GeoAPI changes

TITLE: **GeoAPI alignment with standards**

AUTHORS: **Martin Desruisseaux, Adrian Custer (Geomatys)**
Jody Garnett, Cory Horner,
Graham Davis (Refractions Research)

DATE: **June, 2007**

CATEGORY: **Change Proposal**

Table of Contents

1. Background

This document tracks the changes submitted since GeoAPI 2.0 release. The changes presented in this proposal are intended to bring the GeoAPI interfaces back into close conformance with the updated OGC Abstract Specifications. The last, and therefore current, release of the GeoAPI library, version 2.0, was made on June 7th 2005. That release brought the GeoAPI interfaces into conformance with OGC Abstract Specifications based on the ISO 19000 series of specifications available at that time. Since then, several OGC Abstract Specifications have been updated. Also in this same interval, several minor issues in the GeoAPI interfaces have been identified. It is therefore desirable to update the GeoAPI interfaces.

The proposal presents two sets of modifications to the GeoAPI 2.0 Java interfaces. These modifications would require small updates to the UML diagrams in the OGC GO-1 Implementation Specification document (03-064r10). These changes do not involve any changes to the OGC Abstract Specifications themselves.

If accepted, the changes would lead to two new releases of the GeoAPI interface library, tentatively numbered 2.1 and 2.2.

- GeoAPI 2.1 would include a series of minor modifications that are upward compatible for interface users.
- GeoAPI 2.2 would include modifications that require changes in the source code of interface users.

Note that in both cases, the changes are incompatible for implementors. It is close to impossible to change an interface in the Java language without causing a compatibility break for implementors. In this document, “compatibility” is always to be understood from the point of view of interfaces users.

Known GeoAPI implementors include the GeoTools and JScience open source projects. The former (GeoTools) already applied the proposed changes since our policy is to test the proposals in at least one (ideally two) implementations before to bring them to OGC. The later (JScience) implements only the “coordinate reference systems” part, which is probably the most stable package in GeoAPI.

1.1 Scope of this document

This document presents the changes aimed to bring GeoAPI into conformance with latest specifications. Other changes, like GeoAPI extensions or issues that are specific to the Java language, are presented in a separated document.

2. Contributors

The following organizations (in alphabetical order) have contributed to GeoAPI development through participation of individual employees:

- Axios Engineering
- Geomatys
- GeoSolutions
- Institut de Recherche pour le développement (IRD)
- Leica Geosystems Geospatial Imaging, LLC (LGGI)
- Refractions Research
- The Open Planning Project (TOPP)
- United States Forest Service (USFS)
- University of Applied Sciences Cologne

3. References

[1] GeoAPI 2.0 interfaces: http://geoapi.sourceforge.net/2.0/javadoc

[2] ISO-19103: Conceptual schema language, revisions for 2003 and 2005

[3] ISO-19107: Feature geometry

[4] ISO-19111: Spatial referencing by coordinates, revisions 03-073r1 and 04-046r3

[5] ISO-19115: Geographic information – Metadata, revisions for 2000, 2003 and 2006

[6] OGC 03-064r10: GO-1 Application Objects

Every change proposed in this document has been applied to the “snapshot” javadoc which can be browsed on-line here:

[7] http://geoapi.sourceforge.net/snapshot/javadoc/

Please note that not all interfaces in the above-cited javadoc are covered in this Request For Changes.

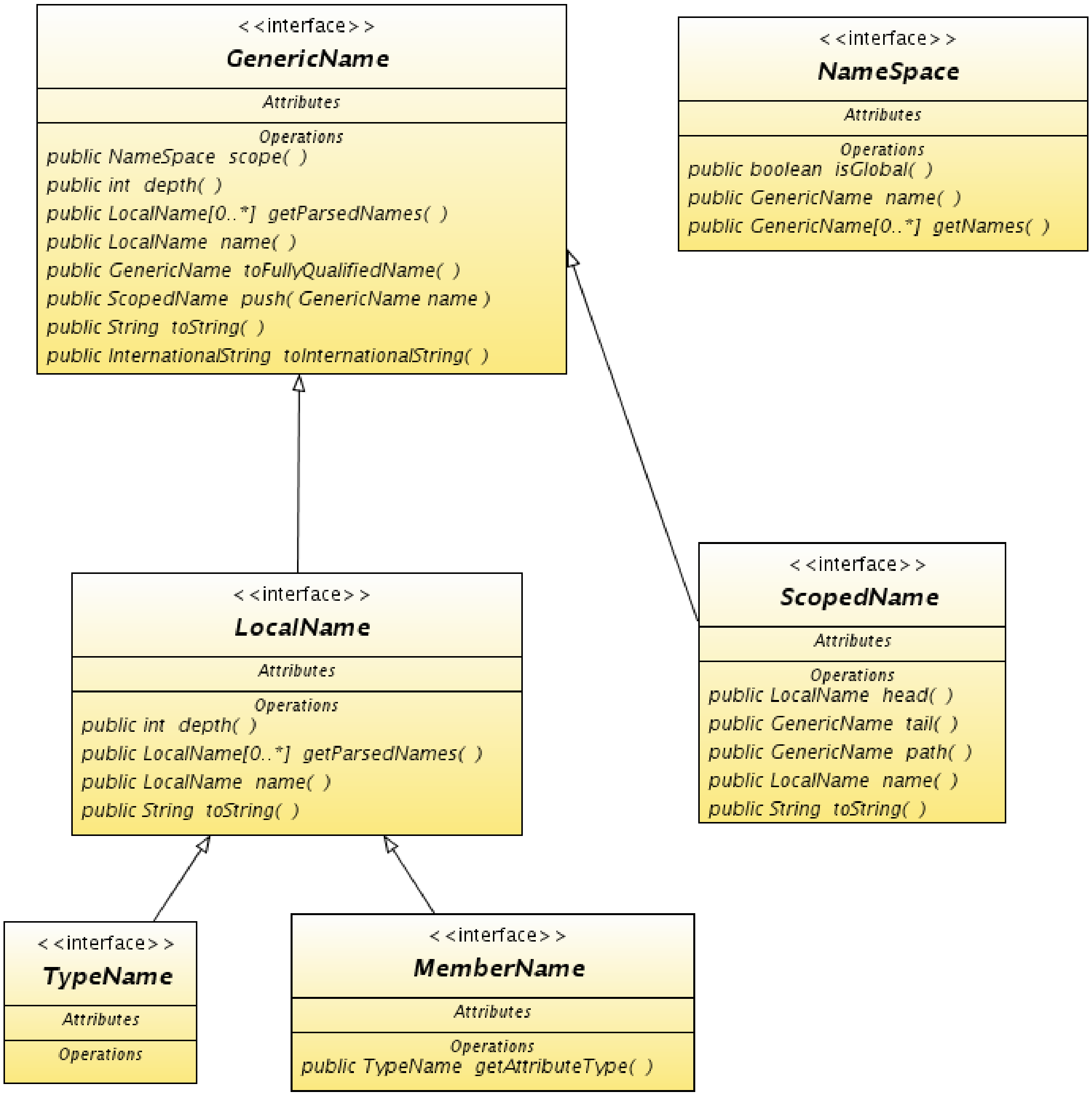
4. Utility

Several of the changes in this package relate to a re-appraisal of the 19103 specification. This work is discussed on the web page:

http://docs.codehaus.org/display/GEOTOOLS/ISO+19123+progress+and+future+plan

which includes a link to the document *ISO 19109 primer* by Bryce Nordgren of the United States Forest Service (USFS) in which he evaluates the ISO 19103 naming scheme.

The diagram below summarizes the name interfaces if 4.1, 4.2 and 4.3 are accepted.


