# GeoServices REST API – relationship with the OGC baseline

OGC document 12-062r1

22 June 2012

## General remarks

The GeoServices REST API is a submission of existing, proven technology to OGC – not a new standard that is developed mostly from scratch within OGC's Interoperability and Specification program. I.e., the GeoServices REST API has been developed outside of OGC over several years. As a result, the specification is for most parts not based on the current OGC baseline. However, it is in general aligned with the OGC abstract specifications. In OGC, the abstract specifications serve as the common conceptual foundation for OGC standards:

"The Abstract Specification provides the conceptual foundation for most OGC specification development activities. OGC standards are built and referenced against the Abstract Specification, thus enabling interoperability between different brands and different kinds of spatial processing systems. The Abstract Specification provides a reference model for the development of OGC standards."[[1]](#footnote-1)

This document is therefore structured by the topics of the OGC abstract specifications.

The GeoServices REST API is an implementation specification. Whenever significant overlapping capabilities with standards of the OGC baseline exist, the alignment with the standard is discussed.

The GeoServices REST API differs in a number of aspects from the current OGC baseline standards – in a complementary way:

* The services use JSON as the standard encoding of data. Support for a JSON encoding is so far not standardized in the relevant standards of the OGC baseline.
* The services of the GeoServices REST API have been developed as part of a single specification and are part of a consistent framework.

While it would be possible to develop new versions of the OGC Web Services standards using a consistent framework and with support for JSON representations and a RESTful "binding", this will likely take significant time due to the unresolved REST-related discussion items, the current organization of OGC SWGs based on the individual standards and the fragmentation into separate standards.

On the other hand, the GeoServices REST API offers a proven specification that is available today and which is used operationally in a large number of applications. I.e., it provides an existing "ecosystem" that other implementations and deployments can build upon.

* The current OGC Web Services usually provide specific capabilities often targeted to comprehensively cover a range of complex use cases and requirements. The success of the OGC standards has shown that they clearly address a market requirement. However, for many use cases a simpler set of requirements is sufficient – and the GeoServices REST API addresses this market. This leads to some restrictions compared to capabilities included in other OGC standards (see below). Support for more complex use cases is traded for simplicity and in some cases improved performance. Simplicity in the sense that both the development of client software and the configuration of services will typically be relatively simple.

A key difference between the GeoServices REST API and the OGC and ISO 19100 standards is that the underlying data model is firmly based on the data model of an underlying database that supports SQL and Simple Feature for SQL (SF-SQL). The OGC standards on the other hand in general use models that are based on the capabilities of UML or XML Schema.

In implementations, this makes a significant difference. The implementation of encoding data in JSON or executing queries according to the GeoServices REST API is straightforward, if a SF-SQL-enabled database is used for data storage. On the other hand, WFS implementations that use a SF-SQL-enabled database for data storage (and this seems to be the most typical approach in practice) will have to implement a non-trivial mapping between XML Schema and the data model in the database – both for encoding the data in XML as well as to transform queries to SQL. As WFS provides no restrictions on the use of XML Schema, a conformant implementation in principle will have to support it all.

Similar issues exist on the client side where commonly used GIS clients are often capable of consuming data encoded using tabular structures, but not data that uses more complex data structures as supported by UML or XML Schema. However, if a client accesses a WFS it has to be prepared to consume data according to XML Schemas of any complexity and quite often this is not the case today. The level 0 of the GML Simple Features Profile is an attempt to address this issue, but to do this properly there would need to be a WFS standard (or conformance class) that restricts the model complexity accordingly. The same applies for WMS that support GetFeatureInfo or SLD. Etc.

There is without doubt a significant market demand for more expressive or complex models and the standards of the current OGC baseline support this demand well. But where this is not required, it is faster, simpler and cheaper to develop and deploy services and clients using data models aligned with the underlying data storage technology.

This is discussed in more detail in the relevant sections below.

This approach to have specific constraints in an implementation standard against the complete model specified in the OGC abstract specifications quite common in existing OGC standards, too. For example, the GML Simple Feature profile level 0 limits the maximum multiplicity of all feature properties to 1 while the OGC feature model has no such restriction. Or, the Simple Feature Common Architecture restricts (in the current version) all geometries to those with linear interpolation while the OGC Feature Geometry abstract specification specifies other interpolation types, too.

