getFileFormat	fileFormat	ISO 19139
Interface DigitalTransferOptions	MD_DigitalTransferOptions	ISO 19115
getUnitsOfDistribution	unitsOfDistribution	ISO 19115
getTransferSize	transferSize	ISO 19115
getOnLines	onLine	ISO 19115
getOffLine	offLine	ISO 19115
Interface Distribution	MD_Distribution	ISO 19115
getDistributionFormats	distributionFormat	ISO 19115
getDistributors	distributor	ISO 19115
getTransferOptions	transferOptions	ISO 19115
Interface Distributor	MD_Distributor	ISO 19115
getDistributorContact	distributorContact	ISO 19115

getDistributionOrderProcesses	distributionOrderProcess	ISO 19115
getDistributorFormats	distributorFormat	ISO 19115
getDistributorTransferOptions	distributorTransferOptions	ISO 19115
Interface Format	MD_Format	ISO 19115
getName	name	ISO 19115
getVersion	version	ISO 19115
getAmendmentNumber	amendmentNumber	ISO 19115
getSpecification	specification	ISO 19115
getFileDecompressionTechnique	fileDecompressionTechnique	ISO 19115
getFormatDistributors	formatDistributor	ISO 19115
Interface Medium	MD_Medium	ISO 19115
getName	name	ISO 19115
getDensities	density	ISO 19115
getDensityUnits	densityUnits	ISO 19115
getVolumes	volumes	ISO 19115
getMediumFormats	mediumFormat	ISO 19115
getMediumNote	mediumNote	ISO 19115
Code list MediumFormat	MD_MediumFormatCode	ISO 19115
CPIO	cpio	ISO 19115
TAR	tar	ISO 19115
HIGH_SIERRA	highSierra	ISO 19115
ISO_9660	iso9660	ISO 19115
ISO_9660_ROCK_RIDGE	iso9660RockRidge	ISO 19115
ISO_9660_APPLE_HFS	iso9660AppleHFS	ISO 19115
Code list MediumName	MD_MediumNameCode	ISO 19115
CD_ROM	cdRom	ISO 19115
DVD	dvd	ISO 19115
DVD_ROM	dvdRom	ISO 19115
FLOPPY_3_HALF_INCH	3halfInchFloppy	ISO 19115
FLOPPY_5_QUARTER_INCH	5quarterInchFloppy	ISO 19115
TAPE_7_TRACK	7trackTape	ISO 19115
TAPE_9_TRACK	9trackTape	ISO 19115
CARTRIDGE_3480	3480Cartridge	ISO 19115
CARTRIDGE_3490	3490Cartridge	ISO 19115
CARTRIDGE_3580	3580Cartridge	ISO 19115
CARTRIDGE_TAPE_4mm	4mmCartridgeTape	ISO 19115
CARTRIDGE_TAPE_8mm	8mmCartridgeTape	ISO 19115
CARTRIDGE_TAPE_1_QUARTER_INCH	1quarterInchCartridgeTape	ISO 19115
DIGITAL_LINEAR_TAPE	digitalLinearTape	ISO 19115

ON_LINE	onLine	ISO 19115
SATELLITE	satellite	ISO 19115
TELEPHONE_LINK	telephoneLink	ISO 19115
HARDCOPY	hardcopy	ISO 19115
Interface StandardOrderProcess	MD_StandardOrderProcess	ISO 19115
getFees	fees	ISO 19115
getPlannedAvailableDateTime	plannedAvailableDateTime	ISO 19115
getOrderingInstructions	orderingInstructions	ISO 19115
getTurnaround	turnaround	ISO 19115

Package org.opengis.metadata.extent

Interface BoundingPolygon	EX_BoundingPolygon	ISO 19115
getPolygons	polygon	ISO 19115
Interface Extent	EX_Extent	ISO 19115
getDescription	description	ISO 19115
getGeographicElements	geographicElement	ISO 19115
getTemporalElements	temporalElement	ISO 19115
getVerticalElements	verticalElement	ISO 19115
Interface GeographicBoundingBox	EX_GeographicBoundingBox	ISO 19115
getWestBoundLongitude	westBoundLongitude	ISO 19115
getEastBoundLongitude	eastBoundLongitude	ISO 19115
getSouthBoundLatitude	southBoundLatitude	ISO 19115
getNorthBoundLatitude	northBoundLatitude	ISO 19115
Interface GeographicDescription	EX_GeographicDescription	ISO 19115
getGeographicIdentifier	geographicIdentifier	ISO 19115
Interface GeographicExtent	EX_GeographicExtent	ISO 19115
getInclusion	extentTypeCode	ISO 19115
Interface SpatialTemporalExtent	EX_SpatialTemporalExtent	ISO 19115
getSpatialExtent	spatialExtent	ISO 19115
Interface TemporalExtent	EX_TemporalExtent	ISO 19115
getExtent	extent	ISO 19108

Interface VerticalExtent	EX_VerticalExtent	ISO 19115
getMinimumValue	minimumValue	ISO 19115
getMaximumValue	maximumValue	ISO 19115
getVerticalCRS	verticalCRS	ISO 19115

Package org.opengis.metadata.identification

Interface AggregateInformation	MD_AggregateInformation	ISO 19115
getAggregateDataSetName	aggregateDataSetName	ISO 19115
getAggregateDataSetIdentifier	aggregateDataSetIdentifier	ISO 19115
getAssociationType	associationType	ISO 19115
getInitiativeType	initiativeType	ISO 19115
Code list AssociationType	DS_AssociationTypeCode	ISO 19115
CROSS_REFERENCE	crossReference	ISO 19115
LARGER_WORD_CITATION	largerWorkCitation	ISO 19115
PART_OF_SEAMLESS_DATABASE	partOfSeamlessDatabase	ISO 19115
SOURCE	source	ISO 19115
STEREO_MATE	stereoMate	ISO 19115
Interface BrowseGraphic	MD_BrowseGraphic	ISO 19115
getFileName	fileName	ISO 19115
getFileDescription	fileDescription	ISO 19115
getFileType	fileType	ISO 19115
Code list CharacterSet	MD_CharacterSetCode	ISO 19115
UCS_2	ucs2	ISO 19115
UCS_4	ucs4	ISO 19115
UTF_7	utf7	ISO 19115
UTF_8	utf8	ISO 19115
UTF_16	utf16	ISO 19115
ISO_8859_1	8859part1	ISO 19115
ISO_8859_2	8859part2	ISO 19115
ISO_8859_3	8859part3	ISO 19115
ISO_8859_4	8859part4	ISO 19115
ISO_8859_5	8859part5	ISO 19115
ISO_8859_6	8859part6	ISO 19115
ISO_8859_7	8859part7	ISO 19115
ISO_8859_8	8859part8	ISO 19115
ISO_8859_9	8859part9	ISO 19115
ISO_8859_10	8859part10	ISO 19115
ISO_8859_11	8859part11	ISO 19115

ISO_8859_12	8859part12	ISO 19115
ISO_8859_13	8859part13	ISO 19115
ISO_8859_14	8859part14	ISO 19115
ISO_8859_15	8859part15	ISO 19115
ISO_8859_16	8859part16	ISO 19115
JIS	jis	ISO 19115
SHIFT_JIS	shiftJIS	ISO 19115
EUC_JP	eucJP	ISO 19115
US_ASCII	usAscii	ISO 19115
EBCDIC	ebcdic	ISO 19115
EUC_KR	eucKR	ISO 19115
BIG_5	big5	ISO 19115
GB2312	GB2312	ISO 19115
Interface DataIdentification	MD_DataIdentification	ISO 19115
getSpatialRepresentationTypes	spatialRepresentationType	ISO 19115
getSpatialResolutions	spatialResolution	ISO 19115
getLanguages	language	ISO 19115
getCharacterSets	characterSet	ISO 19115
getTopicCategories	topicCategory	ISO 19115
getEnvironmentDescription	environmentDescription	ISO 19115
getExtents	extent	ISO 19115
getSupplementalInformation	supplementalInformation	ISO 19115
Interface Identification	MD_Identification	ISO 19115
getCitation	citation	ISO 19115
getAbstract	abstract	ISO 19115
getPurpose	purpose	ISO 19115
getCredits	credit	ISO 19115
getStatus	status	ISO 19115
getPointOfContacts	pointOfContact	ISO 19115
getResourceMaintenances	resourceMaintenance	ISO 19115
getGraphicOverviews	graphicOverview	ISO 19115
getResourceFormats	resourceFormat	ISO 19115
getDescriptiveKeywords	descriptiveKeywords	ISO 19115
getResourceSpecificUsages	resourceSpecificUsage	ISO 19115
getResourceConstraints	resourceConstraints	ISO 19115
getAggregationInfo	aggregationInfo	ISO 19115
Code list InitiativeType	DS_InitiativeTypeCode	ISO 19115
CAMPAIGN	campaign	ISO 19115
COLLECTION	collection	ISO 19115
EXERCISE	exercise	ISO 19115
EXPERIMENT	experiment	ISO 19115