<<interface>>
GenericName
Attributes
Operations
public NameSpace scope( )
public int depth( )
public LocalName[0..*] getParsedNames( )
public LocalName name( )
public GenericName toFullyQualifiedName( )
public ScopedName push(GenericName name)
public String toString( )
public InternationalString toInternationalString( )
<<interface>>
NameSpace
Attributes
Operations
public boolean isGlobal( )
public GenericName name( )
public GenericName[0..*] getNames( )
<<interface>>
LocalName
Attributes
Operations
public int depth( )
public LocalName[0..*] getParsedNames( )
public LocalName name( )
public String toString( )
<<interface>>
ScopedName
Attributes
Operations
public LocalName head( )
public GenericName tail( )
public GenericName path( )
public LocalName name( )
public String toString( )
<<interface>>
TypeName
Attributes
Operations
<<interface>>
MemberName
Attributes
Operations
public TypeName getAttributeType( )

4.1 Add `NameSpace`

Add the `NameSpace` interface from ISO 19103:2005
JIRA task: http://jira.codehaus.org/browse/GEO-71

The `GenericName` interface in GeoAPI 2.0 is confusing. This interface is derived from ISO 19103:2003 figure 13, which has little textual explanation. In GeoAPI 2.0, the scope is wrongly associated to another `GenericName` instance. It should be associated with a `NameSpace` instance instead, but the later interface does not exist in GeoAPI 2.0.

We propose to add a `NameSpace` interface in GeoAPI 2.1/3.0 derived from ISO 19103:2005, associate `GenericName` to this `NameSpace` through a new `scope()` method, and deprecate the current (erroneous) `getScope()` method.

In addition, we propose to add the `head()` and `tail()` methods (derived from ISO 19103) into the `ScopedName` interface.

4.1.1 Affected section(s), table(s), and figure(s)

The changes do not alter the abstract specifications but would cause a change to the GO-1 implementation specification [OGC 03-064r10]: figure 28.

4.1.2 Purposes of the proposed change

Bring GeoAPI and GO-1 more inline with ISO 19103. Clarify what seems to be one of the most confusing areas of GeoAPI 2.0 (according some emails sent on the mailing list).

4.1.3 Reasons for change

Current `GenericName`, `LocalName` and `ScopedName` interfaces are reported to be confusing. The relationship of their members to ISO 19103 is not always obvious, and erroneous in the particular case of the `getScope()` method.

4.1.4 Specific suggested changes

We propose to add the `NameSpace` interface in the `org.opengis.util` package with the following methods derived from ISO 19103:

- `boolean isGlobal();`
- `GenericName name();`
- `Set getNames();`

We would omit methods like `generateID`, `registerID` and `unregisterID` until ISO 19103 becomes more explicit about their purpose.

We propose to deprecate the current `GenericName.getScope()` method (which was used to return another `GenericName`) and add the following method in `GenericName` instead:

- `NameSpace scope();`

We propose to add the following methods in `ScopedName`:

- `LocalName head();`
  Synonymous for the current GeoAPI's `asLocalName()` method, but with naming conformant to ISO 19103.

- `GenericName tail();`
  Replace the `getScope()` method deprecated above, with identical functionality but with naming conformant to ISO 19103.

4.1.5 Consequences of the change

GeoAPI implementors would be required to implement the above-cited interfaces and methods. GeoAPI users would be encouraged to replace their uses of `GenericName.getScope()` by `GenericName.scope()`.

4.1.6 Consequences if not approved

GeoAPI would not conform to the ISO 19103 specification. Some implementors or users might ignore the `GenericName` interface because it doesn't fit their needs.

4.2 Add `TypeName` and `MemberName`

Add the `TypeName` and `MemberName` interfaces from ISO 19103
JIRA task: http://jira.codehaus.org/browse/GEO-71

We propose to add the `TypeName` and `MemberName` interfaces in GeoAPI, which were defined in ISO 19103 but not yet defined in GeoAPI.

4.2.1 Affected section(s), table(s), and figure(s)

In [OGC 03-064r10]: figure 28.

4.2.2 Purposes of the proposed change

Bring GeoAPI and GO-1 more inline with ISO 19103. Some users reported that those interfaces are required for implementing ISO 19103 `Record` and `RecordType`.

4.2.3 Reasons for change

ISO 19103 compliance and support for `Record` and `RecordType` implementations (not part of this section).

4.2.4 Specific suggested changes

We propose to add the following interfaces in the `org.opengis.util` package:

- `TypeName` as a sub-interface of `LocalName`, with no additional method.
- `MemberName` as a sub-interface of `LocalName`, with only the following additional method:
  - `TypeName getAttributeType();`

4.2.5 Consequences of the change

None. This API addition does not break any existing implementations.

4.2.6 Consequences if not approved

GeoAPI would not conform to the ISO 19103 specification. Developers might create their own `TypeName` and `MemberName` interfaces.

4.3 Modify `GenericName`, `LocalName`, and `ScopedName`

Add, rename and replace methods these three interfaces to better match the ISO 19103 specification.
JIRA Task: http://jira.codehaus.org/browse/GEO-71

The interfaces were refactored based on feedback and insights provided by Bryce Nordgren in his document ISO 19109 Primer. This document is available on the following web page:

http://docs.codehaus.org/display/GEOTOOLS/ISO+19123+progress+and+future+plan

These changes complement the changes described above in section (4.1) “Add `NameSpace`”.

4.3.1 Affected section(s), table(s), and figure(s)

In [OGC 03-064r10]: figure 28.

4.3.2 Purposes of the proposed change

The changes aim to better match the ISO 19103 specification.

4.3.3 Reasons for change

The changes were undertaken due to a re-interpretation of the hard to understand and poorly documented ISO 19103 specification.

4.3.4 Specific suggested changes

- Add the method: `int GenericName.depth()` to return the number of scopes in which the `GenericName` exists. This will be 1 for any `LocalName` and 2 or greater for any `ScopedName`. Override the method in `LocalName` to always return 1.
- Rename the method `GenericName.asLocalName()` to `GenericName.name()` and extend the specifications by overriding the method in both `LocalName` and `ScopedName` for dedicated logic.
- Replace the method `GenericName.asScopedName()` by the method `GenericName.toFullyQualifiedName()` with the altered return type.
- Add the method `GenericName.push(GenericName name)` which returns the `GenericName` for which the method is called expanded by the scope of the `GenericName` argument to the method.
- Add the method `ScopedName.path()` to return all the names of the `ScopedName` object except for the last, the `LocalName`.

4.3.5 Consequences of the change

The changes better match the current understanding of the ISO 19103 specification.

4.3.6 Consequences if not approved

If not approved, GeoAPI would follow the older interpretation which would prevent other packages from using the specification as it is currently understood by the community.

4.4 Add `Record`, `RecordSchema`, and `RecordType`

Add the `Record`, `RecordSchema`, and `RecordType` interfaces from ISO 19103:2005
JIRA task: http://jira.codehaus.org/browse/GEO-72

In GeoAPI 2.0, `RecordType` is mapped to `Class` and `Record` is mapped to `Object`. This is fine on the surface, but does not permit the dynamic construction of arbitrary record types at runtime. This proposal fixes this by presenting an implementation of Figure 15 in ISO19103:2005(E). The interfaces "`Schema`", "`RecordSchema`", "`Record`" and "`RecordType`" all encapsulate dictionaries (Maps) which relate some form of `LocalName` to the indicated type. Figure 15 does not clearly indicate that the dictionary functionality should be publicly exposed, so we have included only the explicitly defined `locate()` method.

4.4.1 Affected section(s), table(s), and figure(s)

none.

4.4.2 Purposes of the proposed change

Provides more straightforward GeoAPI to ISO mapping than `Class` and `Object`.

