Open GIS Consortium, Inc.

OpenGIS® Simple Features Test Protocol

For CORBA

Release Date: July 24, 2001

Draft Date: July 23, 2001

1 Overview

This document describes the conformance testing guidelines that the Open GIS Consortium will use in the testing of software implementations for conformance to its specification entitled Open GIS Simple Features for CORBA, Revision 1.0.

The Open GIS Consortium, Inc. (OGC) maintains a brand (in the form of a certification mark) that cannot be used in connection with a software product by any organization unless they have submitted a software product to OGC's conformance testing, successfully completed this testing, and received a certificate stating such success. Organizations that have earned the certification mark may use it in ways defined within this document. This set of rules ensures that users who buy products that are branded can be sure that the products carrying the certification mark have been submitted to a testing process. The primary purpose of the conformance testing process is to protect the value of the OpenGIS® brand as an element of OGC's program to promote interoperability between diverse geoprocessing systems.

This document defines the following:

- A general description of the conformance tests and the scope of the tests
- A description of the test data
- A description of the test queries
- A description of allowable adaptations and guidelines for documenting these adaptations

2 Test Description and Scope

According to the OpenGIS Simple Features Specification for CORBA, Revision 1.0 specification (hereafter referred to as "the specification"), there are two modules.

FEATURE MODULE

GEOMETRY MODULE

The specification reads:

> *The purpose of this specification is to provide interfaces to allow GIS software engineers to develop applications that expose functionality required to access and manipulate geospatial information comprising features with 'simple' geometry using OMG's CORBA technology.*

And:

> *The specification is broad enough to allow maximal flexibility in implementation. In particular, it has been designed with two implementation models is mind:*
>
> - *the exposure of existing (legacy) geospatial data and applications whether they be RDBMS or proprietary file repositories through some form of object 'wrapping'.*
> - *the development of new distributed object-oriented GIS applications.*
>
> *This was undertaken to ensure that the OpenGIS Interoperability specification provides a low cost entry point for existing players in the GIS marketplace while allowing a natural progression towards implementations based on the increasingly popular and powerful distributed and object-oriented technologies such as Java and the Internet. In particular, care was taken to ensure that the powerful aspects of the O-O programming paradigm were exploitable through this specification.*

Test suite software is provided to verify conformance of:

1. Feature module interfaces, that enable to access features in a server independent way.

2. Geometry module interfaces that address geometries and Spatial Reference System.

The test suite software is provided as a Java application program. The source code for program is included, as it is necessary to modify the code for implementation-specific calls.

The test scripts check that all mandatory features have been implemented, and can be called, but they do not check that the returned values are correct. In general, the scope of the tests is to exercise each functional aspect of the specification at least once. The test questions and answers are defined to test that the specified functionality exists and is operable. Care has been taken to ensure that the tests are not at the level of rigor that a product quality control process or certification test might be. However, because some of the answers are further examined for reasonableness (for example, the area of a polygon is tested for correctness to two or three significant figures).

Unfortunately, the test scripts cannot be completely implementation independent. Small modifications must be made for a specific implementation. For example, each ORB products may have different method name and manners to calling methods. OGC recognizes that this test suite software will probably have to be adapted to work with software products seeking conformance.

The following sections further describe each test alternative.

2.1 Feature Module

The specification reads:

> *Real world entities such as "Roads" are typically represented as features comprising a set of spatial and non-spatial attribute values (e.g., a geometry such as a line string representing the road's spatial extent, a string representing its name, etc.). Features may have an associated set of operations or behavior. Features are also referred to as feature instances.*
>
> *System engineers categorize representations of these real world entities as feature types. A feature type defines the set of properties (and possibly behavior) that characterize features of that type. A property has a name and type. Properties may be of any (IDL) type, including simple types (shorts, longs, floats, strings etc.), constructed types (structs, unions, sequences) and object references (including references to other features). Every feature is of primarily one type (systems using type inheritance may allow features to be of multiple types – the use of inheritance, whether single or multiple, is an implementation issue).*
>
> *Feature collections are groups of features constructed for various purposes. Feature collections come in two fundamental flavors. Feature collections supporting the concept of "containment" own their constituent features i.e. the persistence of the member features is affected through the collection. If a feature is removed from its containing feature collection (without being moved to another) it ceases to exist. Other feature collections provide support for organizing and managing existing features without owning them i.e. features that are contained in other feature collections but which are grouped in some way towards a particular end: e.g. to scope a query. Features are contained in one and only one container feature collection, although they may be, by reference, members of other, non-containing, feature collections. A client's primary access to a given feature will typically be through a feature collection.*