INVESTIGATION	investigation	ISO 19115
MISSION	mission	ISO 19115
SENSOR	sensor	ISO 19115
OPERATION	operation	ISO 19115
PLATFORM	platform	ISO 19115
PROCESS	process	ISO 19115
PROGRAM	program	ISO 19115
PROJECT	project	ISO 19115
STUDY	study	ISO 19115
TASK	task	ISO 19115
TRIAL	trial	ISO 19115
Interface Keywords	MD_Keywords	ISO 19115
getKeywords	keyword	ISO 19115
getType	type	ISO 19115
getThesaurusName	thesaurusName	ISO 19115
Code list KeywordType	MD_KeywordTypeCode	ISO 19115
DISCIPLINE	discipline	ISO 19115
PLACE	place	ISO 19115
STRATUM	stratum	ISO 19115
TEMPORAL	temporal	ISO 19115
THEME	theme	ISO 19115
Code list Progress	MD_ProgressCode	ISO 19115
COMPLETED	completed	ISO 19115
HISTORICAL_ARCHIVE	historicalArchive	ISO 19115
OBSOLETE	obsolete	ISO 19115
ON_GOING	onGoing	ISO 19115
PLANNED	planned	ISO 19115
REQUIRED	required	ISO 19115
UNDER_DEVELOPMENT	underDevelopment	ISO 19115
Interface RepresentativeFraction	MD_RepresentativeFraction	ISO 19115
doubleValue		Java
getDenominator	denominator	ISO 19115
equals		Java
hashCode		Java
Interface Resolution	MD_Resolution	ISO 19115
getEquivalentScale	equivalentScale	ISO 19115
getDistance	distance	ISO 19115

Interface ServiceIdentification	SV_ServiceIdentification	ISO 19115
Code list TopicCategory	MD_TopicCategoryCode	ISO 19115
FARMING	farming	ISO 19115
BIOTA	biota	ISO 19115
BOUNDARIES	boundaries	ISO 19115
CLIMATOLOGY_METEOROLOGY_ATMOSPHERE climatologyMeteorologyAtmosphere	ISO 19115	
ECONOMY	economy	ISO 19115
ELEVATION	elevation	ISO 19115
ENVIRONMENT	environment	ISO 19115
GEOSCIENTIFIC_INFORMATION	geoscientificInformation	ISO 19115
HEALTH	health	ISO 19115
IMAGERY_BASE_MAPS_EARTH_COVER imageryBaseMapsEarthCover	ISO 19115	
INTELLIGENCE_MILITARY	intelligenceMilitary	ISO 19115
INLAND_WATERS	inlandWaters	ISO 19115
LOCATION	location	ISO 19115
OCEANS	oceans	ISO 19115
PLANNING_CADASTRE	planningCadastre	ISO 19115
SOCIETY	society	ISO 19115
STRUCTURE	structure	ISO 19115
TRANSPORTATION	transportation	ISO 19115
UTILITIES_COMMUNICATION	utilitiesCommunication	ISO 19115
Interface Usage	MD_Usage	ISO 19115
getSpecificUsage	specificUsage	ISO 19115
getUsageDate	usageDateTime	ISO 19115
getUserDeterminedLimitations	userDeterminedLimitations	ISO 19115
getUserContactInfo	userContactInfo	ISO 19115

Package org.opengis.metadata.lineage

Interface Algorithm	LE_Algorithm	ISO 19115-2
getCitation	citation	ISO 19115-2
getDescription	description	ISO 19115-2
Interface Lineage	LI_Lineage	ISO 19115
getStatement	statement	ISO 19115
getProcessSteps	processStep	ISO 19115
getSources	source	ISO 19115

Interface NominalResolution	LE_NominalResolution	ISO 19115-2
getScanningResolution	scanningResolution	ISO 19115-2
getGroundResolution	groundResolution	ISO 19115-2
Interface Processing	LE_Processing	ISO 19115-2
getIdentifier	identifier	ISO 19115-2
getSoftwareReferences	softwareReference	ISO 19115-2
getProcedureDescription	procedureDescription	ISO 19115-2
getDocumentations	documentation	ISO 19115-2
getRunTimeParameters	runTimeParameters	ISO 19115-2
getAlgorithms	algorithm	ISO 19115-2
Interface ProcessStep	LI_ProcessStep	ISO 19115
getDescription	description	ISO 19115
getRationale	rationale	ISO 19115
getDate	dateTime	ISO 19115
getProcessors	processor	ISO 19115
getSources	source	ISO 19115
getOutputs	output	ISO 19115-2
getProcessingInformation	processingInformation	ISO 19115-2
getReports	report	ISO 19115-2
Interface ProcessStepReport	LE_ProcessStepReport	ISO 19115-2
getName	name	ISO 19115-2
getDescription	description	ISO 19115-2
getFileType	fileType	ISO 19115-2
Interface Source	LI_Source	ISO 19115
getDescription	description	ISO 19115
getScaleDenominator	scaleDenominator	ISO 19115
getSourceReferenceSystem	sourceReferenceSystem	ISO 19115
getSourceCitation	sourceCitation	ISO 19115
getSourceExtents	sourceExtent	ISO 19115
getSourceSteps	sourceStep	ISO 19115
getProcessedLevel	processedLevel	ISO 19115-2
getResolution	resolution	ISO 19115-2

Package org.opengis.metadata.maintenance

Code list MaintenanceFrequency	MD_MaintenanceFrequencyCode	ISO 19115
CONTINUAL	continual	ISO 19115
DAILY	daily	ISO 19115

WEEKLY	weekly	ISO 19115
FORTNIGHTLY	fortnightly	ISO 19115
MONTHLY	monthly	ISO 19115
QUARTERLY	quarterly	ISO 19115
BIANNUALLY	biannually	ISO 19115
ANNUALLY	annually	ISO 19115
AS_NEEDED	asNeeded	ISO 19115
IRREGULAR	irregular	ISO 19115
NOT_PLANNED	notPlanned	ISO 19115
UNKNOWN	unknown	ISO 19115
Interface MaintenanceInformation	MD_MaintenanceInformation	ISO 19115
getMaintenanceAndUpdateFrequency	maintenanceAndUpdateFrequency	ISO 19115
getDateOfNextUpdate	dateOfNextUpdate	ISO 19115
getUserDefinedMaintenanceFrequency	userDefinedMaintenanceFrequency	ISO 19115
getUpdateScopes	updateScope	ISO 19115
getUpdateScopeDescriptions	updateScopeDescription	ISO 19115
getMaintenanceNotes	maintenanceNote	ISO 19115
getContacts	contact	ISO 19115
Code list ScopeCode	MD_ScopeCode	ISO 19115
ATTRIBUTE	attribute	ISO 19115
ATTRIBUTE_TYPE	attributeType	ISO 19115
COLLECTION_HARDWARE	collectionHardware	ISO 19115
COLLECTION_SESSION	collectionSession	ISO 19115
DATASET	dataset	ISO 19115
SERIES	series	ISO 19115
NON_GEOGRAPHIC_DATASET	nonGeographicDataset	ISO 19115
DIMENSION_GROUP	dimensionGroup	ISO 19115
FEATURE	feature	ISO 19115
FEATURE_TYPE	featureType	ISO 19115
PROPERTY_TYPE	propertyType	ISO 19115
FIELD_SESSION	fieldSession	ISO 19115
SOFTWARE	software	ISO 19115
SERVICE	service	ISO 19115
MODEL	model	ISO 19115
TILE	tile	ISO 19115
Interface ScopeDescription	MD_ScopeDescription	ISO 19115
getAttributes	attributes	ISO 19115
getFeatures	features	ISO 19115
getFeatureInstances	featureInstances	ISO 19115
getAttributeInstances	attributeInstances	ISO 19115
getDataset	dataset	ISO 19115