4.4.3 Reasons for change

Some existing API need to reference `Record` and `RecordType` from ISO 19103.

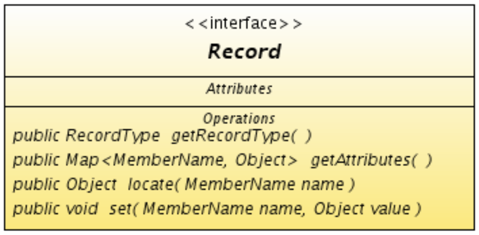
4.4.4 Specific suggested changes

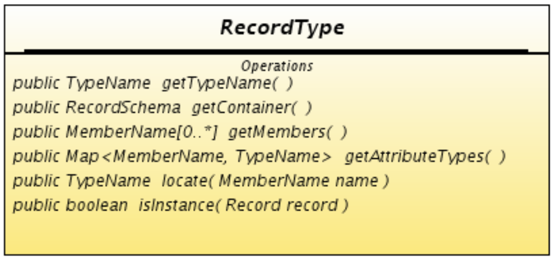
Add the interfaces illustrated in the diagram below. Note that the proposed interfaces contains 3 methods not defined in ISO 19103, listed before the diagram. Those methods expose in a more explicit way some functionalities that would be provided anyway by pure ISO 19103 interfaces, in an attempt to reduce the confusion.

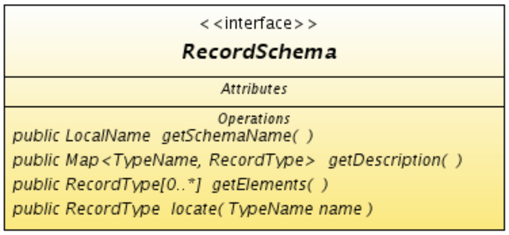
- **`void set(MemberName name, Object value)`**
  Set the value for the attribute of the specified name. This is functionally equivalent to `getAttributes().put(name,value)`. Remind that `name` keys are constrained to `RecordType.getMembers()`.

- **`Set<MemberName> getMembers()`**
  Returns the set of attribute names defined in this `RecordType`'s dictionary. If there are no attributes, this method returns the empty set. This method is functionally equivalent to `getAttributeTypes().keySet()`. The `NameSpace` associated with a `RecordType` contains only members of this `RecordType`. There is no potential for conflict with sub-packages.

- **`boolean isInstance(Record record)`**
  Determines if the specified record is compatible with this record type. This method returns `true` if the specified `record` argument is non-null and the following condition holds: `getMembers().equals(record.getAttributes().keySet())`


<<interface>>
Record
Attributes
Operations
public RecordType getRecordType( )
public Map<MemberName, Object> getAttributes( )
public Object locate(MemberName name)
public void set(MemberName name, Object value)


RecordType
Operations
public TypeName getTypeName( )
public RecordSchema getContainer( )
public MemberName[0..*] getMembers( )
public Map<MemberName, TypeName> getAttributeTypes( )
public TypeName locate(MemberName name)
public boolean isInstance(Record record)


<<interface>>
RecordSchema
Attributes
Operations
public LocalName getSchemaName( )
public Map<TypeName, RecordType> getDescription( )
public RecordType[0..*] getElements( )
public RecordType locate(TypeName name)

4.4.5 Consequences of the change

This is a compatible change.

4.4.6 Consequences if not approved

Some methods in GeoAPI will continue to reference J2SE `Class` and `Object` instead of ISO `RecordType` and `Record`.

4.5 Add `UnlimitedInteger`

Add the `UnlimitedInteger` class from ISO 19103
JIRA task: none

ISO 19103 defines `Decimal` , `Vector` , `Real` , `Number` , `UnlimitedInteger` and `Integer` types. All of them have a Java counterpart as array or various `java.lang.Number` subclasses, except `UnlimitedInteger`. Because there is requests from users for providing a clear mapping between ISO 19103 and GeoAPI, we propose to fill the `UnlimitedInteger` hole by providing this class in GeoAPI.

In order to fit `UnlimitedInteger` in the `java.lang.Number` framework, it must be a concrete class rather than an interface. However this class is small, straightforward and its API clearly defined by its J2SE `Number` parent class.

4.5.1 Affected section(s), table(s), and figure(s)

None. The `UnlimitedInteger` type is not mentioned in the GO-1 specification and doesn't need to, since the GO-1 specification do not cover the full GeoAPI range.

4.5.2 Purposes of the proposed change

Provides a closer “one to one” mapping between ISO 19103 and Java types used by GeoAPI.

4.5.3 Reasons for change

This is a user request on GeoAPI mailing list. Users want a more straightforward ISO – GeoAPI relationship when possible.

4.5.4 Specific suggested changes

Provides the following class in `org.opengis.util` package (`Number` and `Comparable` are standard Java class or interface):

```
public final class UnlimitedInteger extends Number implements Comparable
```

Implement the following methods required by standard Java API:

`intValue()`, `longValue()`, `floatValue()`, `doubleValue()`, `toString()`, `hashCode()`, `equals(Object)`, `compareTo(Object)`.

Declare the following constants, which mirror usage in other standard Java types:

`MIN_VALUE`, `MAX_VALUE`, `NEGATIVE_INFINITY`, `POSITIVE_INFINITY`

Negative and positive infinities are implemented using `Integer.MIN_VALUE` and `Integer.MAX_VALUE` as sentinel values. The usage of `Integer.MAX_VALUE` sentinel value is consistent with standard `java.util.Collection.size()` specification, which already use this sentinel value for everything greater than 2^{31}-1.

4.5.5 Consequences of the change

This change does not cause any compatibility breaks. It would bring one more class into GeoAPI which is expected to be used progressively in the future.

4.5.6 Consequences if not approved

A hole would continue to exist in the ISO 19103 to GeoAPI mapping. Interfaces that need to represent an unlimited integer may develop their own workaround.

5. Metadata

5.1 Add new ISO 19115:2003 interfaces and code lists

The GeoAPI 2.0 metadata package was derived from the ISO 19115:2000 specification published on the OGC website as OGC 01-111. New interfaces and code lists were added in ISO 19115:2003. We propose to add them as defined in ISO 19115:2003 with the modifications published in Errata for the specification ISO 19115:2003 / Cor.1:2006(E).

5.1.1 Affected section(s), table(s), and figure(s)

None, since metadata interfaces are not documented in the GO-1 specification. The GeoAPI metadata package is a straight translation from ISO 19115 UML into Java interfaces, therefore documenting them in GO-1 would duplicate ISO 19115.

5.1.2 Purposes of the proposed change

Bring GeoAPI inline with the latest ISO 19115 specification.

5.1.3 Reasons for change

GeoAPI 2.0 is derived from a slightly outdated ISO 19115 specification.

5.1.4 Specific suggested changes

All changes below are to be applied in the `org.opengis.metadata` package.

- Add code list `identification.AssociationType`
  from ISO 19115 `DS_AssociationTypeCode`
  JIRA task: http://jira.codehaus.org/browse/GEO-100
- Add code list `identification.InitiativeType`
  from ISO 19115 `DS_InitiativeTypeCode`
  JIRA task: http://jira.codehaus.org/browse/GEO-101
- Add interface `identification.RepresentativeFraction`
  from ISO 19115 `DS_RepresentiveFraction`
  JIRA task: http://jira.codehaus.org/browse/GEO-93

5.1.5 Consequences of the change

This change does not cause any compatibility breaks, since they are only additions without any change in the existing interfaces.

5.1.6 Consequences if not approved

GeoAPI would not conform to the ISO 19115:2003 specification.