2.2 Geometry Module

The specification reads:

> *Interoperable geoprocessing requires the unambiguous exposure of geometric entities. The set of interfaces included in this proposal provides a means through which various geoprocessors may expose geometric entities to each other. The interfaces are based on the following abstract model.*
>
> *All geometric entities belong to an abstract class of 'geometries'. All have a number of common characteristics e.g. all have spatial extent, all use some form of spatial reference system, etc.*
>
> *Geometries are categorized by their dimension as zero-dimensional geometries (points), one-dimensional geometries (curves) and two-dimensional geometries (surfaces). This model can be extended to three dimensions (solids), four dimensions (hyper-solids) and higher dimensions if necessary.*
>
> *Additionally, individual ('simple') geometries may be aggregated to form composite geometries or geometry collections. These geometry collections form a separate category of geometries. In general, geometry collections may be composed from geometries of different dimensionality (heterogeneous collections). Geometry collections, which comprise only geometries of a single dimension (homogenous collections), are specializations of this general type. Geometry collections may, of course, be further restricted by various implementations.*
>
> *All Geometries are capable of exposing their underlying coordinate geometries in the form of Well-known Structures (WKSs). The semantics of the coordinates (i.e. the mapping between*

coordinates in coordinate space and real world locations) is provided by a Spatial Reference System (SRS).

2.3 Option of Test Items

The specification reads:

The specification is broad enough to allow maximal flexibility in implementation.
The purpose of this test is to confirm interoperability of candidate implementations and they are tested if the minimum interfaces for interoperability are implemented.
This specification includes some interfaces and methods for client convenience or future enhancement as well as the essential interfaces.
To support these additional interfaces and methods, however, seems to place unreasonable obligations on server implementation. And SFCORBA should be aligned with SFSQL and SFCOM. For this reason, the following interfaces and methods are option so that OpenGIS compliance does not demand its implementation. For details of each interfaces and methods, see section 4.

2.4 Test Suite Program

As mentioned above, the test suite software is provided as a Java application program. The source code for program is included, as it is necessary to modify the code for implementation-specific calls and environment. After modifying java source file, it is necessary to compile them and create java – class files to be run the test suite program. Note that the test suite software is based on JDK version 1.2.1. The following steps describe the process of running the test:

Install Candidate Server Implementation

Installation of the candidate implementation will be specific to that implementation. No guidelines are given on the process for installing Server implementation.

Run the test program

The test program may be run as an executable. Submitting organizations may choose to run the test. The dialog shown below will be displayed, enabling the testing personnel to select the interface to e tested.