getOther	other	ISO 19115

Package org.opengis.metadata.quality

Interface AbsoluteExternalPositionalAccuracy	DQ_AbsoluteExternalPositionalAccuracy	ISO 19115
Interface AccuracyOfATimeMeasurement	DQ_AccuracyOfATimeMeasurement	ISO 19115
Interface Completeness	DQ_Completeness	ISO 19115
Interface CompletenessCommission	DQ_CompletenessCommission	ISO 19115
Interface CompletenessOmission	DQ_CompletenessOmission	ISO 19115
Interface ConceptualConsistency	DQ_ConceptualConsistency	ISO 19115
Interface ConformanceResult	DQ_ConformanceResult	ISO 19115
getSpecification	specification	ISO 19115
getExplanation	explanation	ISO 19115
pass	pass	ISO 19115
Interface CoverageResult	QE_CoverageResult	ISO 19115-2
getSpatialRepresentationType	spatialRepresentationType	ISO 19115-2
getResultSpatialRepresentation	resultSpatialRepresentation	ISO 19115-2
getResultContentDescription	resultContentDescription	ISO 19115-2
getResultFormat	resultFormat	ISO 19115-2
getResultFile	resultFile	ISO 19139
Interface DataQuality	DQ_DataQuality	ISO 19115
getScope	scope	ISO 19115
getReports	report	ISO 19115
getLineage	lineage	ISO 19115
Interface DomainConsistency	DQ_DomainConsistency	ISO 19115
Interface Element	DQ_Element	ISO 19115
getNamesOfMeasure	nameOfMeasure	ISO 19115
getMeasureIdentification	measureIdentification	ISO 19115
getMeasureDescription	measureDescription	ISO 19115

getEvaluationMethodType	evaluationMethodType	ISO 19115
getEvaluationMethodDescription	evaluationMethodDescription	ISO 19115
getEvaluationProcedure	evaluationProcedure	ISO 19115
getDates	dateTime	ISO 19115
getResults	result	ISO 19115
Code list EvaluationMethodType	DQ_EvaluationMethodTypeCode	ISO 19115
DIRECT_INTERNAL	directInternal	ISO 19115
DIRECT_EXTERNAL	directExternal	ISO 19115
INDIRECT	indirect	ISO 19115
Interface FormatConsistency	DQ_FormatConsistency	ISO 19115
Interface GriddedDataPositionalAccuracy	DQ_GriddedDataPositionalAccuracy	ISO 19115
Interface LogicalConsistency	DQ_LogicalConsistency	ISO 19115
Interface NonQuantitativeAttributeAccuracy	DQ_NonQuantitativeAttributeAccuracy	ISO 19115
Interface PositionalAccuracy	DQ_PositionalAccuracy	ISO 19115
Interface QuantitativeAttributeAccuracy	DQ_QuantitativeAttributeAccuracy	ISO 19115
Interface QuantitativeResult	DQ_QuantitativeResult	ISO 19115
getValues	value	ISO 19115
getValueType	valueType	ISO 19115
getValueUnit	valueUnit	ISO 19115
getErrorStatistic	errorStatistic	ISO 19115
Interface RelativeInternalPositionalAccuracy	DQ_RelativeInternalPositionalAccuracy	ISO 19115
Interface Result	DQ_Result	ISO 19115
Interface Scope	DQ_Scope	ISO 19115
getLevel	level	ISO 19115
getLevelDescription	levelDescription	ISO 19115
getExtent	extent	ISO 19115
Interface TemporalAccuracy	DQ_TemporalAccuracy	ISO 19115
Interface TemporalConsistency	DQ_TemporalConsistency	ISO 19115

Interface TemporalValidity	DQ_TemporalValidity	ISO 19115
Interface ThematicAccuracy	DQ_ThematicAccuracy	ISO 19115
Interface ThematicClassificationCorrectness	DQ_ThematicClassificationCorrectness	ISO 19115
Interface TopologicalConsistency	DQ_TopologicalConsistency	ISO 19115
Interface Usability	QE_Usability	ISO 19115-2

Package org.opengis.metadata.spatial

Code list CellGeometry	MD_CellGeometryCode	ISO 19115
POINT	point	ISO 19115
AREA	area	ISO 19115
Interface Dimension	MD_Dimension	ISO 19115
getDimensionName	dimensionName	ISO 19115
getDimensionSize	dimensionSize	ISO 19115
getResolution	resolution	ISO 19115
Code list DimensionNameType	MD_DimensionNameTypeCode	ISO 19115
ROW	row	ISO 19115
COLUMN	column	ISO 19115
VERTICAL	vertical	ISO 19115
TRACK	track	ISO 19115
CROSS_TRACK	crossTrack	ISO 19115
LINE	line	ISO 19115
SAMPLE	sample	ISO 19115
TIME	time	ISO 19115
Interface GCP	MI_GCP	ISO 19115-2
getGeographicCoordinates	geographicCoordinates	ISO 19115-2
getAccuracyReports	accuracyReport	ISO 19115-2
Interface GCPCollection	MI_GCPCollection	ISO 19115-2
getCollectionIdentification	collectionIdentification	ISO 19115-2
getCollectionName	collectionName	ISO 19115-2
getCoordinateReferenceSystem	coordinateReferenceSystem	ISO 19115-2
getGCPs	gcp	ISO 19115-2

Interface GeolocationInformation	MI_GeolocationInformation	ISO 19115-2
getQualityInfo	qualityInfo	ISO 19115-2
Interface GeometricObjects	MD_GeometricObjects	ISO 19115
getGeometricObjectType	geometricObjectType	ISO 19115
getGeometricObjectCount	geometricObjectCount	ISO 19115
Code list GeometricObjectType	MD_GeometricObjectTypeCode	ISO 19115
COMPLEX	complex	ISO 19115
COMPOSITE	composite	ISO 19115
CURVE	curve	ISO 19115
POINT	point	ISO 19115
SOLID	solid	ISO 19115
SURFACE	surface	ISO 19115
Interface Georectified	MD_Georectified	ISO 19115
isCheckPointAvailable	checkPointAvailability	ISO 19115
getCheckPointDescription	checkPointDescription	ISO 19115
getCornerPoints	cornerPoints	ISO 19115
getCenterPoint	centerPoint	ISO 19115
getPointInPixel	pointInPixel	ISO 19115
getTransformationDimensionDescription	transformationDimensionDescription	ISO 19115
getTransformationDimensionMapping	transformationDimensionMapping	ISO 19115
getCheckPoints	checkPoint	ISO 19115-2
Interface Georeferenceable	MD_Georeferenceable	ISO 19115
isControlPointAvailable	controlPointAvailability	ISO 19115
isOrientationParameterAvailable	orientationParameterAvailability	ISO 19115
getOrientationParameterDescription	orientationParameterDescription	ISO 19115
getGeoreferencedParameters	georeferencedParameters	ISO 19115
getParameterCitations	parameterCitation	ISO 19115
getGeolocationInformation	geolocationInformation	ISO 19115-2
Interface GridSpatialRepresentation	MD_GridSpatialRepresentation	ISO 19115
getNumberOfDimensions	numberOfDimensions	ISO 19115
getAxisDimensionProperties	axisDimensionProperties	ISO 19115
getCellGeometry	cellGeometry	ISO 19115
isTransformationParameterAvailable	transformationParameterAvailability	ISO 19115
Code list PixelOrientation	MD_PixelOrientationCode	ISO 19115
CENTER	center	ISO 19115
LOWER_LEFT	lowerLeft	ISO 19115

LOWER_RIGHT	lowerRight	ISO 19115
UPPER_RIGHT	upperRight	ISO 19115
UPPER_LEFT	upperLeft	ISO 19115
Interface SpatialRepresentation	MD_SpatialRepresentation	ISO 19115
Code list SpatialRepresentationType	MD_SpatialRepresentationTypeCode	ISO 19115
VECTOR	vector	ISO 19115
GRID	grid	ISO 19115
TEXT_TABLE	textTable	ISO 19115
TIN	tin	ISO 19115
STEREO_MODEL	stereoModel	ISO 19115
VIDEO	video	ISO 19115
Code list TopologyLevel	MD_TopologyLevelCode	ISO 19115
GEOMETRY_ONLY	geometryOnly	ISO 19115
TOPOLOGY_1D	topology1D	ISO 19115
PLANAR_GRAPH	planarGraph	ISO 19115
FULL_PLANAR_GRAPH	fullPlanarGraph	ISO 19115
SURFACE_GRAPH	surfaceGraph	ISO 19115
FULL_SURFACE_GRAPH	fullSurfaceGraph	ISO 19115
TOPOLOGY_3D	topology3D	ISO 19115
FULL_TOPOLOGY_3D	fullTopology3D	ISO 19115
ABSTRACT	abstract	ISO 19115
Interface VectorSpatialRepresentation	MD_VectorSpatialRepresentation	ISO 19115
getTopologyLevel	topologyLevel	ISO 19115
getGeometricObjects	geometricObjects	ISO 19115

Package org.opengis.parameter

Interface GeneralParameterDescriptor	CC_GeneralOperationParameter	ISO 19111
createValue		
getMinimumOccurs	minimumOccurs	ISO 19111
getMaximumOccurs	maximumOccurs	ISO 19111
Interface GeneralParameterValue	CC_GeneralParameterValue	ISO 19111
getDescriptor	parameter	ISO 19111
clone		Java