5.2 Rename misspelled interfaces

Add `FormatConsistency` and `NonQuantitativeAttributeAccuracy` interfaces.
Deprecate `FormalConsistency` and `NonQuantitativeAttributeCorrectness`
JIRA task: http://jira.codehaus.org/browse/GEO-93

Some interfaces were either misspelled in GeoAPI 2.0, or their name changed as a result of ISO 19115 change.

5.2.1 Affected section(s), table(s), and figure(s)

None, since metadata interfaces are not documented in the GO-1 specification. The GeoAPI metadata package is a straight translation from ISO 19115 UML into Java interfaces, therefore documenting them in GO-1 would duplicate ISO 19115.

5.2.2 Purposes of the proposed change

Bring GeoAPI inline with latest ISO 19115 specification.

5.2.3 Reasons for change

GeoAPI 2.0 is derived from a slightly outdated ISO 19115 specification.

5.2.4 Specific suggested changes

- In `org.opengis.metadata.quality` package, create a `FormatConsistency` interface with the same content than the old `FormalConsistency`. Deprecate the later and set it as a sub-interface of the new interface in order to ease transition.
- In the `org.opengis.metadata.quality` package, create a `NonQuantitativeAttributeAccuracy` interface with the same content as the old `NonQuantitativeAttributeCorrectness`. Deprecate the latter and set it as a sub-interface of the new interface in order to ease transition.

5.2.5 Consequences of the change

This change does not cause immediate compatibility breakage since the interfaces to be renamed are leafs and are are not directly referenced by any other GeoAPI method. Users of those interfaces will need to update their code, but they can postpone this task for as long as the deprecated interfaces are not removed.

5.2.6 Consequences if not approved

GeoAPI would not conform to the ISO 19115:2003 specification.

5.3 Add `CharacterSet`

Add final class `metadata.identification.CharacterSet` from ISO19115:2003
JIRA task: http://jira.codehaus.org/browse/GEO-105

GeoAPI 2.0 mapped ISO 19115 `MD_CharacterSetCode` to `java.nio.charset.Charset` in an effort to leverage standard Java API as much as possible. GeoAPI users reported that it was a cause of difficulties and asked for a direct mapping from ISO 19115 to GeoAPI.

5.3.1 Affected section(s), table(s), and figure(s)

None, since neither `CharacterSet` nor `Charset` are mentioned in GO-1 specification.

5.3.2 Purposes of the proposed change

Bring GeoAPI inline with latest ISO 19115 specification.

5.3.3 Reasons for change

This is a GeoAPI user request which aims to give a more flexible and more straightforward ISO – GeoAPI mapping.

5.3.4 Specific suggested changes

- Add a `CharacterSet` code list as defined in ISO 19115:2003 `MD_CharacterSetCode`.
- Add the following method:
  - `DataIdentification.getCharacterSets();`
- Deprecate the following method:
  - `DataIdentification.getCharacterSet();`
- Change the return type from `java.nio.charset.Charset` to `CharacterSet` for the following methods:
  - `MetaData.getCharacterSet();`

5.3.5 Consequences of the change

These are compatible changes, except for the last one (the return type change) which would be performed for the GeoAPI 3.0 release only. Users will need to update their code, but legacy code will continue to work until we remove the deprecated methods.

5.3.6 Consequences if not approved

The matching between ISO 19115 and GeoAPI would be less straightforward.

5.4 Match Cardinality of Updated Specification

> **The updated ISO 19115:2003 changed the cardinality of certain return types. GeoAPI now returns `Collection <? extends Type>` rather than `Type`.**
>
> **JIRA task: http://jira.codehaus.org/browse/GEO-93**

The 2003 version of the ISO 19115 specification modified certain return types to return several objects rather than a single object. The update to GeoAPI proposes to modify the return type from a single object to a `java.util.Collection` parametrized as `<? extends "Type">` which will allow implementors to narrow both the Collection type to any subclass of `Collection` and the parameteric type to any subclass of the “`Type`”. This modification does mean that the returned collection has to be considered read-only by users based on the nature of Java generics.

5.4.1 Affected section(s), table(s), and figure(s)

These changes should have no effect on any specification documents.

5.4.2 Purposes of the proposed change

The changes aim to bring GeoAPI in line with the ISO 19115:2003 specification.

5.4.3 Reasons for change

The ISO specification changed the cardinality of several elements between the 2000 and 2003 revisions.

5.4.4 Specific suggested changes

Deprecate:
```
   String metadata.citation.Telephone.getVoice()
```
and add:
```
  Collection <String>
    metadata.identification.Telephone.getVoices()
```

Deprecate:
```
   String metadata.citation.Telephone.getFacsimile()
```
and add:
```
  Collection <String>
    metadata.identification.Telephone.getFacsimiles()
```

```
metadata.constraint.LegalConstraints.getOtherConstraints()
       returns       Collection<? extends InternationalString>
       not           InternationalString
```

Deprecate:

```
  RangeDimension metadata.content.CoverageDescription.getDimension()
```

and add:

```
  Collection <? extends RangeDimension>
    metadata.content.CoverageDescription.getDimensions()
```

Deprecate:

```
   Charset metadata.identification.DataIdentification.getCharacterSet()
```

and add:

```
  Collection <CharacterSet>
    metadata.identification.DataIdentification.getCharacterSets()
```

Deprecate:

```
   ScopeCode metadata.maintenance.MaintenanceInformation.getUpdateScope()
```

and add:

```
  Collection <ScopeCode>
    metadata.maintenance.MaintenanceInformation.getUpdateScopes()
```

Deprecate:

```
   metadata.maintenance.MaintenanceInformation.getUpdateScopeDescription()
```

and add:

```
  Collection <? extends ScopeDescription>
    metadata.maintenance.MaintenanceInformation.getUpdateScopeDescriptions()
```

Deprecate:

```
 InternationalString
   metadata.maintenance.MaintenanceInformation.getMaintenanceNote()
```

and add:

```
  Collection <? extends InternationalString>
    metadata.maintenance.MaintenanceInformation.getMaintenanceNotes()
```

Deprecate:

```
   ResponsibleParty metadata.MetaData.getContact()
```

and add:

```
  Collection <? extends ResponsibleParty>
    metadata.MetaData.getContacts()
```

Deprecate:

```
   Date[] metadata.quality.Element.getDate()
```

and add:

```
  Collection <Date>
    metadata.quality.Element.getDates()
```

Deprecate:

```
   Result metadata.quality.Element.getResult()
```

and add:

```
  Collection <? extends ResponsibleParty>
    metadata.quality.Element.getResults()
```

5.4.5 Consequences of the change

This is a compatible change except for `LegalConstraints.getOtherConstraints()`. Users will need to update their code but they can wait until the deprecated methods are removed.

5.4.6 Consequences if not approved

GeoAPI would not conform to the ISO 19115:2003 specification.

5.5 Return New Types from ISO 19115:2003

The 2003 update to the ISO 19115 specification introduced several new types which are useful as return values from various methods. We propose updating several methods to use these new return types.

JIRA task: http://jira.codehaus.org/browse/GEO-93

For some methods defined in GeoAPI 2.0, they interfaces matching the type defined in ISO specification were not available. Now that GeoAPI 2.1/3.0 defines some missing interfaces, the return type of some existing methods can be updated in order to return the same type than the one defined by ISO. The type changes are summarized below:

`java.lang.Class`	to	`org.opengis.util.RecordType`
`org.opengis.util.LocalName`	to	`org.opengis.util.MemberName`
`java.util.Date`	to	`org.opengis.temporal.TemporalPrimitive`
`long` or `double` primitive	to	`org.opengis.metadata.identification.RepresentativeFraction`
`long` primitive	to	`org.opengis.temporal.PeriodDuration`
`java.nio.charset.Charset`	to	`org.opengis.metadata.identification.CharacterSet`
`java.lang.Object`	to	`org.opengis.util.Record`

5.5.1 Affected section(s), table(s), and figure(s)

None.