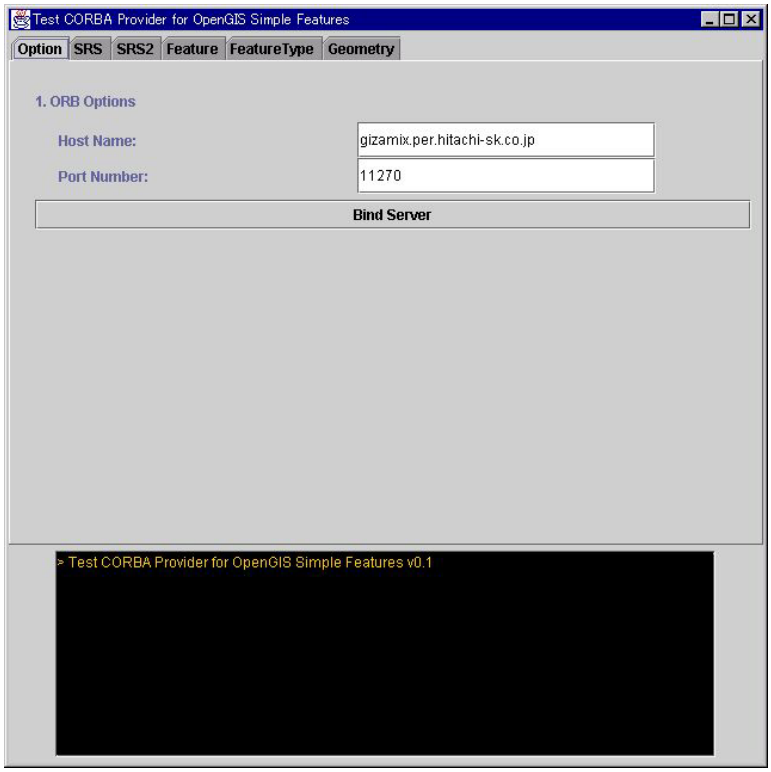
Choose the Tests You Wish to Perform

Testing personnel can use the check boxes to select the components of the candidate implementation to be tested.

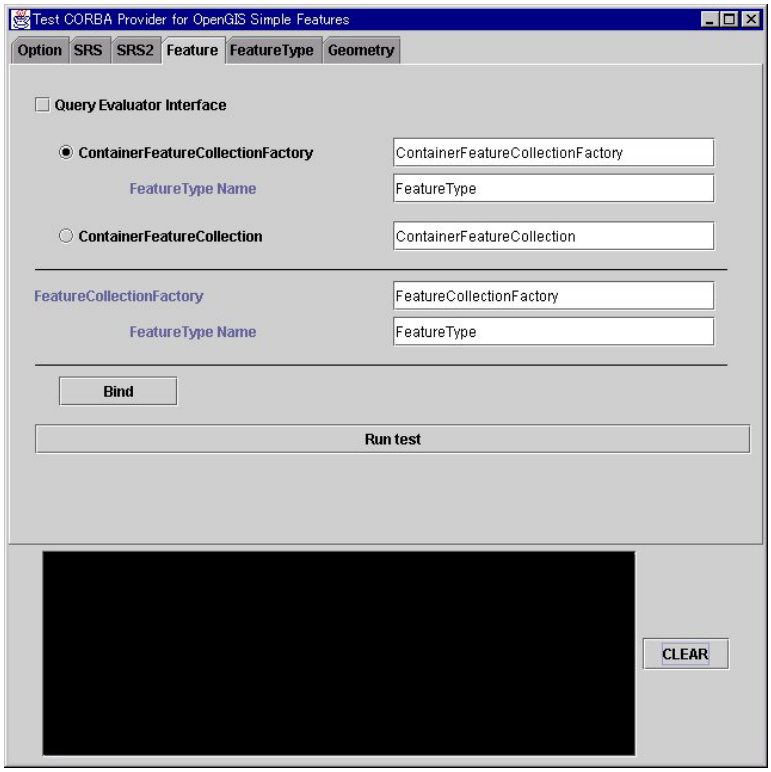
Press the "Run Tests" Button

When the test program is run, the selected tests are performed, and the dialog displays a message indicating success or failure for each test.