Class InvalidParameterCardinalityException		
getParameterName		
Class InvalidParameterNameException	GC_InvalidParameterName	OGC 01004
getParameterName		
Class InvalidParameterTypeException		
getParameterName		
Class InvalidParameterValueException	GC_InvalidParameterValue	OGC 01004
getParameterName		
getValue		
Interface ParameterDescriptor	CC_OperationParameter	ISO 19111
getValueClass	type	ISO 19111
getValidValues		
getDefaultValue	defaultValue	ISO 19111
getMinimumValue	minimumValue	ISO 19111
getMaximumValue	maximumValue	ISO 19111
getUnit		
Interface ParameterDescriptorGroup	CC_OperationParameterGroup	ISO 19111
descriptors	parameter	ISO 19111
descriptor		
Class ParameterNotFoundException		
getParameterName		
Interface ParameterValue	CC_ParameterValue	ISO 19111
getUnit		
doubleValue		
intValue	integerValue	ISO 19111
booleanValue	booleanValue	ISO 19111
stringValue	stringValue	ISO 19111
doubleValueList	valueList	ISO 19111
intValueList	integerValueList	ISO 19111
valueFile	valueFile	ISO 19111
getValue	value	ISO 19111
setValue		
Interface ParameterValueGroup	CC_ParameterValueGroup	ISO 19111
values	parameterValue	ISO 19111
parameter		

groups
addGroup

Package org.opengis.referencing

Interface AuthorityFactory	CS_CoordinateSystemAuthorityFactory	OGC 01009
getAuthority	getAuthority	OGC 01009
getAuthorityCodes		
getDescriptionText	descriptionText	OGC 01009
createObject		
Interface IdentifiedObject	IO_IdentifiedObject	ISO 19111
getName	name	ISO 19111
getAlias	alias	ISO 19111
getIdentifiers	identifier	ISO 19111
getRemarks	remarks	ISO 19111
toWKT		
Class NoSuchAuthorityCodeException		
getAuthority		
getAuthorityCode		
Interface ObjectFactory		
Interface ReferenceIdentifier	RS_Identifier	ISO 19115
getCodeSpace	codeSpace	ISO 19115
getVersion	version	ISO 19115
Interface ReferenceSystem	RS_ReferenceSystem	ISO 19115
getDomainOfValidity	domainOfValidity	ISO 19111
getScope	scope	ISO 19111

Package org.opengis.referencing.crs

Interface CompoundCRS	SC_CompoundCRS	ISO 19111
getComponents	componentReferenceSystem	ISO 19111
Interface CoordinateReferenceSystem	SC_CRS	ISO 19111
getCoordinateSystem		

Interface CRSAuthorityFactory	CS_CoordinateSystemAuthorityFactory	OGC 01009
createCoordinateReferenceSystem	createHorizontalCoordinateSystem	OGC 01009
createCompoundCRS	createCompoundCoordinateSystem	OGC 01009
createDerivedCRS		
createEngineeringCRS		
createGeographicCRS	createGeographicCoordinateSystem	OGC 01009
createGeocentricCRS		
createImageCRS		
createProjectedCRS	createProjectedCoordinateSystem	OGC 01009
createTemporalCRS		
createVerticalCRS	createVerticalCoordinateSystem	OGC 01009
Interface CRSFactory	CS_CoordinateSystemFactory	OGC 01009
createCompoundCRS	createCompoundCoordinateSystem	OGC 01009
createEngineeringCRS	createLocalCoordinateSystem	OGC 01009
createImageCRS		
createTemporalCRS		
createVerticalCRS	createVerticalCoordinateSystem	OGC 01009
createGeocentricCRS		
createGeographicCRS	createGeographicCoordinateSystem	OGC 01009
createDerivedCRS	createFittedCoordinateSystem	OGC 01009
createProjectedCRS	createProjectedCoordinateSystem	OGC 01009
createFromXML	createFromXML	OGC 01009
createFromWKT	createFromWKT	OGC 01009
Interface DerivedCRS	SC_DerivedCRS	ISO 19111
Interface EngineeringCRS	SC_EngineeringCRS	ISO 19111
Interface GeneralDerivedCRS	SC_GeneralDerivedCRS	ISO 19111
getBaseCRS	baseCRS	ISO 19111
getConversionFromBase	conversion	ISO 19111
Interface GeocentricCRS	SC_GeocentricCRS	ISO 19111
getCoordinateSystem	coordinateSystem	ISO 19111
Interface GeodeticCRS	SC_GeodeticCRS	ISO 19111
Interface GeographicCRS	SC_GeographicCRS	ISO 19111
getCoordinateSystem	coordinateSystem	ISO 19111

Interface ImageCRS	SC_ImageCRS	ISO 19111
Interface ProjectedCRS	SC_ProjectedCRS	ISO 19111
getCoordinateSystem	coordinateSystem	ISO 19111
getDatum	datum	ISO 19111
Interface SingleCRS	SC_SingleCRS	ISO 19111
getDatum	datum	ISO 19111
Interface TemporalCRS	SC_TemporalCRS	ISO 19111
Interface VerticalCRS	SC_VerticalCRS	ISO 19111

Package org.opengis.referencing.cs

Interface AffineCS	CS_AffineCS	ISO 19111
Code list AxisDirection	CS_AxisDirection	ISO 19111
OTHER	CS_AO_Other	OGC 01009
NORTH	north	ISO 19111
NORTH_NORTH_EAST	northNorthEast	ISO 19111
NORTH_EAST	northEast	ISO 19111
EAST_NORTH_EAST	eastNorthEast	ISO 19111
EAST	east	ISO 19111
EAST_SOUTH_EAST	eastSouthEast	ISO 19111
SOUTH_EAST	southEast	ISO 19111
SOUTH_SOUTH_EAST	southSouthEast	ISO 19111
SOUTH	south	ISO 19111
SOUTH_SOUTH_WEST	southSouthWest	ISO 19111
SOUTH_WEST	southWest	ISO 19111
WEST_SOUTH_WEST	westSouthWest	ISO 19111
WEST	west	ISO 19111
WEST_NORTH_WEST	westNorthWest	ISO 19111
NORTH_WEST	northWest	ISO 19111
NORTH_NORTH_WEST	northNorthWest	ISO 19111
UP	up	ISO 19111
DOWN	down	ISO 19111
GEOCENTRIC_X	geocentricX	ISO 19111
GEOCENTRIC_Y	geocentricY	ISO 19111
GEOCENTRIC_Z	geocentricZ	ISO 19111
FUTURE	future	ISO 19111

PAST	past	ISO 19111
COLUMN_POSITIVE	columnPositive	ISO 19111
COLUMN_NEGATIVE	columnNegative	ISO 19111
ROW_POSITIVE	rowPositive	ISO 19111
ROW_NEGATIVE	rowNegative	ISO 19111
DISPLAY_RIGHT	displayRight	ISO 19111
DISPLAY_LEFT	displayLeft	ISO 19111
DISPLAY_UP	displayUp	ISO 19111
DISPLAY_DOWN	displayDown	ISO 19111
Interface CartesianCS	CS_CartesianCS	ISO 19111
Interface CoordinateSystem	CS_CoordinateSystem	ISO 19111
getDimension		
getAxis	axis	ISO 19111
Interface CoordinateSystemAxis	CS_CoordinateSystemAxis	ISO 19111
getAbbreviation	axisAbbrev	ISO 19111
getDirection	axisDirection	ISO 19111
getMinimumValue	minimumValue	ISO 19111
getMaximumValue	maximumValue	ISO 19111
getRangeMeaning	rangeMeaning	ISO 19111
getUnit	axisUnitID	ISO 19111
Interface CSAuthorityFactory		
createCoordinateSystem		
createCartesianCS		
createPolarCS		
createCylindricalCS		
createSphericalCS		
createEllipsoidalCS		
createVerticalCS		
createTimeCS		
createCoordinateSystemAxis		
createUnit	createLinearUnit	OGC 01009
Interface CSFactory		
createCoordinateSystemAxis		
createCartesianCS		
createAffineCS		
createPolarCS		
createCylindricalCS		
createSphericalCS		
createEllipsoidalCS		

createVerticalCS createTimeCS createLinearCS createUserDefinedCS		
Interface CylindricalCS	CS_CylindricalCS	ISO 19111
Interface EllipsoidalCS	CS_EllipsoidalCS	ISO 19111
Interface LinearCS	CS_LinearCS	ISO 19111
Interface PolarCS	CS_PolarCS	ISO 19111
Code list RangeMeaning	CS_RangeMeaning	ISO 19111
EXACT	exact	ISO 19111
WRAPAROUND	wraparound	ISO 19111
Interface SphericalCS	CS_SphericalCS	ISO 19111
Interface TimeCS	CS_TimeCS	ISO 19111
Interface UserDefinedCS	CS_UserDefinedCS	ISO 19111
Interface VerticalCS	CS_VerticalCS	ISO 19111