5.5.2 Purposes of the proposed change

Bring GeoAPI inline with ISO specification.

5.5.3 Reasons for change

Interfaces not available in GeoAPI 2.0 will be available in GeoAPI 3.0.

5.5.4 Specific suggested changes

`metadata.content.CoverageDescription.getAttributeDescription()`
returns `RecordType` not `Class`.

`metadata.content.RangeDimension.getSequenceIdentifier()`
returns `MemberName` not `LocalName`.

`metadata.extent.TemporalExtent.getExtent()`
returns `TemporalPrimitive` not `Date`.

`metadata.identification.Resolution.getEquivalentScale()`
returns `RepresentativeFraction` not `double`.

`metadata.lineage.Source.getScaleDenominator()`
returns `RepresentativeFraction` not `long`.

`metadata.maintenance.MaintenanceInformation.getUserDefinedMaintenanceFrequency`
returns `PeriodDuration` not `long`.

`metadata.Metadata.getCharacterSet()`
returns `CharacterSet` not `Charset`.

`metadata.quality.QuantitativeResult.getValueType()`
returns `RecordType not Class.`

Deprecate `Object metatdata.spatial.Georeferenceable.getParameters()`
for `Record metatadata.spatial.Georeferenceable.getGeoreferencedParameters()`

5.5.5 Consequences of the change

Users will need to update their code.

5.5.6 Consequences if not approved

GeoAPI would not conform with the ISO 19115 specification.

5.6 Deprecated methods dropped from specification

> **The 2003 version of the ISO 19115 specification drops a number of methods from the standard. We propose deprecating these methods for the next release so that the methods can be removed during the subsequent release cycle.**
>
> **JIRA task: http://jira.codehaus.org/browse/GEO-93**

ISO19115, version 2003, no longer includes the method `getFeatureCatalogueSupplement()` as part of the `ApplicationSchemaInformation` interface as shown in Figure A.14. The method `getIdentifierTypes()` in the interface `metadata.citation.Citation` has been dropped from ISO 19115:2003 figure A.16. We propose deprecating the methods for this release and removing it from the next release.

5.6.1 Affected section(s), table(s), and figure(s)

The change will not modify any specification but will bring GeoAPI back into conformance with the current ISO 19115 specification.

5.6.2 Purposes of the proposed change

The deprecations would ensure GeoAPI could come back into compliance with the updated ISO specification in a future release while honoring the "deprecate then remove" commitment to project users.

5.6.3 Reasons for change

The ISO specification has been updated with the removal of some methods.

5.6.4 Specific suggested changes

Deprecate the following methods:

- **`ApplicationSchemaInformation.getFeatureCatalogueSupplement()`**
- `citation.`**`Citation.getIdentifierTypes()`**
- `extent.`**`VerticalExtent.getUnit()`**
- `identification.`**`DataIdentification.getGeographicBox()`**
- `identification.`**`DataIdentification.getGeographicDescription()`**

5.6.5 Consequences of the change

This is a compatible change. The deprecation would begin the process of addressing an update to the specification in which a previously existing method was removed. The change will enable implementors and users to avoid those methods or to alter existing code ensuring their code will still work with a future version of GeoAPI in which this method has been dropped.

5.6.6 Consequences if not approved

If the deprecation proposal is not accepted, GeoAPI will have no graceful way to transition towards the updated ISO specification. GeoAPI could not remove the methods in version 2.2 or 3.1 and would therefore remain behind latest ISO specification.

5.7 Add new methods

Add methods that are new in ISO 19115:2003 compared to ISO 19115:2000
JIRA task: none.

Due to an update in the 2003 version of the ISO 19115 specification, some new methods need to be added to various interfaces. For example the specification added a method returning zero or one instances of name, a `CharacterString`, as seen in figure A.16 of ISO19115:2003. In order to implement this update, we propose adding a method `getName()` which returns either null or a String with the name if there is one.

5.7.1 Affected section(s), table(s), and figure(s)

The additions will not affect any specification document.

5.7.2 Purposes of the proposed change

The change would bring GeoAPI into conformance with ISO 19115:2003.

5.7.3 Reasons for change

The ISO 19115 specification was updated.

5.7.4 Specific suggested changes

In `metadata.citation.OnLineResource`, add method:
- `String getName()`

In `metadata.identification.Identification`, add method:
- `Collection<? extends AggregateInformation> getAggregationInfo()`

In `metadata.maintenance.MaintenanceInformation`, add methods:
- `Collection <? extends Responsibile Party> getContacts()`

In `metadata.maintenance.ScopeDescription`, add methods:

- `public Set<? extends AttributeType> getAttributeInstances()`
- `public String getDataset()`
- `public String getOther()`

In `metadata.MetaData`, add method:

- `String getDataSetUri()`

5.8 Expand the Role and Scope code lists

> **Some subclasses of `CodeList` should be modified to add a new list element required by the updated ISO specification.**
> **JIRA task: http://jira.codehaus.org/browse/GEO-93**

The 2003 version of the ISO 19115 specification includes an `AUTHOR` element in the `metatdata.citation.Role` code list as seen in figure A.16. We propose expanding the size of the `Role` `CodeList` by one element and adding the `AUTHOR` element to the class.
The `CodeList` subclass `metadata.maintenance.Scope` gained an extra element in the 2003 version of the ISO 19115 specification. The “`TILE`” elements needs to be added.

5.8.1 Affected section(s), table(s), and figure(s)

The change will have no effect on any specification document.

5.8.2 Purposes of the proposed change

The expanded `CodeList` would be fully compliant with the updated specification.

5.8.3 Reasons for change

The `CodeList` element has been added in the current version of the ISO19115 specification.

5.8.4 Specific suggested changes

- Add the `AUTHOR` field to the `Role` code list.
- Add the `TILE` field to the `ScopeCode` code list.

5.8.5 Consequences of the change

The changes allow conformance with the updated specification.

5.8.6 Consequences if not approved

Failure to accept this request would mean that GeoAPI continue missing one of the `Role` and `ScopeCode` codes in the current specification.

5.9 Modify `VerticalExtent`

> **Due to a change in the ISO 19115 standard, in the interface `metadata.extent.VerticalExtent`, the method `getVerticalDatum()` has been replaced by `getVerticalCRS()`.**
> **JIRA task: http://jira.codehaus.org/browse/GEO-93**

To accommodate a change in the ISO standard, we propose deprecating the existing method

```
VerticalDatum  metadata.extent.VerticalExtent.getVerticalDatum()
```

and adding the new method

```
VerticalCRS  metadata.extent.VerticalExtent.getVerticalCRS()
```

so as to comply with the new standard.

5.9.1 Affected section(s), table(s), and figure(s)

This change does not alter any standards.

5.9.2 Purposes of the proposed change

The change would bring GeoAPI into compliance with the new ISO standard.

5.9.3 Reasons for change

The standard was changed.

5.9.4 Specific suggested changes

Deprecate

```
VerticalDatum  metadata.extent.VerticalExtent.getVerticalDatum()
Unit  metadata.extent.VerticalExtent.getUnit()
```

in favour of the new

```
  VerticalCRS  metadata.extent.VerticalExtent.getVerticalCRS()
```

5.9.5 Consequences of the change

GeoAPI implementors would have to add the method and GeoAPI users would get one release cycle to update their code.

5.9.6 Consequences if not approved

GeoAPI would not conform to the new standard.