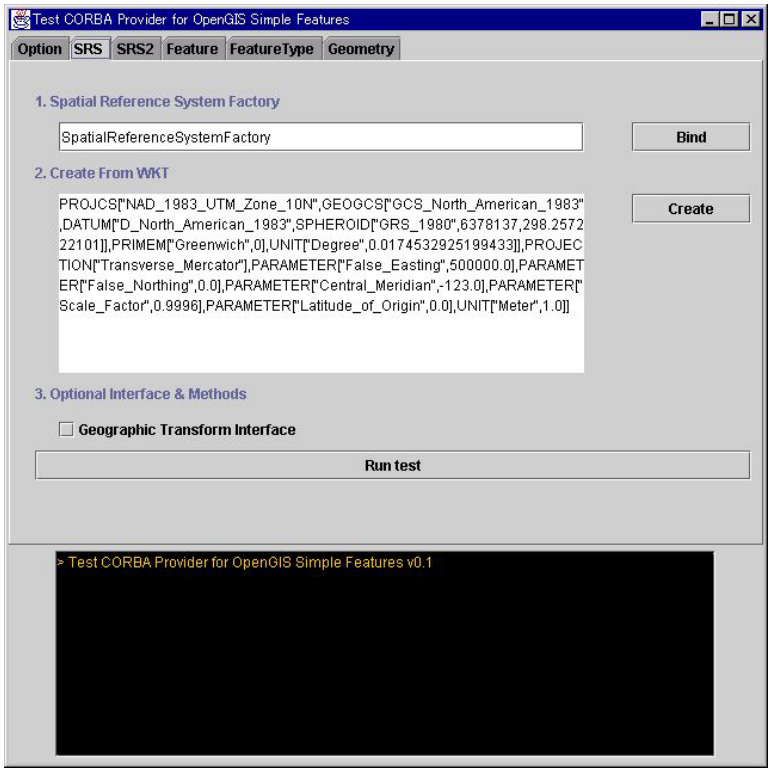
The dialog has some TABs (See following figures.) and initial settings for the test can be made at each TAB.



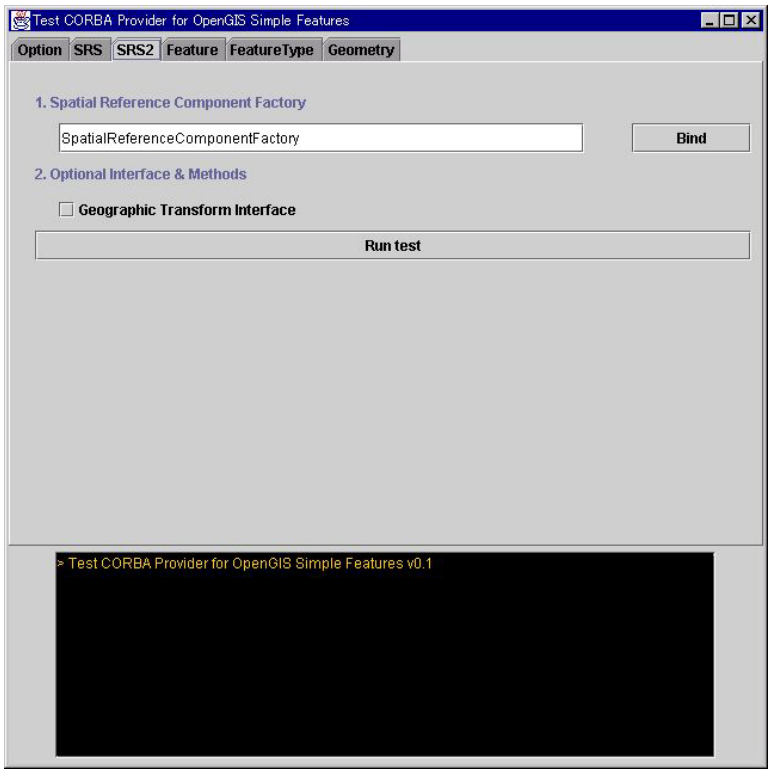
Testing personnel can set Host Name and Port Number to connect available ORB for test.