Package org.opengis.referencing.datum

Interface Datum	CD_Datum	ISO 19111
getAnchorPoint	anchorPoint	ISO 19111
getRealizationEpoch	realizationEpoch	ISO 19111
getDomainOfValidity	domainOfValidity	ISO 19111
getScope	scope	ISO 19111
Interface DatumAuthorityFactory	CS_CoordinateSystemAuthorityFactory	OGC 01009
createDatum createEngineeringDatum createImageDatum		
createVerticalDatum	createVerticalDatum	OGC 01009
createTemporalDatum		
createGeodeticDatum	createHorizontalDatum	OGC 01009
createEllipsoid	createEllipsoid	OGC 01009

createPrimeMeridian	createPrimeMeridian	OGC 01009
Interface DatumFactory	CS_CoordinateSystemFactory	OGC 01009
createEngineeringDatum	createLocalDatum	OGC 01009
createGeodeticDatum	createHorizontalDatum	OGC 01009
createImageDatum		
createTemporalDatum		
createVerticalDatum	createVerticalDatum	OGC 01009
createEllipsoid	createEllipsoid	OGC 01009
createFlattenedSphere	createFlattenedSphere	OGC 01009
createPrimeMeridian	createPrimeMeridian	OGC 01009
Interface Ellipsoid	CD_Ellipsoid	ISO 19111
getAxisUnit	getAxisUnit	OGC 01009
getSemiMajorAxis	semiMajorAxis	ISO 19111
getSemiMinorAxis	semiMinorAxis	ISO 19111
getInverseFlattening	inverseFlattening	ISO 19111
isIvfDefinitive	isIvfDefinitive	OGC 01009
isSphere	isSphere	ISO 19111
Interface EngineeringDatum	CD_EngineeringDatum	ISO 19111
Interface GeodeticDatum	CD_GeodeticDatum	ISO 19111
getEllipsoid	ellipsoid	ISO 19111
getPrimeMeridian	primeMeridian	ISO 19111
Interface ImageDatum	CD_ImageDatum	ISO 19111
getPixelInCell	pixelInCell	ISO 19111
Code list PixelInCell	CD_PixelInCell	ISO 19111
CELL_CENTER	cell center	ISO 19111
CELL_CORNER	cell corner	ISO 19111
Interface PrimeMeridian	CD_PrimeMeridian	ISO 19111
getGreenwichLongitude	greenwichLongitude	ISO 19111
getAngularUnit	getAngularUnit	OGC 01009
Interface TemporalDatum	CD_TemporalDatum	ISO 19111
getOrigin	origin	ISO 19111
Interface VerticalDatum	CD_VerticalDatum	ISO 19111
getVerticalDatumType	vertDatumType	ISO 19111

Code list VerticalDatumType	CD_VerticalDatumType	ISO 19111
OTHER_SURFACE	other surface	ISO 19111
GEOIDAL	geoidal	ISO 19111
DEPTH	depth	ISO 19111
BAROMETRIC	barometric	ISO 19111

Package org.opengis.referencing.operation

Interface ConcatenatedOperation	CC_ConcatenatedOperation	ISO 19111
getOperations	coordOperation	ISO 19111
Interface ConicProjection		
Interface Conversion	CC_Conversion	ISO 19111
getSourceCRS	sourceCRS	ISO 19111
getTargetCRS	targetCRS	ISO 19111
getOperationVersion	operationVersion	ISO 19111
Interface CoordinateOperation	CC_CoordinateOperation	ISO 19111
getSourceCRS	sourceCRS	ISO 19111
getTargetCRS	targetCRS	ISO 19111
getOperationVersion	operationVersion	ISO 19111
getCoordinateOperationAccuracy	coordinateOperationAccuracy	ISO 19111
getDomainOfValidity	domainOfValidity	ISO 19111
getScope	scope	ISO 19111
getMathTransform	getMathTransform	OGC 01009
Interface CoordinateOperationAuthorityFactory	CT_CoordinateTransformationAuthorityFactory	OGC 01009
createOperationMethod		
createCoordinateOperation	createFromTransformationCode	OGC 01009
createFromCoordinateReferenceSystemCodes	createFromCoordinateSystemCodes	OGC 01009
Interface CoordinateOperationFactory	CT_CoordinateTransformationFactory	OGC 01009
createOperation	createFromCoordinateSystems	OGC 01009
createConcatenatedOperation		
createDefiningConversion		
Interface CylindricalProjection		

Interface Formula	CC_Formula	ISO 19111
getFormula	formula	ISO 19111
getCitation	formulaCitation	ISO 19111
Interface MathTransform	CT_MathTransform	OGC 01009
getSourceDimensions	getDimSource	OGC 01009
getTargetDimensions	getDimTarget	OGC 01009
transform	transform	OGC 01009
transform	transformList	OGC 01009
derivative	derivative	OGC 01009
inverse	inverse	OGC 01009
isIdentity	isIdentity	OGC 01009
toWKT	getWKT	OGC 01009
Interface MathTransform1D		
Interface MathTransform2D		
createTransformedShape		
Interface MathTransformFactory	CT_MathTransformFactory	OGC 01009
getAvailableMethods		
getLastMethodUsed		
getDefaultParameters		
createBaseToDerived		
createParameterizedTransform	createParameterizedTransform	OGC 01009
createAffineTransform	createAffineTransform	OGC 01009
createConcatenatedTransform	createConcatenatedTransform	OGC 01009
createPassThroughTransform	createPassThroughTransform	OGC 01009
createFromXML	createFromXML	OGC 01009
createFromWKT	createFromWKT	OGC 01009
Interface Matrix	PT_Matrix	OGC 01009
getNumRow		Vecmath
getNumCol		Vecmath
getElement		Vecmath
setElement		Vecmath
isIdentity		
clone		Java
Class NoninvertibleTransformException		
Interface OperationMethod	CC_OperationMethod	ISO 19111
getFormula	formulaReference	ISO 19111

getSourceDimensions	sourceDimensions	ISO 19111
getTargetDimensions	targetDimensions	ISO 19111
getParameters	parameter	ISO 19111
Class OperationNotFoundException		
Interface PassThroughOperation	CC_PassThroughOperation	ISO 19111
getOperation	coordOperation	ISO 19111
getModifiedCoordinates	modifiedCoordinate	ISO 19111
Interface PlanarProjection		
Interface Projection		
Interface SingleOperation	CC_SingleOperation	ISO 19111
getMethod	method	ISO 19111
getParameterValues	parameterValue	ISO 19111
Interface Transformation	CC_Transformation	ISO 19111
getSourceCRS	sourceCRS	ISO 19111
getTargetCRS	targetCRS	ISO 19111
getOperationVersion	operationVersion	ISO 19111
Class TransformException		
getLastCompletedTransform		
setLastCompletedTransform		

Package org.opengis.util

Class CodeList	CodeList	ISO 19103
valueOf		
family		
names		
name		
identifier		
ordinal		
equals		Java
toString		Java
Interface CodeList.Filter		
accept		