## Topic 1 – Feature Geometry (ISO 19107)

The geometry model of the GeoServices REST API is consistent with the Simple Feature Common Architecture and as such with Topic 1 / ISO 19107 (a normative reference of the Simple Feature Common Architecture).

The encoding of geometries specified in the GeoServices REST API is a JSON encoding of the Simple Feature model and covers the geometries specified in the Simple Feature Common Architecture.

GeoJSON, a JSON geometry encoding developed outside of OGC, is not used in the GeoServices REST API as the JSON encoding of the GeoServices REST API predates GeoJSON. However, the extension mechanisms in the GeoServices REST API allow implementations or future extensions to support GeoJSON, too.

## Topic 2 – Spatial refererencing by coordinates (ISO 19111)

Spatial references in the GeoServices REST API, too, are consistent with the Simple Feature Common Architecture and as such with Topic 2 / ISO 19111 (a normative reference of the Simple Feature Common Architecture).

The encoding for coordinate reference systems is the Well-Known Text (WKT) representation specified in the Simple Feature Common Architecture standard. Note that WKT references the 2003 version of ISO 19111. In the meantime, Topic 2 and ISO 19111 have been updated, but WKT hasn't.

In addition to WKT, the GeoServices REST API specifies a list of well-known coordinate reference systems which are identified by a numeric identifier. These will be submitted to the OGC-NA along with their definition in WKT to register a persistent OGC URI for each coordinate reference system.

The GeoServices REST API is aligned with the OGC Axis Order Manifesto. All spatial coordinate reference systems used in the API are either geodetic or projected 2D systems and the axis order is defined for all coordinate reference systems. For example, in geodetic coordinate reference systems, longitude is always the first axis, latitude the second.

## Topic 3 - Locational Geometry Structures

As discussed above, spatial coordinate reference systems used in the API are either geodetic or projected 2D systems. For map data, image coordinates are used, too. The use of all coordinate reference systems is consistent with Topic 3.

## Topic 4 - Stored Functions and Interpolation

The GeoServices REST API Image Service provides stored function capabilities described in this topic. The specific stored functions supported are documented in the discussion of Topic 6 below.

## Topic 5 – Features

The GeoServices REST API is consistent with Topic 5 and ISO 19109 and supports a profile of the General Feature Model.

The GeoServices REST API uses the following terminology in the feature context that differs from the use in the OGC baseline: A feature is a "Topic 5 feature" with a geometry property. A table is a "Topic 5 feature type" without a geometry property. A layer is either a composition of other layers or a feature type with a geometry property plus additional information related to the presentation of the features.

The profile of the General Feature Model has the following restrictions:

* Only one geometry property per feature type

The GeoServices REST API feature model supports only one geometry property per feature while no such limitation exists in Topic 5 or ISO 19109. The limitation in the GeoServices REST API is deliberate as in practice it is not a limitation for most data providers, but it simplifies user interfaces and software development.

For cases, where a feature should be associated with multiple geometries, this could be represented in the GeoServices REST API feature model, too. The "default" geometry of the feature becomes the geometry of the feature in the GeoServices REST API. An additional geometry property of the feature is then represented as an additional related feature that has the additional geometry and a relationship to the "main" feature.

This represents the information of the full feature model except for two aspects:

1. The related features may not exist without a relationship to a main feature. To make this enforceable, a requirement could be added that such constraints must be supported by the server implementation for relationships.
2. It is not obvious that the additional, related feature is conceptually not a feature on its own, but represents an additional geometry of the main feature. One way to make this explicit (so that a client can discover this in cases where this is considered important) could be using the name of the relationship.
* Properties with complex values or a maximum multiplicity greater than 1, use of inheritance

The GeoServices REST API uses a data model based on the model used by relational database management systems – with additional support for geometries as discussed above. As a result, properties in an application schema with structured values (data types), properties with a maximum multiplicity greater than 1, or type hierarchies in general either need to be avoided or converted to the GeoServices REST API feature model (in which case this is in principle no different than the conversion of an application schema to an SQL definition in the context of the Simple Feature Access – SQL standard). While the latter is possible, it is generally recommended to use the GeoServices REST API in conjunction with data that fits well with its simpler feature model.