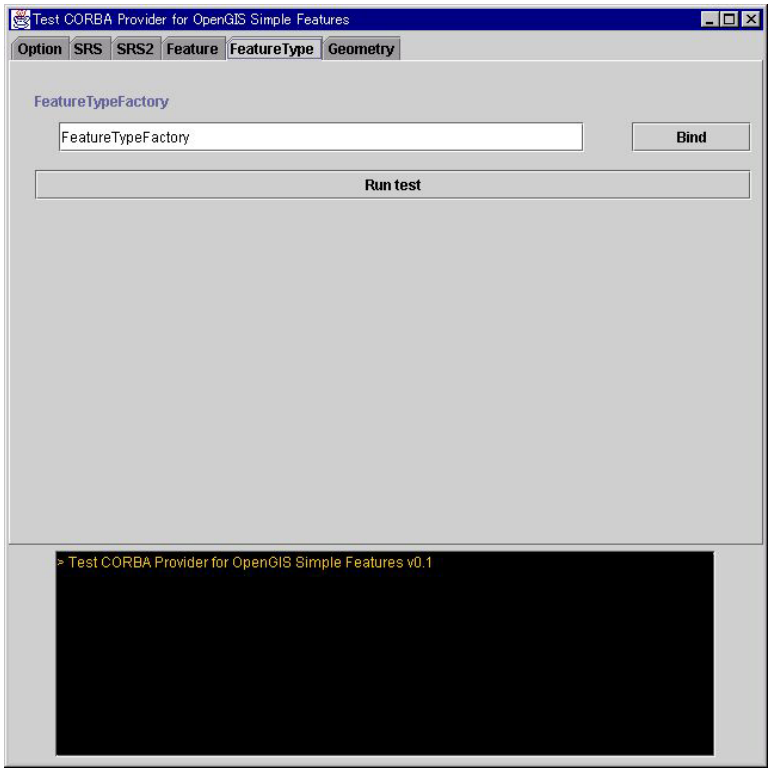
The Feature interfaces can be tested by using this dialog. Testing personnel can choose ContainerFeatureCollectionFactory or ContainerFeatureCollection to bind with the Server.



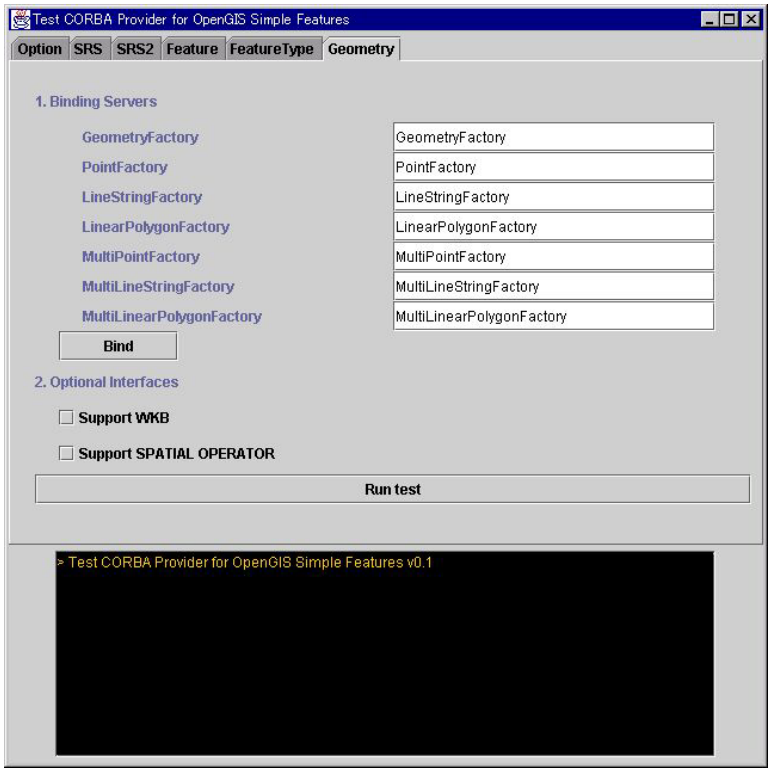
Testing personnel can specify the name of implementation's SpatialReferenceSystemFactory and WKT for creating. The Spatial Reference System to be created here will be used in the test for Geometry interfaces later.



The methods of SpatialReferenceComponentFactory interface can be tested from here. Before the test, it is necessary to specify the name of implementation's SpatialReferenceComponentFactory.



If candidate server implementation supports FeatureTypeFactory, the test for this interface can be done from here.



To test Geometry interfaces, it is necessary to specify the name of each implementation's Factory. Before this test, it is necessary to create Spatial Reference System for Geometry at the SRS TAB.