5.10 Fix various constant values

> **Several constant values should be altered due to changes in the ISO standard or to errors in earlier implementations.**
>
> **JIRA task:** http://jira.codehaus.org/browse/GEO-96

GeoAPI defines a number of constant values. We propose changing some of these due to several different reasons. Some constants were incorrectly defined in GeoAPI 2.0, some constants have moved in the current standard.

5.10.1 Affected section(s), table(s), and figure(s)

These changes do not affect any of the standard documents.

5.10.2 Purposes of the proposed change

To match the current version of the standard and to fix earlier errors.

5.10.3 Reasons for change

Because the standards have been updated and errors have been found.

5.10.4 Specific suggested changes

Due to a spelling error, deprecate

```
metadata.identification.TopicCategory.HEALT
```

but add

```
metadata.identification.TopicCategory.HEALTH
```

Due to an update to the ISO specification, deprecate

```
metadata.Identifier.VERSION_KEY
```

which has been replaced by

```
referencing.ReferenceIdentifier.VERSION_KEY
```

(see the Referencing section, Add interface `ReferenceIdentifier`)

Deprecate

```
metadata.spatial.GeometricObjectType.COMPLEXES
metadata.spatial.GeometricObjectType.COMPOSITES
```

but add

```
metadata.spatial.GeometricObjectType.COMPLEX
metadata.spatial.GeometricObjectType.COMPOSITE
```

5.10.5 Consequences of the change

Closer conformance to ISO 19115:2003.

5.10.6 Consequences if not approved

GeoAPI would not conform to the ISO 19115:2003 specification.

5.11 Update UML annotation for `ServiceIdentification`

Update the UML annotation for the interface `ServiceIdentification`.
JIRA task: http://jira.codehaus.org/browse/GEO-97

The UML annotation for the interface `metadata.identification.ServiceIdentification` was changed by an errata document “Errata for the spec ISO 19115:2003 / Cor.1:2006(E)” which altered the UML name of the class from `MD_ServiceIdentification` to `SV_ServiceIdentification`.

5.11.1 Affected section(s), table(s), and figure(s)

This change should have no effect on any specification document.

5.11.2 Purposes of the proposed change

The changes bring the annotation for this interface into conformance with the amended specification.

5.11.3 Reasons for change

The specification was amended.

5.11.4 Specific suggested changes

The `@UML` annotation for the interface `metadata.identification.ServiceIdentification` will be altered from

```
@UML(identifier="MD_ServiceIdentification", specification=ISO_19115)
```

to

```
@UML(identifier="SV_ServiceIdentification", specification=ISO_19115)
```

5.11.5 Consequences of the change

The changes match the amended specification.

5.11.6 Consequences if not approved

Without changes, the annotation would be incorrect for the current, amended specification but instead would match the outdated, specification IS0 19115:2003.

6. Geometry (Spatial Schema)

> NOTE: *Because the first proposed change in the "GeoAPI specifics changes" (5.1) affects the names of all the objects in the package and the second proposed change (5.2) alters the name of a package used in several places, this document adopts a naming scheme as if both of those proposals had been accepted.*

6.1 Add missing geometry interfaces

Add ISO 19107 interfaces omitted from GeoAPI 2.0 because of time constraint.
JIRA task: http://jira.codehaus.org/browse/GEO-1

Add the following interfaces derived from ISO 19107:

- `geometry.aggregate.MultiCurve`
- `geometry.aggregate.MultiSurface`
- `geometry.complex.CompositePoint`

6.1.1 Affected section(s), table(s), and figure(s)

none.

6.1.2 Purposes of the proposed change

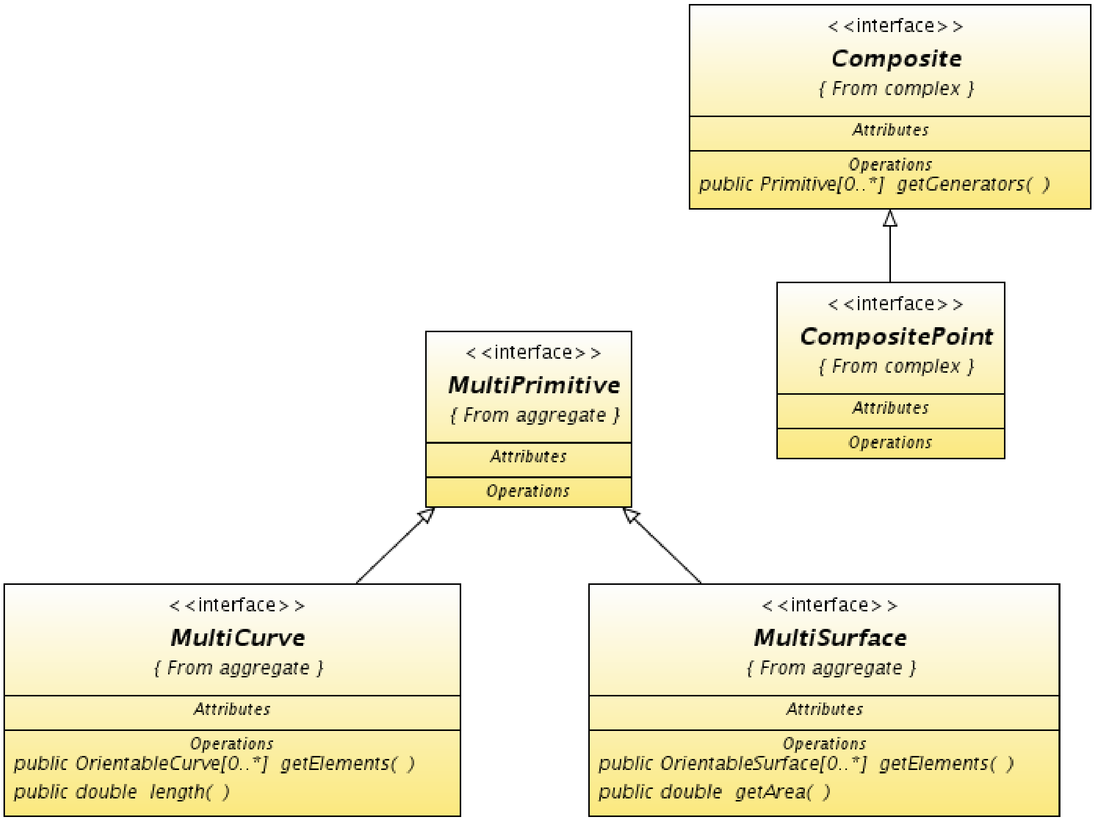
Add interfaces defined in ISO 19107 but not yet implemented as Java interfaces.

6.1.3 Reasons for change

Those interfaces are required by geometry implementors.

6.1.4 Specific suggested changes

Add the `MultiCurve`, `MultiSurface` and `CompositePoint` interfaces illustrated in the diagram below. Note: `MultiPrimitive` and `Composite` interfaces already exist and are unchanged.



6.1.5 Consequences of the change

This is a compatible change.

6.1.6 Consequences if not approved

Required interfaces would still missing.

6.2 Relax obligation of "element to owner" associations

Relax the "element to owner" associations obligation in geometry interfaces.
JIRA task: http://jira.codehaus.org/browse/GEO-63

According figure 11 (`GM_Curve` UML model) in ISO 19107, the `Segmentation` association is navigable from `GM_Curve` to `GM_CurveSegment` but doesn't need to be navigable in the opposite way. GeoAPI provides navigation in both direction, and wrongly annotate the `GM_CurveSegment` → `GM_Curve` navigation as mandatory. Some implementers reported to have ignored all "*element to owner*" associations. We propose to make optional all methods implementing an association in an "*element to owner*" way not required by ISO 19107.

6.2.1 Affected section(s), table(s), and figure(s)

In [OGC 03-064r10]: figures 14, 15, 16.