* Feature identifiers

The GeoServices REST API uses for performance reasons integer values for object identifiers managed by the server. The approach is consistent with the requirements for persistent feature identifiers in Topic 5.

ISO 19118 (not an OGC abstract specification) uses string as the value domain for object identifiers. As the object identifiers are created and maintained by the server, the use of integers is not inconsistent with ISO 19118 as all integers may be represented unambiguously as strings. This is similar to WFS where the server creates and maintains the object identifiers using a subset of strings (based on the ID type of XML), too.

* Due to the use of JSON as the standard representation, the values of any non-spatial attribute (see Topic 1 for these) are limited to the standard JSON types: strings, numbers, and boolean. Date/time values are encoded as numbers (number of milliseconds since epoch (January 1, 1970) in UTC). This is a more restricted set of types than offered by ISO/TS 19103 or XML Schema.

In summary, the GeoServices REST API feature model supports similar feature models like the GML Simple Feature Level 0 (GML-SF L0), but with the following differences:

* GML-SF L0 supports multiple geometry properties per feature, GeoServices REST API only one.
* GML-SF L0 supports additional value types like date/time, measurements, hyperlinks, etc. where in the GeoServices REST API only more basic types are supported (the standard JSON types) and other types have to be mapped to these.
* GeoServices REST API supports tables that can be related to a feature type to attach additional and more structured attributes to a feature.

## Topic 6 - Schema for coverage geometry and functions (ISO 19123)

The GeoServices REST API is consistent with Topic 6. It supports only a subset of the full conceptual model specified by Topic 6. In the Image Service, the following coverages are supported:

* the domain is a 2D rectified grid and the coordinate reference system is a 2D projected or geodetic CRS;
* the range for each cell consists of one or more numeric values (bands). The data type for each numeric value is same, e.g. an unsigned 8-bit integer, and each cell has the same number of bands.
* the interpolation types supported during resampling are bilinear and nearest neighbor (both listed in Annex C of Topic 6) – plus cubic convolution (looks at the 16 nearest cell centers and fits a smooth curve through the points to find the value) and majority resampling (determines the new value of the cell based on the most popular values within the filter window).

As a consequence, the more specific term image is used instead of the term coverage.

The resource model of the GeoServices REST API Image Service goes beyond Topic 6 in making the underlying raw raster files and their mosaicking transparent. The raster data sets and files are modelled as features. Note that this is, however, not part of the core conformance class.

## Topic 7 - Earth Imagery (ISO 19101-2)

This Topic is in fact a reference model for imagery. As such, the GeoServices REST API supports only part of the capabilities specified in Topic 7. There is no known inconsistency with the abstract specification.

## Topic 8 - Relationships Between Features

The GeoServices REST API is consistent with Topic 8.

Natively, only lightweight, binary relationships are supported. As discussed in sub-clause 2.10 of Topic 8, such relationships may be used to represent more complex relationships, too.

## Topic 10 - Feature Collections

The GeoServices REST API is consistent with Topic 10.

In particular, the following types of feature collections identified in Topic 10 (see sub-clause 1.2) are used as part of the specification:

* GIS database files
* the persistent or non-persistent ad hoc collection of features present at any moment in a GIS workspace.

## Topic 11 – Metadata (ISO 19115)

The GeoServices REST API follows a similar approach to representing metadata like OGC capabilities documents. I.e., metadata about a service or data is not explicitly represented using the types defined in Topic 11, but can in general be mapped to the corresponding ISO 19115 type. The same applies to service metadata (ISO 19119, topic 12).