3 Test Data

The data is a synthetic data set, developed, by hand, to exercise the functionality of the specification. It is a set of features that make up a map (see Figure 1) of a fictional location called Blue Lake. This section describes the test data in detail.

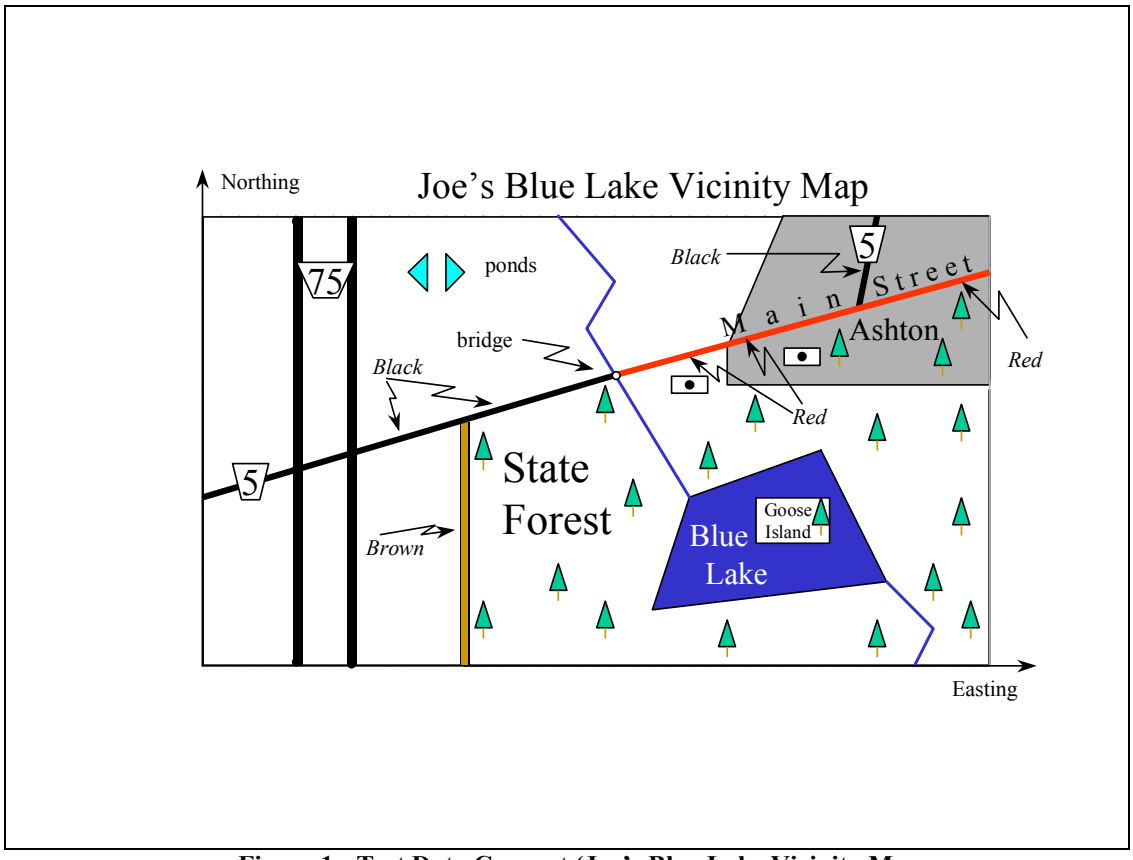


Figure 1: Test Data Concept (Joe's Blue Lake Vicinity Map

3.1 Test Data Semantics

The semantics of this data set are as follows:

A rectangle of the Earth is shown in UTM coordinates. Horizontal coordinates take meaning from POSC Horizontal Coordinate System #32214. Note 500,000 meters false Easting, and WGS74. Units are meters. (See http://www.petroconsultants.com/epsgweb/epsg.htm)