6.2.2 Purposes of the proposed change

Bring GeoAPI and GO-1 more inline with ISO 19107, which does not require this navigability.

6.2.3 Reasons for change

The "*element to owner*" associations are not required by ISO 19107 and uneasy to implement, so some implementers ignore them.

6.2.4 Specific suggested changes

We propose to annotate the `CurveSegment.getCurve()` method as optional. We propose to apply the same to every methods implementing a "children to parent" navigability not required by ISO 19107. Affected methods are:

• `CurveSegment.getCurve()`	from association `Segmentation` in fig. 11
• `SurfacePatch.getSurface()`	from association `Segmentation` in fig. 21
• `OrientablePrimitive.getPrimitive()`	from association `Oriented` in fig. 10
• `OrientableCurve.getComposite()`	from association `Composition` in fig. 26
• `OrientableSurface.getComposite()`	from association `Composition` in fig. 26

To make the intend more readable, we propose to override some methods with narrowed specification:

• `Primitive.getProxy()`	obligation is `CONDITIONAL`
• `Point.getProxy()`	obligation is `FORBIDDEN`
• `Curve.getProxy()`	obligation is `MANDATORY`
• `Solid.getProxy()`	obligation is `FORBIDDEN`

6.2.5 Consequences of the change

Annotation change in some methods in GeoAPI interfaces which were not used by implementors anyway.

6.2.6 Consequences if not approved

Those methods are likely to stay ignored by implementers and raise some confusion on user side.

6.3 Add `ParamForPoint.getDistances()`

Add a `getDistances()` method in `ParamForPoint`
JIRA task: http://jira.codehaus.org/browse/GEO-64

ISO 19107 section 6.4.7.5 defines the following method:

```
GM_GenericCurve::paramForPoint(p : DirectPosition) : Set<Distance>,
DirectPosition
```

In my understanding, this is multiple return values where the first return value is a set and the second one a single value. Since Java do not supports multiple return values, GeoAPI returns a `ParamForPoint` object which contains both the distance and the position. However, this `ParamForPoint` currently interface contains only the following methods:

- `double getDistance();`
- `DirectPosition getPosition();`

It should have something like the following method in addition:

- `Set<Number> getDistances();`

6.3.1 Affected section(s), table(s), and figure(s)

None.

6.3.2 Purposes of the proposed change

Bring GeoAPI more inline with ISO 19107.

6.3.3 Reasons for change

The method is missing in GeoAPI interface compared to ISO 19107 specification.

6.3.4 Specific suggested changes

Add a `getDistances()` (plural form) method in the `ParamForPoint` interface. We would keep the existing `getDistance()` (singular form) method as a convenient and more efficient way to get the distance in the common case where the curve is simple.

6.3.5 Consequences of the change

GeoAPI implementors would be required to add the above-cited methods in their implementations. No action needed for GeoAPI users.

6.3.6 Consequences if not approved

GeoAPI users would not be able to get all distances for non-simple curve. It would be more restrictive than what ISO 19107 section 6.4.7.5 allows.

6.4 Add `geometry.aggregate.MultiPoint.getElements()`

Modify the interface `geometry.aggregate.MultiPoint` to include a method `Set <Point> getElements()`
JIRA task: none

Add a method defined in ISO 19107 but not yet present in GeoAPI.

6.4.1 Affected section(s), table(s), and figure(s)

none.

6.4.2 Purposes of the proposed change

Bring GeoAPI inline with ISO 19107.

6.4.3 Reasons for change

Geometry implementors need this method.

6.4.4 Specific suggested changes

Add a `Set <Point> getElements()` method as defined in ISO 19107.

6.4.5 Consequences of the change

This is a compatible change.

6.4.6 Consequences if not approved

GeoAPI would not conform to the ISO 19107 specification.

6.5 Rename `Geometry.getDistance(Geometry)` to `distance(Geometry)`

Add the method `Geometry.distance(Geometry)`
Deprecate the method `Geometry.getDistance(Geometry)` method and redirect to `Geometry.distance(Geometry)`.
JIRA task: http://jira.codehaus.org/browse/GEO-112

6.5.1 Affected section(s), table(s), and figure(s)

In [OGC 03-064r10]: Figure 13.

6.5.2 Purposes of the proposed change

Bring GeoAPI in line with Java naming conventions.

6.5.3 Reasons for change

The current `Geometry` interface has a minor deviation from usual Java naming conventions. The `distance(Geometry)` method computes the return value from the given argument. It is not a getter method returning a geometry property, consequently the `get` prefix is inappropriate.

6.5.4 Specific suggested changes

Add the `distance(Geometry)` method as defined in ISO 19107. Deprecate the `getDistance(Geometry)` method and redirect its calls to `distance(Geometry)`. The `getDistance(Geometry)` method would be removed in the future.

6.5.5 Consequences of the change

Existing implementations and user code would continue to work, but would need to be changed to use the `distance` rather than `getDistance` method prior to the removal of the old method in a future version of GeoAPI.

6.5.6 Consequences if not approved

GeoAPI would not strictly conform to the usual Java naming conventions.

7. Referencing

7.1 Add class `cs.RangeMeaning`

Add the `RangeMeaning` code list from ISO 19111 revision 04-046r3
JIRA task: http://jira.codehaus.org/browse/GEO-23

ISO 19111 revision 04-046r3 defines `RangeMeaning`, which was not in revision 03-073r1 (the revision used by current GeoAPI interfaces). We propose to add a new `RangeMeaning` code list in the `org.opengis.referencing.cs` package, and the three new methods specified in ISO 19111 to the existing `CoordinateSystem` interface.

7.1.1 Affected section(s), table(s), and figure(s)

In [OGC 03-064r10]: figure 20.

7.1.2 Purposes of the proposed change

Restore GeoAPI conformance with ISO 19111.

7.1.3 Reasons for change

ISO 19111 specification has been updated.

7.1.4 Specific suggested changes

Add the `RangeMeaning` code list in the `org.opengis.referencing.cs` package.

Add the following three methods to the `cs.CoordinateSystemAxis` interface (derived from table 30 in ISO 19111 revision 04-046r3):

- `double getMinimumValue();`
  returns `Double.NEGATIVE_INFINITY` if unbounded.
- `double getMaximumValue();`
  returns `Double.POSITIVE_INFINITY` if unbounded.
- `RangeMeaning getRangeMeaning();`

7.1.5 Consequences of the change

GeoAPI implementors would be required to add the above-cited methods in their implementations (even if they choose to return null values). No action needed for GeoAPI users.

7.1.6 Consequences if not approved

GeoAPI would be left behind ISO 19111. Implementors may have to rely on heuristic rules in order to guess if an axis has “wraparound” range meaning.

7.2 Add interface `crs.GeodeticCRS`

Add a `GeodeticCRS` interface as specified in ISO 19111 revision 04-046r3
JIRA task: http://jira.codehaus.org/browse/GEO-62

ISO 19111 revision 04-046r3 defines `GeodeticCRS`, which was not there in revision 03-073r1 (the revision used by current GeoAPI interfaces). We propose to add a new `GeodeticCRS` interface in the `org.opengis.referencing.crs` package.

`GeodeticCRS` replaces both `GeographicCRS` and `GeocentricCRS` interfaces in revision 04-046r3. For GeoAPI, we do not suggest to delete those interfaces. We suggest to set them as `GeodeticCRS` sub-interfaces instead. This is both in order to preserve compatibility with existing applications, and also because the `GeographicCRS` interface has real value to developers. Geographic CRS are widely used, and the `GeographicCRS` interface provides type-safety for applications that require this specific kind of CRS.

7.2.1 Affected section(s), table(s), and figure(s)

In [OGC 03-064r10]: figures 22 and 23.