For example, in the Map Service Root resource has several properties that represent general metadata about the service:

|  |  |  |
| --- | --- | --- |
| **GeoServices Map Service – Map Service Resource** | **Topic 11 / 12** | **OWS Common 2.0** |
| serviceDescription | SV\_ServiceDescription. abstract | ServiceIdentification/abstract |
| mapName | SV\_ServiceDescription.citation.title | ServiceIdentification/title |
| description | SV\_ServiceDescription.abstract | ServiceIdentification/abstract |
| copyrightText | SV\_ServiceIdentification.restrictions | ServiceIdentification/accessConstraints |
| fulLExtent | SV\_ServiceIdentification.extent | BoundingBox |
| documentInfo | other information, e.g. keywords or about responsible parties at the discretion of the service provider; in the GeoServices REST API represented as key-value-pairs |

Typically, serviceDescription will contain a description of the service and description an overview of the underlying data.

## Topic 12 - The OpenGIS Service Architecture (ISO 19119)

The GeoServices REST API service architecture is consistent with Topic 12.

There is overlap between its services and the existing OGC Web Service (OWS) standards, which is discussed in this section. A general difference is that the OWS standards use XML as the mandatory encoding while the GeoServices REST API requires JSON. Another difference is that OWS usually provide a single HTTP endpoint and use query parameters, XML or XML embedded in SOAP messages for remote procedure calls. The GeoServices REST API provides multiple resources each with their own URL, a client then navigates between the resources and interacts with them using HTTP (clause 6 of the Core document contains additional discussion on this).

For the **GeoServices REST API Catalog Service** see Topic 13 below.

The **GeoServices REST API Map Service** has overlap with three OGC Web Services: WMS, WMTS and WFS.

The functionality of WMS is basically covered by the GeoServices Map Service with the following simplifications:

* For the reasons discussed above, the features retrieved by an identify query are limited to the GeoServices REST API feature model and implementations are required to support JSON.
* In WMS, layers may be arbitrarily constructed from features. I.e., a feature may be part of multiple layers, the features in a layer may come from multiple feature types and not all features of a feature type may be in the same layer(s). In the GeoServices REST API this is simplified to a layer per feature type. This removes some flexibility, but simplifies the process of publishing data in map layers.
* In WMS, a layer may be presented using one of several styles. In the GeoServices REST API there is only one style per layer. In addition to simplifying the process of publishing data in map layers, this also reflects common practice as it is recommended to cache the map in a tile pyramid for performance reasons and using several styles would have a significant impact on the data storage for the map cache.

As the GeoServices Map Service specification has been developed in parallel to WMS, the parameter names and syntax of requests have some overlaps, but differ in general. The table below illustrates this as an example roughly comparing GeoServices Map Service and WMS parameters in the Export Map and GetMap operations:

|  |  |
| --- | --- |
| **GeoServices Map Service - Export Map** | **Web Map Service 1.3.0 - GetMap** |
| bbox | bbox |
| size | width,height |
| imageSR | crs |
| format | outputFormat |
| layers | layers |
| transparent | transparent |
| time | time |
| f | - |
| dpi | - |
| layerDefs | - |
| layerTimeOptions | - |
| - | elevation |
| - | exception |
| - | bgcolor |

The GeoServices Map Service parameters have been defined in the context of the complete GeoServices REST API framework and should be seen in this context – and not directly compared to parameters of the OGC Web Services. As discussed in the beginning, the GeoServices REST API services should be seen as complementary to the existing OGC Web Services. Over time, harmonization between the specifications should be considered with regards to the necessary migration for existing implementations.

The Map Service has overlap with parts of WMTS, but again does not provide all capabilities a WMTS has to support. In the Map Service a layer is stored in one tile pyramid with a fixed image format, a fixed coordinate reference system and in the style of the layer. WMTS supports multiple formats, coordinate reference systems, and styles. The restrictions in the GeoServices REST API were chosen as this reduces the resource requirements of the server (storage size, processing power) and is sufficient in many cases.

The integration of the capabilities to provide maps of user-defined scale, size and location (ExportMap, similar to GetMap of WMS) and access to cached tiles (similar to WMTS) in a single service has advantages as most of the data and metadata (layers, styling information, features) is the same no matter which resource is used. Separate WMS and WMTS services with no obvious relationship, but two capabilities documents, two GetFeatureInfo operations, etc, is more complicated both for the data publisher who has to configure the services and the user.