- Blue Lake (which has an island named Goose Island) is the prominent feature.
- There is a watercourse flowing from North to South. The portion from the top neat line to the lake is called Cam Stream. The portion from the lake to the bottom neat line has no name (Name value is "Null")

- There is an area place named Ashton.
- There is a State Forest whose administrative area includes the lake and a portion of Ashton. Roads form the boundary of the State Forest. The "Green Forest" is the State Forest minus the lake.
- Route 5 extends across the map. It is two lanes where shown in black. It is four lanes where shown in Red.
- There is a major divided highway, Route 75, shown in a double black line, one line for each part of the divided highway. These two lines are seen as a multi-line.
- There is a bridge (Cam Bridge) where the road goes over Cam Stream, a point feature.
- Main Street shares some pavement with Route 5, and is always four lanes wide.
- There are two buildings along Main Street; each can be seen either as a point or as a rectangle footprint.
- There is a one-lane road forming part of the boundary of the State Forest, shown in brown.
- The are two fish ponds, which are seen as a collective, not as individuals; that is, they are a multi-polygon.

3.2 Test Data Points and Coordinates

Figure 2 depicts the points that are used to represent the map.

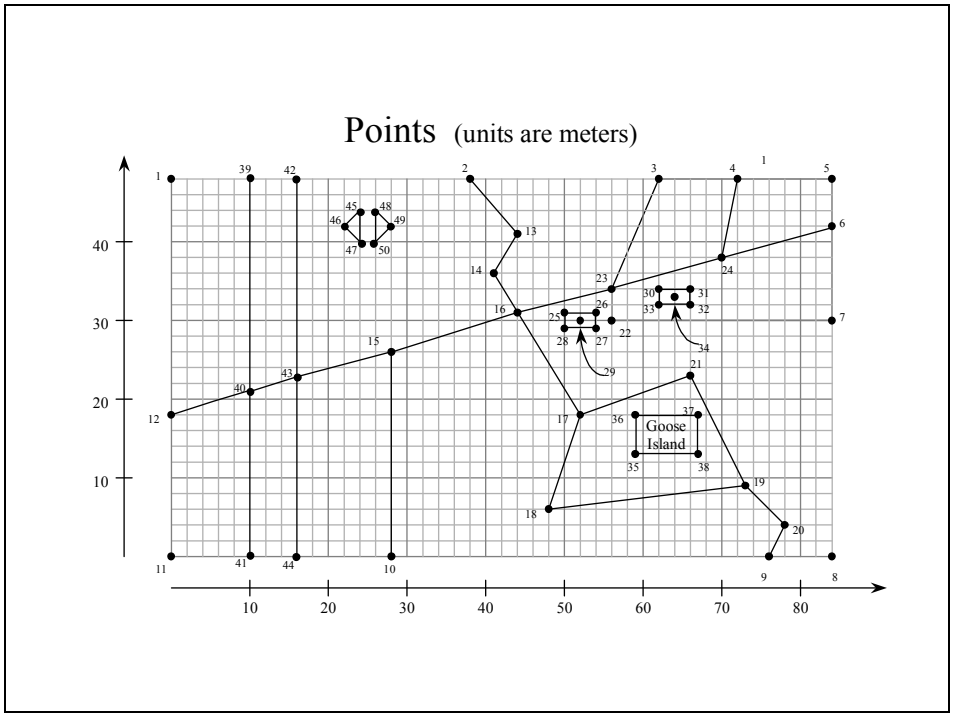


Figure 2: Points in the Blue Lake data set.

The following table gives these coordinates associated with each point.