7.2.2 Purposes of the proposed change

Bring GeoAPI and GO-1 more inline with ISO 19111.

7.2.3 Reasons for change

ISO 19111 specification has been updated.

7.2.4 Specific suggested changes

We propose to add a new `GeodeticCRS` interface as a sub-interface of `SingleCRS`, and set `GeographicCRS` and `GeocentricCRS` as sub-interfaces of `GeodeticCRS`.

We therefore comment out the `getDatum()` method of both `GeographicCRS` and `GeocentricCRS` since the method is required by `SingleCRS` which is now a parent interface.

7.2.5 Consequences of the change

This is a compatible change both for implementors and users.

7.2.6 Consequences if not approved

GeoAPI would not conform to the ISO 19111 specification.

7.3 Add interface `ReferenceIdentifier`

Add the `ReferenceIdentifier` interface of ISO 19115:2003
JIRA task: none.

ISO 19115:2003 defines `RS_Identifier`, as shown in Figure A.9 in section A.2.7 of 19115:2003. We propose to add a `ReferenceIdentifier` interface class in the `org.opengis.referencing` package as a sub-class of `org.opengis.metadata.Identifier` to better mirror the actual specifications.

7.3.1 Affected section(s), table(s), and figure(s)

In [OGC 03-064r10]: Figure “2” on Page 59.

7.3.2 Purposes of the proposed change

Bring GeoAPI back into compliance with the updated specification.

7.3.3 Reasons for change

This is a user request for a closer match between GeoAPI interfaces and ISO 19111. In GeoAPI 2.0 the Identifier interface combined `RS_Identifier` and `MD_Identifier`.

7.3.4 Specific suggested changes

Add interface file `ReferenceIdentifier.java`, and move the `Identifier.getVersion()` method to `ReferenceIdentifier`.

Change the javadoc and two methods in `referencing/IdentifiedObject.java` to reflect the new interface.

- `ReferenceIdentifier getName();`
  changed the return type from `Identifier`
- `Set<ReferenceIdentifier> getIdentifiers();`
  changed the return type from `Set<Identifier>`

Deprecate the method `getVersion()` in the `org.opengis.metadata.Identifier` interface.

7.3.5 Consequences of the change

Users and implementers would have to choose between the `Identifier` and `ReferenceIdentifier` interfaces depending on their use of `MD_Identifier` and `RS_Identifier`, respectively. This decision would be required before the `getVersion()` method is removed in a future version of the GeoAPI.

7.3.6 Consequences if not approved

GeoAPI would not conform to the ISO 19111 and ISO 19115 specifications.

8. Filter

8.1 Version 1.1 support

Implement the changes to support version 1.1 of the Filter Encoding Implementation Specification.

JIRA task: http://jira.codehaus.org/browse/GEO-120

GeoAPI 2.1/3.0 has been aligned with version 1.1 of the Filter Encoding specification. We propose implementing the following changes in order to support the new version of the Filter specification.

Implement the expanded `Identity` classes. This required moving from the old 1.0 specification identification system using `FeatureID` to the new 1.1 system using `GmlObjectId` as explained in section 11 of the 04-095 specification. GeoAPI proposed to create the new `filter.identity` package to implement this change. Add the `filter.Id` interface. Deprecate the `filter.FeatureID` interface in favour of the `filter.identity.FeatureID` which is still used for GML 2. Add `GmlObjectID` matching the specification.

Implement the capability classes to support the new capability query information. Version 1.1 of the FilterEncoding specification requires implementations to support the creation of a capabilities fragment for the Capabilities document returned by a Web Feature Server as seen in section 16 of the 04-095 specification. We propose adding the `filter.capabilites` package to implement this change.

Implement the sorting features of the 1.1 version of the specification. Version 1.1 of the Filter Encoding specification adds a sorting system. We propose adding the `filter.sort` package to implement this change.

8.1.1 Affected section(s), table(s), and figure(s)

In [OGC 03-064r10]: figures 1 and 35, 36.

8.1.2 Purposes of the proposed change

These changes bring the GeoAPI representation of Filter in line with the Filter 1.1 specification.

These allow the GeoAPI representation of Filter to be used on additional content other than just Features. This is consistent with the reuse of the Filter specification by both the Web Feature Server specification and the Catalog specification.

8.1.3 Reasons for change

These changes were undertaken to align GeoAPI Filter interfaces with the Filter 1.1 specification.

These changes were undertaken to allow GeoAPI Filter interfaces to work on additional Objects other than Feature. This allows Java applications to use reuse Filter over a range of subject matter; most specifically in making requests from an OGC OWS Catalog.

These changes allow GeoAPI Expression to work on a range of additional subject matter. Interfaces making use of Expression also benefit from an increase in scope.

8.1.4 Specific suggested changes

Add the new expressions and filters. We propose creating the following interfaces:

- `ExcludeFilter`
- `NilExpression`
- `IncludeFilter`
- `PropertyIsNotEqualTo`

Add the following methods and fields:

- `filter.BinaryComparisonOperator.isMatchingCase()`
- `filter.FilterFactory.equal(...)`
- `filter.FilterFactory.notEqual(...)`

  All three methods support the choice of matching or ignoring the case of characters in Comparison operations, as described on page 13 of the 04-095 specification document.

- `filter.expression.Expression.NIL`

  To support the new `NilExpression` interface.

- `filter.Filter.INCLUDE`
- `filter.Filter.EXCLUDE`

  To support the new include and exclude filters

- `filter.expression.Expression.eval(Object o)`

  Deprecating the `.eval(Feature f)` method; note this last call is still permitted since `Feature` is an `Object`.

- `filter.expression.Expression.evaluate(Object o)`
- `filter.Filter.evaluate(Object o)`

  Changed from `.evaluate(Feature f)` to work with any kind of data such as metadata. Also leaves room for future changes to enable Graphics and SLD to play in the same space.

- `filter.expression.Expression.evaluate(Object object, Class<T> context)`

  To specify the "context" in which the value is to be used, used to resolve ambiguity when evaluating Filter expressions (where a String is the lowest common denominator).

- `filter.FilterFactory.id(Set<Identifier> ids)`

  Needed to allow the creation of these Filter 1.1 constructs.

- `filter.FilterFactory.featureId(String id)`
- `filter.FilterFactory.gmlObjectId(String id)`

  Needed for to allow for the creation of these Filter 1.1 constructs (in particular we needed these when parsing Filter 1.1. documents)

- `filter.FilterFactory.sort(...)`

Supports the sort by contract

- `filter.FilterVisitor.visitNullFilter(Object o)`
    - enable the visitor to handle a 'null' event when a Filter was expected
- `filter.FilterVisitor.visit(Id filter, Object extraData)`
    - replaces `visit(FeatureId filter, Object extraData)`

Deprecate the following methods:

- `filter.FilterVisitor.visit(PropertyIsNotEqualTo filter, Object extraData)`
    - A mistake from GeoAPI 2.0

8.1.5 Consequences of the change

These changes allow the Filter interfaces to represent a Filter 1.1 document.

These changes allow the Expression interfaces to work on a range of objects including Feature, Medata, and Graphic.

Data structures that make use of Expression also benefit, as an example GeoAPI Style interfaces can now be expected to work on both Features and Graphics. This will allow us to remove duplication when we update GeoAPI to the SLD 1.1 specification over the course of a future change request.

8.1.6 Consequences if not approved

If these changes are not approved the GeoAPI Filter interfaces would not be usable as a representation of Filter 1.1 documents.

If these changes are not approved will will need to explore other avenues for filtering Metadata and Graphic. The most direct solution is making Record and Metadata extend Feature, as seen in the Deegree project, this would bring those interfaces out of alignment with the ISO 19103 and ISO 19115 specifications.