In addition to the overlap with WMS and WMTS as discussed above, the Map Service has overlap with a read-only WFS, too, as it supports queries. The reason for this approach is analogous to the one discussed in the previous paragraph as with a single service type a wide range a usage patterns for accessing information read-only from a server can be addressed.

The Map Service also has extensions beyond WMS/WMTS/WFS in that it allows attaching arbitrary documents to any feature or providing a HTML popup for any feature. These capabilities are often requested to improve the interaction of the user with the service.

The **GeoServices REST API Feature Service** has overlap with a transactional WFS. Read access to the data is similar to the Map Service as discussed above since the Map Service supports query capabilities. Like in the Map Service, the Feature Service uses the GeoServices REST API feature model described in the section on topic 5 and implementations are required to support JSON.

In addition to the query capabilities of the Map Service, the Feature Service also supports querying related features that are associated by a relationship. This corresponds to Web Feature Services with a capability for traversing/resolving one level of local Xlinks, i.e. links within the same data set.

Additionally, the Feature Service supports feature editing capabilities, similar to the WFS transaction interface:

|  |  |
| --- | --- |
| **GeoServices Map Service – Feature Service feature editing operation** | **WFS 2.0** |
| Add Features | Transaction with <Insert> elements only |
| Update Features | Transaction with <Replace> elements only |
| Delete Features | Transaction with <Delete> elements only |
| Apply Edits | Transaction with <Insert>, <Replace> and <Delete> elements; delete instructions are restricted to selections by the resource identifier |

Like the Map Service, the Feature Service has extensions beyond WFS in that it allows attaching arbitrary documents to any feature, provide a HTML popup for any feature, or edit these.

Unlike WFS, the Feature Service also provides information about the styling of features in maps to support applications directly that, for example, provide a feature editing capability. Such applications may present feature information in maps on the client side just by communicating with the Feature Service; there is no need to interact with a separate Map Service.

Another extension that the Feature Service provides beyond WFS or other OWS are feature templates. These are associated with a feature type and may provide information to improve the workflow of creating new features in an application. A template contains two parts: Default values for the attributes the feature ("feature prototype") and a recommendation for a user interface tool for creating the feature geometry (e.g. the recommendation to use a tool for drawing a rectangle).

On the other hand, the Feature Service does not provide WFS capabilities such as stored queries, explicit locking, resolving remote references, response paging, feature versioning, sorting results, etc.

The **GeoServices REST API Image Service** has overlap with the Web Coverage Service (WCS).

The information about the underlying raster data (e.g. the range type and the extent) is provided in the Image Service Root resource and, if supported, the Raster Catalog Item resources where in WCS the GetCapablities and Describe Coverage operation responses provide this information.

In the Image Service, the Export Image resource provides access to images including support for resampling and clipping; in WCS, the GetCoverage operation provides similar capabilities.

One noteworthy difference is that Export Image supports portrayal via the rendering rule parameter, where in the OGC Web Services portrayal is strictly separated and a WMS/SLD on top of a WCS is required to provided such capabilities. The portrayal capabilities of the Image Service are lower than the coverage portrayal capabilities of SLD/SE[[2]](#footnote-2).

Another difference is that the Image Service provides capabilities to query and identify raster features. These capabilities are consistent with the same resources in the Map Service.

The Image Service is limited to coverages with the constraints discussed in the comparison with Topic 6 above.

Another difference already mentioned in the discussion of Topic 6 is that the Image Service provides access to the raster catalog and the raw raster files, a capability not supported by WCS or any other OWS standard.

This different packaging of the capabilities in the Image Service compared to the packaging of capabilities in the OWS standards was selected as this bundle of resources provides support for many of the image-related use cases via a single service.

In many cases, the specific capabilities of the Image Service are limited compared with the capabilities of WCS (e.g. supported coverage domains, range types and portrayal rules). These restrictions are intentional as it simplifies the resource model and interface, but at the same time already supports many of the image-related use cases.

The **GeoServices REST API Geoprocessing** **and Geometry Services** have overlap with WPS. The Geometry Service operations could be implemented as synchronously executable tasks in a Geoprocessing Service.