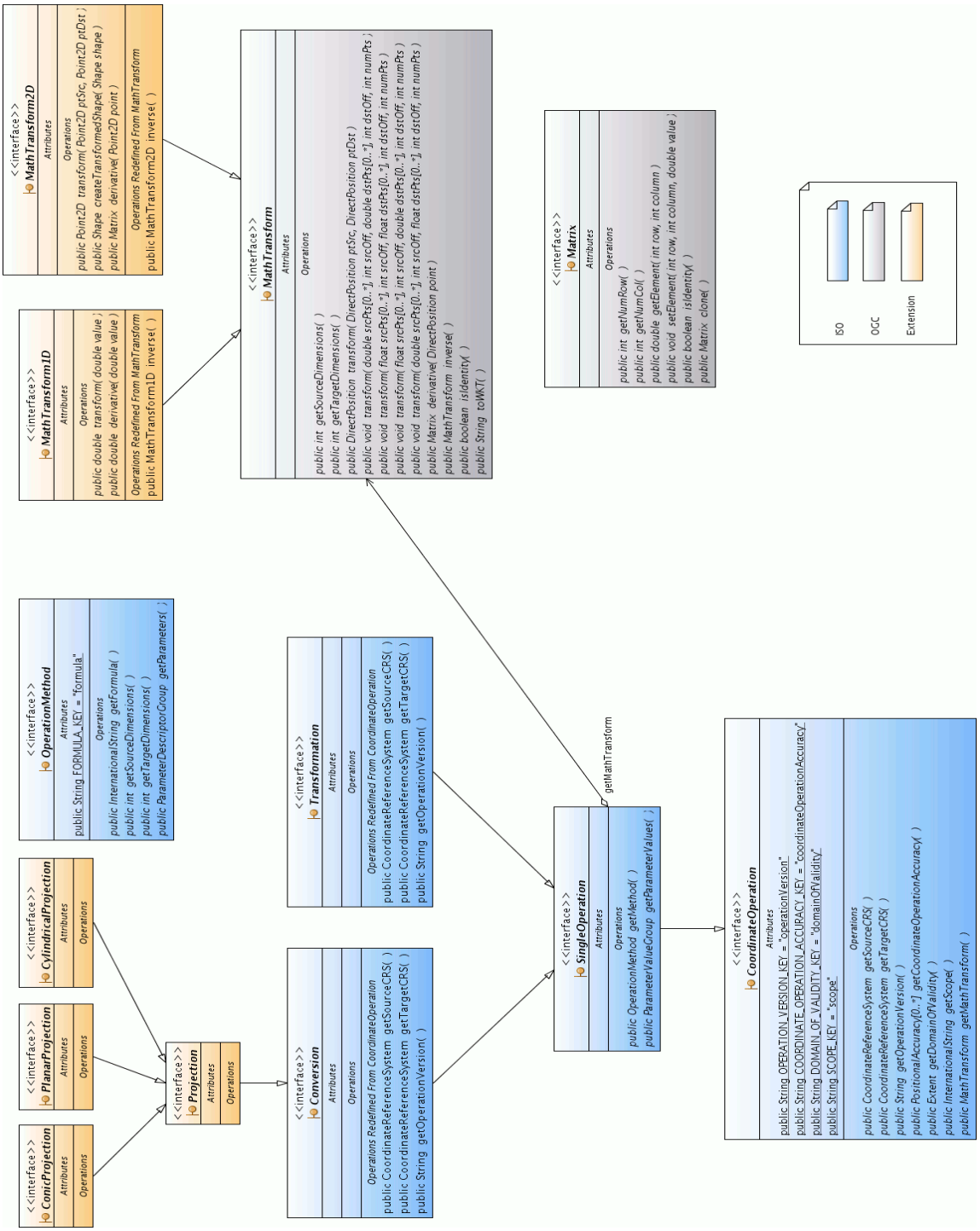
codename		
Interface Factory		
getVendor		
Class FactoryException		
Interface GenericName	GenericName	ISO 19103
scope	scope	ISO 19103
depth	depth	ISO 19103
getParsedNames	parsedName	ISO 19103
head	head	ISO 19103
tip		
toFullyQualifiedName		
push	push	ISO 19103
toString		Java
toInternationalString		
Interface InternationalString		
Interface LocalName	LocalName	ISO 19103
Interface MemberName	MemberName	ISO 19103
getAttributeType	attributeType	ISO 19103
Interface NameFactory		
createInternationalString		
createNameSpace		
createTypeName		
createLocalName		
createGenericName		
parseGenericName		
Interface NameSpace	NameSpace	ISO 19103
isGlobal	isGlobal	ISO 19103
name	name	ISO 19103
Class NoSuchIdentifierException		
getIdentifierCode		
Interface Record	Record	ISO 19103
getRecordType	recordType	ISO 19103

getAttributes	memberValue	ISO 19103
locate	locate	ISO 19103
set		
Interface RecordSchema	RecordSchema	ISO 19103
getSchemaName	schemaName	ISO 19103
getDescription	description	ISO 19103
locate	locate	ISO 19103
Interface RecordType	RecordType	ISO 19103
getContainer		
getMemberTypes	memberTypes	ISO 19103
getMembers		
locate	locate	ISO 19103
isInstance		
Interface ScopedName	ScopedName	ISO 19103
tail	tail	ISO 19103
path		
Interface Type	Type	ISO 19103
getTypeName	typeName	ISO 19103
Interface TypeName	TypeName	ISO 19103

Annex D

(informative)

UML Diagram for referencing Operation types



Annex E

(Informative)

Departures from the ISO/OGC standards

The following sections list all the departures from the ISO standards taken by the GeoAPI interface library. The rationale for these departures fall into the following categories:

- Departures due to constraints of the Java language
- Departures due to historical reasons
- Departures for harmonization between the different specifications
- Departures for closer integration with the Java environment
- Changes of name without change in functionality
- Generalizations due to relaxation of ISO/OGC restrictions
- Addition of elements not in the ISO/OGC specifications
- Extensions for convenience, without introduction of new functionality

E.1 Departures due to constraints of the Java language

Unions in org.opengis.referencing.cs package

ISO 19111 defines `GeodeticCS`, `EngineeringCS` and `ImageCS` unions for type safety, which ensures, for example, that a `GeodeticCRS` only be associated to a `CartesianCS`, an `EllipsoidalCS` or a `SphericalCS`. However the union construct found in some languages like C/C++ is not available in Java. In the particular case of `ImageCS`, the same type-safety objective can be obtained through a slight change in the interface hierarchy (see the departure documented in `CartesianCS`). For the other two unions (`GeodeticCS` and `EngineeringCS`), no workaround is proposed.

Parent of CartesianCS interface

ISO 19111 defines `CartesianCS` as a direct sub-type of `CoordinateSystem`. ISO also defines `ImageCS` as the union of `AffineCS` and `CartesianCS`, for use by `ImageCRS`. Because the union construct found in some languages like C/C++ does not exist in Java, GeoAPI defines `CartesianCS` as a sub-type of `AffineCS` in order to achieve the same type safety; also, GeoAPI does not define `ImageCS` but uses `AffineCS` instead. In this hierarchy, `CartesianCS` is considered a special case of `AffineCS` where all axes are perpendicular to each other.

Union in Ellipsoid interface

ISO 19111 defines the union named `secondDefiningParameter` as being either `semiMinorAxis` or `inverseFlattening`. The union construct (defined in some languages like C/C++) does not exist in Java. GeoAPI changed the interface to require both ellipsoidal parameters (in addition to the `semiMajorAxis` parameter which is mandatory in any case), as was done in OGC 01-009. However, implementors could readily permit users to only provide one of the two parameters by creating a class which calculates the second parameter from the first. For precision, GeoAPI imports the `isIvfDefinitive` attribute from OGC 01-009 to enable the user to establish which of the two parameters was used to define the instance.

Position union

ISO 19107 defines `Position` as a union of `DirectPosition` and `Point` but unions are not allowed in Java. GeoAPI defines `Position` as the base interface of both types so the two conditional accessor methods, `getPoint()` and `getDirectPosition()`, can be replaced by an `instanceof` check. However, the `getDirectPosition()` has been retained with different semantics, conceptually returning a `DirectPosition` at the same location. The conditionality has also been changed to mandatory since all three types conceptually have a well defined location.

Obligation.FORBIDDEN

ISO specifications sometime override a parent method with a comment saying that the method is not allowed for a particular class. Since there is no construct in Java for expressing this constraint in the method signature, GeoAPI defines a `FORBIDDEN` obligation (not in original ISO specifications) to be used with the `@UML` annotation and which adds a flag in the Java documentation.

E.2 Departures due to historical reasons

ReferenceSystem

This interface was initially derived from an ISO 19111 specification published in 2003. Later revisions (in 2005) rely on an interface defined in ISO 19115 instead. The annotations were updated accordingly, but this interface is still defined in the referencing package instead of the metadata package for this historical reason.

ReferenceSystem.getDomainOfValidity()

This method has been kept conformant with the specification published in 2003. Later revisions changed the multiplicity, so the return type should now be a collection. The singleton has been preserved in GeoAPI for historical reasons, and also because the `Extent` attributes already allow collections.

Method getScope() in ReferenceSystem, Datum **and** CoordinateOperation interfaces

This method has been kept conformant with the specification published in 2003. The revision published in 2007 replaced the singleton by a collection and changed the obligation from "optional" to "mandatory", requiring a return value of "not known" if the scope is unknown. This change is still under review.

In the particular case of `ReferenceSystem`, a later revision moved this attribute to subclasses, but GeoAPI keeps this method here for historical reasons.

GeocentricCRS **and** GeographicCRS

Those interfaces are kept conformant with the specification published in 2003. The 2007 revision of ISO 19111 removed the `GeographicCRS` and `GeocentricCRS` types, handling both using the `GeodeticCRS` parent type. GeoAPI keeps them since the distinction between those two types is in wide use.