Like the Geoprocessing Service, WPS defines a standardized interface that facilitates the publishing of geospatial processes, and the discovery of and binding to those processes by clients. "Processes" are called "tasks" in the GeoServices REST API and include any algorithm, calculation or model that operates on spatially referenced data. "Publishing" means making available machine-readable binding information as well as human-readable metadata that allows service discovery and use. In the GeoServices REST API this information is provided via the Root and the Task resources.

As part of the GeoServices REST API, the usual constraints discussed earlier in this document apply, i.e. the feature model is the GeoServices REST API feature model and JSON is a mandatory representation. In addition to these general constraints, the Geoprocessing Service also specifies an explicit set of complex data structures where WPS leaves this more generic and open. As a result, all tasks in a Geoprocessing Service may be published as a WPS, but publishing arbitrary WPS processes using a Geoprocessing Service may not be possible or not straightforward.

Both the Geoprocessing Service and WPS support synchronous and asynchronous execution of tasks/processes.

The **GeoServices REST API Geocoding Service** has overlap with OpenLS Location Utility Service both provide similar geocoding and reverse geocoding capabilities.

## Topic 13 - Catalog Services

The GeoServices REST API Catalog is consistent with Topic 13.

The geospatial resources in the catalogue are restricted to services and for each service to information needed in practice to access and use each service. As a result, no query interface like in the OGC Catalogue Service (CSW) is required or supported and all catalogued resources are accessed by navigation from the root resource.

Using a CSW-type approach for the GeoServices Catalog Service would have added complexity to the interaction with the service but without getting benefits.

Also, unlike CSW, which is a generic catalog service, that may be used to catalog a wide range of resources, the GeoServices Catalog has a clear focus: It provides a catalog of all services provided by that server.

## Topic 15 - Image Exploitation Services

This Topic is a comprehensive description of services needed for image exploitation in various use cases. In most cases, the individual service types are described in other Topics in more detail. Some of the capabilities described in the Topic are supported by the GeoServices REST API Image Service, but many are not. There is, however, no conflict between the Topic and the GeoServices REST API.

## Topic 16 - Image Coordinate Transformation Services

The GeoServices REST API Image Service supports image coordinate transformations as described in Topic 16. However, the capabilities of the GeoServices REST API are limited to 2D projected and geodetic coordinate reference systems and 2D rectified grids. As not every Image Service may support such capabilities, these are not required by the core conformance class.

## Other OGC abstract specifications

The following OGC abstract specifications are not implemented by the GeoServices REST API:

* Topic 14 - Semantics and Information Communities
* Topic 17 - Location Based Mobile Services
* Topic 18 - Geospatial Digital Rights Management Reference Model (GeoDRM RM)
* Topic 19 – Linear Referencing Methodology (ISO 19148)
* Topic 20 – Observations and Measurements (ISO 19156)

## Other OGC standards

There are OGC standards that are not related to an OGC abstract specification. This includes portrayal related standards, i.e., Styled-Layer Descriptor (SLD), Symbology Encoding (SE) and KML.

To simplify the authoring, the management and the use of services, a simpler model than the one in SLD and SE is supported by the GeoServices REST API:

* As discussed in the Map Service, a one-to-one correspondence between a layer, a feature type with geometry, and a style exists where in SLD/SE a layer can consist of arbitrarily selected features and may have multiple styles.
* SE provides greater flexibility in expressing symbolizers and filter rules than the symbols and renderers of the GeoServices REST API does. This simplifies the definition of such styles and rules in graphic user interfaces, but already supports the creation of sophisticated maps.

In this, the GeoServices REST API is similar to KML which also follows a quite different conceptual model to portrayal than SLD/SE.

When more complex requirements to portrayal exist than supported by the GeoServices REST API, the use of WMS/SLD is recommended.

Unlike the GeoServices REST API, WMS/SLD also supports feature portrayal based on portrayal rules specified by the client.

1. http://www.opengeospatial.org/standards [↑](#footnote-ref-1)
2. see also the discussion under "Other OGC standards" below. [↑](#footnote-ref-2)