AxisDirection.FUTURE **and** PAST

Those codes were defined in an older specification (2003) and removed in more recent edition (2007), but has been kept in GeoAPI.

CSFactory **and** CSAuthorityFactory

Added for consistency with CRS and datum factories. This CS factory was not defined in the OGC specification because OGC 01-009 was created before ISO 19111 and had no equivalent of the ISO Coordinate System types.

E.3 Departures for harmonization between the different specifications

Package org.opengis.metadata

Omitted the reference system package, since it duplicates ISO 19111 / OGC Topic 2. This follows the lead of ISO 19111, which states:

> "Normative reference to ISO 19115 is restricted as follows: in this international standard, normative reference to ISO 19111 excludes the `MD_CRS` class and its components classes." *(ISO 19111:2007, section 3 "Normative References")*

IdentifiedObject

ISO 19111 defines an `IdentifiedObjectBase` interface. The latter is omitted in GeoAPI because the split between `IdentifiedObject` and `IdentifiedObjectBase` in the ISO/OGC

specification was a workaround for introducing `IdentifiedObject` in ISO 19111 without changing the `ReferenceSystem` definition in ISO 19115 but GeoAPI does not need this workaround.

Package org.opengis.parameter

Moved the GeneralParameterDescriptor, ParameterDescriptor, ParameterDescriptorGroup, GeneralParameterValue, ParameterValue, ParameterValueGroup, InvalidParameterNameException, InvalidParameterTypeException and InvalidParameterValueException interfaces from `org.opengis.referencing.operation` to `org.opengis.parameter`. With this move, GeoAPI has extended the use of these parameter classes to a more general use rather than only for referencing operation types.

Factory

This interface is not part of the OGC specification. It is added for uniformity, in order to provide a common base class for all factories.

ObjectFactory

This interface is not part of any OGC specification. It is added for uniformity, in order to provide a common base class for all referencing factories producing `IdentifiedObject` instances.

E.4 Departures for closer integration with the Java environment

CodeList.name(), ordinal(), family() **and** valueOf(…)

Provided by analogy with the methods in the JSE 5 `Enum` class. The `family()` method is a special case provided by analogy with `Enum.family()`, which was defined in a initial draft of JSE 5 before the final release.

Matrix.getNumRow(), getNumCol(), getElement() **and** setElement()

Needed for making the matrix useable. The method signature matches the one of `GMatrix` in the `vecmath` package, for straightforward implementation.

VerticalExtent.getVerticalCRS()

ISO 19115 specifies a generic `CoordinateReferenceSystem` instead of the more restrictive `VerticalCRS`. GeoAPI uses the more specific type for type-safety and consistency with `TemporalExtent` usage. However this restriction prevents usage of Height above the ellipsoid when only the constants defined in the `VerticalDatumType` code list are used. If

such height is wanted, implementors need to extend the above code list with their own `ELLIPSOIDAL` constant.

DerivedCRS

ISO 19111 defines a `DerivedCRSType` code list. The latter is omitted in GeoAPI since Java expressions like `(baseCRS instanceof FooCRS)` provides the same capability with more flexibility.

MathTransform2D

This interface is not part of OGC specification. It has been added in GeoAPI for close integration with the Java2D library. The API defined in this interface matches the `java.awt.geom.AffineTransform` API.

E.5 Changes of name without change in functionality

GeographicExtent.getInclusion()

The ISO identifier is "`extentTypeCode`" and defines the value 1 for inclusion, and 0 for exclusion. GeoAPI uses a name which better expresses the meaning of the return value.

GeneralDerivedCRS.getConversionFromBase()

"`conversion`" may be confusing as a method name since it does not indicate which CRS is the source or which is the target. OGC document 01-009 used the `toBase()` method name. By analogy with 01-009, GeoAPI defines a method name which contains the "`FromBase`" expression.

GeneralParameterDescriptor, ParameterDescriptor **and** ParameterDescriptorGroup

GeoAPI uses a name which contains the "`Descriptor`" word for consistency with other libraries in Java (e.g. `ParameterListDescriptor` in *Java Advanced Imaging*).

ParameterValueGroup.getDescriptor()

The ISO name was "`group`". GeoAPI uses "`descriptor`" instead in order to override the `getDescriptor()` generic method provided in the parent interface. In addition the "`descriptor`" name makes more apparent that this method returns an abstract definition of parameters - not their actual values - and is consistent with usage in other Java libraries like the *Java Advanced Imaging* library.

ParameterValue.doubleValue()

Renamed the method from "`value`" to "`doubleValue`" for consistency with `Number.doubleValue()` and the other "`*Value`" methods defined in this interface.

ParameterValue.doubleValueList()

Renamed the method from "`valueList`" to "`doubleValueList`" both for consistency with `doubleValue()` and also because, like `doubleValue()`, this method returns an array of `double` primitives rather than a `Measure` object.

ParameterValue.intValue()

Renamed the method from "`integerValue`" to "`intValue`" for consistency with `Number.intValue()` and the `int` Java primitive type.

ParameterValue.intValueList()

Renamed the attribute from "`integerValueList`" to "`intValueList`" for consistency with `intValue()`.

E.6 Generalizations due to relaxation of ISO/OGC restrictions

GeneralParameterDescriptor.getMaximumOccurs()

Moved up (in the interface hierarchy) the `maximumOccurs` method from `ParameterDescriptorGroup` into this super-interface, for parallelism with the `minimumOccurs` method.

GenericName.head()

ISO defines this method in `ScopedName` only. GeoAPI defines it in the base class since `LocalName` can return a sensible value for it. This reduces the need for casts.

CoordinateReferenceSystem.getCoordinateSystem() method

ISO 19111 defines this method for `SingleCRS` only. GeoAPI declares this method in this parent interface for user convenience, since CS dimension and axes are commonly requested information and will always be available, directly or indirectly, even for `CompoundCRS`.

CompoundCRS.getComponents()

According ISO 19111, "*A Compound CRS is a coordinate reference system that combines two or more coordinate reference systems, none of which can itself be compound*". However this constraint greatly increases the cost of extracting metadata (especially the CRS identifier) of the three-dimensional part of a spatio-temporal CRS. Note also that in "*Coordinate Transformation Services*" (OGC document 01-009), a compound CRS was specified as a pair of arbitrary CRS ("head" and "tail") where each could be another compound CRS, allowing the creation of a tree. GeoAPI follows that more general strategy.

Record.getAttributes()

Figure 15 in ISO 19103:2005 specifies a cardinality of 1. However, this seems to contradict the semantics of the `locate(name)` and `RecordType.getMemberTypes()` methods.

AuthorityFactory.createObject(…)

This method is not part of the OGC specification. It has been added to leverage the capability of factories that can automatically determine the type of the requested object at runtime.

E.7 Addition of elements not in the ISO/OGC specifications

CodeList.identifier()

Defined because each `CodeList` has a UML identifier in addition of the Java programmatic name.

CodeList.names()

Defined because each `CodeList` has at least two names, the Java programmatic name and the UML identifier, while some subclasses have additional names.

CodeList.Filter

The inner `CodeList.Filter` interface is not part of the OGC specification. It has been added because `CodeList` is one of the few concrete classes in GeoAPI and there is a need to give some user control over the behavior of the `CodeList` implementation.

ParameterDescriptor.getValidValues() **and** getUnit()

Those methods are not part of ISO specification. They are provided as a complement of information.

GeneralParameterDescriptor.createValue(…) **and**
ParameterDescriptorGroup.createValue(…)

Those methods are not part of the ISO specification. They are provided in GeoAPI as a kind of factory methods.

CoordinateOperationFactory.createOperation(…)

This method has been added at user request, in order to specify the desired transformation path when many are available.

CoordinateOperationFactory.createConcatenatedOperation(…)

This method has been added because OGC 01-009 does not define a factory method for creating such object.

CoordinateOperationAuthorityFactory.createOperationMethod(…)

This method has been added because OGC 01-009 does not define a factory method for creating such object.

AuthorityFactory.getAuthorityCodes()

This method is not part of the OGC specification but has been added as a way to publish the capabilities of a factory.

MathTransformFactory.getAvailableMethods()

This method is not part of the OGC specification. It has been added as a way to publish the capabilities of a factory.

MathTransformFactory.getLastMethodUsed()

This method is not part of the OGC specification. It has been added because this information appears to be needed in practice. A more object-oriented approach would have been to return a `{MathTransform, OperationMethod}` tuple in the `createParameterizedTransform(…)` method, but we wanted to keep the latter unchanged for historical reasons (it is inherited from OGC 01-009) and because only a minority of use cases need the operation method.

Note that the existence of this method does not break thread-safety if the implementor stores this information in a `ThreadLocal` variable.

MathTransformFactory.getDefaultParameters(…) **and** createBaseToDerived(…)

Those methods are part of the GeoAPI mechanism for defining the math transform parameters or deriving other transforms.

MathTransform1D

This interface is not part of the OGC specification. It has been added as a complement of `MathTransform2D` and because the 1D case provides opportunities for optimization through a transform method accepting a single double primitive type.

Projection, ConicProjection, CylindricalProjection, PlanarProjection,

Those interfaces are not part of the ISO specification. They have been added in GeoAPI at user request, in order to provide a way to know the kind of map projection.

NameFactory

Added in order to provide constructors for `GenericName` and related interfaces.

InternationalString

Added this new type in order to distinguish between localizable and non-localizable character strings. Not all character strings should be localizable; for example *Well Know Text* or code names should probably be language neutral. Since the ISO/OGC UML does not say which character strings are localizable and which ones are not, we have done our own guesses in GeoAPI.

GenericName.toInternationalString()

This method is not part of the ISO specification. It has been added to provide a way to localize the name.

IdentifiedObject.toWKT()

This method is not part of the OGC specification. It has been added in order to provide the converse of the `CRSFactory.createFromWKT(String)` method, which is defined in OGC 01-009.

RecordType.getContainer()

This is the `TypeList` association in figure 15 of ISO 19103:2005, but navigable in the opposite way. The navigation in the ISO way is represented by the `RecordSchema.getDescription().values()`.

FactoryException, InvalidParameterCardinalityException, InvalidParameterTypeException, MismatchedDimensionException, NoSuchAuthorityCodeException, NoSuchIdentifierException, NoninvertibleTransformException, OperationNotFoundException, ParameterNotFoundException, TransformException

Those exceptions are not part of the OGC specification.

E.8 Extensions for convenience, without introduction of new functionality

DirectPosition and Envelope

Those interfaces were moved into the `org.opengis.geometry` package for convenience.

Envelope.getCoordinateReferenceSystem() **and** getDimension()

ISO does not define those methods - the CRS or the dimension can be obtained only through one of the corner `DirectPosition` objects. GeoAPI adds those methods for convenience as a more direct way of obtaining the information and to free the user from the need to choose an arbitrary corner (very defensive code might feel the need to get the value from both corners to check they were the same).

Envelope.getMinimum(), getMaximum(), getMedian **and** getSpan()

Those methods are not part of ISO specification. GeoAPI adds those methods for convenience and efficiency, since some implementations might store the minimum and maximum ordinate values directly in the `Envelope` itself rather than in a contained `DirectPosition` corner.

ScopedName.path()

This method is not part of ISO specification. It has been added in GeoAPI as a complement of the ISO `tail()` method.

GenericName.tip(), toFullyQualifiedName() **and** toString()

Those methods are not part of ISO specification. They do not provide any additional information compared to that accessible though the standard methods defined by ISO, but provide easier to access frequently requested information.

RecordType.getMembers() **and** isInstance(…)

Those methods provide no additional information compared to the ISO standard methods, but are declared in GeoAPI as a convenient shortcut.

Record.set(…)

This method provides no additional functionality compared to the ISO standard methods, but is declared in GeoAPI as a convenient shortcut.

ParameterDescriptorGroup.descriptor(…)
ParameterValueGroup.parameter(…), groups(…) and addGroup(…)

Those methods are not part of the ISO specification. They have been added in an attempt to make the interfaces easier to use.

Matrix.isIdentity()

Added as a convenience for a frequently requested operation.

Annex F

(informative)

Comparison with legacy OGC specifications

The ISO specifications from the 19100 series supersede some OGC specifications. In areas where specifications overlap, the ISO data types were used. However some standards may still refer to the legacy OGC specification data types. For example, the OGC defines the *Well Known Text* format using its own referencing terminology. This annex lists the legacy OGC types retained in GeoAPI together with the ISO replacement when there is one.

F.1 Comparison of OGC 01-009 with ISO 19111

OGC 01-009	ISO 19111 or 19107	GeoAPI
PT_CoordinatePoint	DirectPosition	DirectPosition
PT_Envelope	GM_Envelope	Envelope
PT_Matrix		Matrix
CS_AxisInfo	CS_CoordinateSystemAxis	CoordinateSystemAxis
CS_AxisOrientationEnum	CS_AxisOrientation	AxisOrientation
CS_CompoundCoordinateSystem	SC_CompoundCRS	CompoundCRS
CS_CoordinateSystem	SC_CoordinateReferenceSystem	CoordinateReferenceSystem
CS_CoordinateSystemAuthorityFactory		CRSAuthorityFactory
CS_CoordinateSystemFactory		CRSFactory
CS_Datum	CD_Datum	Datum
CS_DatumType	CD_VerticalDatumType	VerticalDatumType
CS_Ellipsoid	CD_Ellipsoid	Ellipsoid
CS_FittedCoordinateSystem	SC_DerivedCRS	DerivedCRS
CS_GeocentricCoordinateSystem	SC_GeodeticCRS	GeocentricCRS
CS_GeographicCoordinateSystem	SC_GeodeticCRS	GeographicCRS
CS_HorizontalDatum	CD_GeodeticDatum	GeodeticDatum
CS_Info	IO_IdentifiedObject	IdentifiedObject
CS_LocalCoordinateSystem	SC_EngineeringCRS	EngineeringCRS
CS_LocalDatum	CD_EngineeringDatum	EngineeringDatum
CS_PrimeMeridian	CD_PrimeMeridian	PrimeMeridian
CS_ProjectedCoordinateSystem	SC_ProjectedCRS	ProjectedCRS
CS_Projection	CC_Conversion	Projection
CS_ProjectionParameter	CC_ParameterValue	ParameterValue
CS_VerticalCoordinateSystem	SC_VerticalCRS	VerticalCRS
CS_VerticalDatum	CD_VerticalDatum	VerticalDatum
CS_WGS84ConversionInfo		WGS84ConversionInfo
CT_CoordinateTransformation	CC_CoordinateOperation	CoordinateOperation
CT_CoordinateTransformationAuthorityFactory		CoordinateOperationAuthorityFactory
CT_CoordinateTransformationFactory		CoordinateOperationFactory
CT_MathTransform		MathTransform
CT_MathTransformFactory		MathTransformFactory
CT_Parameter	CC_ParameterValue	ParameterValue

Annex G

(informative)

Reference Implementation

The GeoAPI library is released along with a Reference Implementation to demonstrate its viability and ensure that functional client code can be written with the release of this specification.

The "proof of concept" implementation is provided by the Apache Spatial Information System (SIS) project (http://sis.apache.org/) version 0.8 or above. This implementation is free software, licensed to all under the terms of the Apache License, version 2, and therefore open for study, modification and redistribution, the latter under some constraints specified by the Apache license.

Annex H Revision history

Date	Release	Author	Paragraph modified	Description
2009-04-08	3.0.0-Draft	Adrian Custer	All	Initial Public Draft
2009-09-06	3.0.0-Draft-r1	Martin Desruisseaux	Annex	List of departures
2010-02-11	3.0.0-Draft-r2	Martin Desruisseaux	8.1.1, 10.1, annex F	Clarifications
2016-11-07	3.0.1	Martin Desruisseaux	3, 8.1.6, 8.2, annex G	Replaced JSR-275 by JSR-363

Bibliography

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

[2] IEC 60027 (all parts), *Letter symbols to be used in electrical technology*.

[3] ISO 1000, *SI units and recommendations for the use of their multiples and of certain other units*.

[4] ISO 19103, *Geographic information – Conceptual schema language*. 2005.

[5] ISO 19115, *Geographic information – Metadata*. 2003.

[6] ISO 19115, *Geographic information – Metadata / Corrigendum* 1. 2006.

[7] ISO 19103, *Geographic information – Spatial referencing by coordinates*. 2nd Edition, 2007.

[8] Nordgren, *Bryce, Tools from ISO 19103: A GeoAPI Interface Proposal*. USDA Forest Service.

[9] Nordgren, Bryce, *An ISO-19109 Primer (and comparison to the ComplexFeature effort in GeoTools)*. USDA Forest Service.

[10] Nordgren, Bryce, *An ISO19123 Coverage Primer (and GeoAPI/GeoTools integration guide)*. USDA Forest Service.

[11] Nordgren, Bryce, *The ISO TC/211 Image Concept: An integrated review and definition*. USDA Forest Service.

[12] Daly, Martin, ed. OGC 01-009 *Coordinate Transformation Services*. Revision 1.00

[13] Reynolds, Greg, ed. OGC 03-064 GO-1 *Application Objects*. Version 1.0