Point	Easting	Northing	Point	Easting	Northing
1	0	48	26	52	31
2	38	48	27	52	29
3	62	48	28	50	29
4	72	48	29	52	30
5	84	48	30	62	34
6	84	42	31	66	34
7	84	30	32	66	32
8	84	0	33	62	32
9	76	0	34	64	33
10	28	0	35	59	13
11	0	0	36	59	18
12	0	18	37	67	18
13	44	41	38	67	13
14	41	36	39	10	48
15	28	26	40	10	21
16	44	31	41	10	0
17	52	18	42	16	48
18	48	6	43	16	23
19	73	9	44	16	0
20	78	4	45	24	44
21	66	23	46	22	42
22	56	30	47	24	40
23	56	34	48	26	44
24	70	38	49	28	42
25	50	31	50	26	40

The following gives entire test data in Well Known Text (WKT) format, as supplied in the file **sample.txt**:

"PROJCS['UTM_ZONE_14N', GEOGCS['World Geodetic System 72',DATUM['WGS_72', SPHEROID
['NWL_10D', 6378135, 298.26]],PRIMEM['Greenwich', 0], UNIT['Meter', 1.0]],
PROJECTION['Transverse_Mercator'], PARAMETER['False_Easting', 500000.0],
PARAMETER['False_Northing', 0.0], PARAMETER['Central_Meridian', -99.0],
PARAMETER['Scale_Factor', 0.9996], PARAMETER['Latitude_of_origin', 0.0],UNIT['Meter',1.0]]"
17
101,"Blue Lake","POLYGON((52 18, 66 23, 73 9, 48 6, 52 18), (59 18, 67 18, 67 13,59 13, 59 18))"
102,"Route 5","LINESTRING(0 18, 10 21, 16 23, 28 26, 44 31)"
103,"Route 5","LINESTRING(44 31, 56 34, 70 38)"
104,"Route 5","LINESTRING(70 38, 72 48)"
105,"Main Street","LINESTRING(70 38, 84 42)"
106,"Dirt Road by Green Forest","LINESTRING(28 26, 28 0)"
109,"Green Forest","MULTIPOLYGON(((28 26, 28 0, 84 0, 84 42, 28 26), (52 18, 66 23,73 9, 48 6, 52 18)), □
((59 18, 67 18, 67 13, 59 13, 59 18)))"
110,"Cam Bridge","POINT(44 31)"
111,"Cam Stream","LINESTRING(38 48, 44 41, 41 36, 44 31, 52 18)"
112,"Cam Stream","LINESTRING(76 0, 78 4, 73 9)"
113,"123 Main Street","GEOMETRYCOLLECTION(POINT(52 30), POLYGON((50 31, 54 31, 5429, 50 29, 50 31)))"
114,"215 Main Street","GEOMETRYCOLLECTION(POINT(64 33), POLYGON((66 34, 62 34, 6232, 66 32, 66 34)))"

115,"Neat Line","POLYGON((0 0, 0 48, 84 48, 84 0, 0 0))"

117,"Ashton","POLYGON((62 48, 84 48, 84 30, 56 30, 56 34, 62 48))"

118,"Goose Island","POLYGON((67 13, 67 18, 59 18, 59 13, 67 13))"

119,"Route 75","MULTILINESTRING((10 48, 10 21, 10 0), (16 0, 10 23, 16 48))"

120,"Stock Pond","MULTIPOLYGON(((24 44, 22 42, 24 40, 24 44)), ((26 44, 26 40,28 42, 26 44)))"

4 Conformance Items

This section details the tests that are to be executed. Each test constitutes a Conformance Item in the terminology of the Conformance Testing Program document. The ID in the following tables is used to reference the specific Conformance Item:

4.1 Feature

ID	Functionality Tested	Description		Result
	Interfaces	***Methods***	***Required***	
F1	Feature	feature_type, get_geometry, property_exists, get_property, set_property, delete_property, get_property_sequence, get_property_iterator, destroy,	mandatory	A "passed" or "Failed" message is displayed at the end of test.
F2	FeaturePropertySet	next, next_n, destroy, reset	mandatory	
F3	FeatureFactory	create_feature, create_features	mandatory	
F4	FeatureType	name, property_def_exists, get_property_def, get_property_def_sequence, get_property_def_iterator, destroy	mandatory	
F5		get_parents, get_children	option	
F6	FeatureTypeFactory	create	option	
F7	PropertyDefIterator	next, next_n, destroy, reset	mandatory	
F8	FeatureCollection	number_features, add_element, merge, insert_element_at, replace_element_at, remove_element_at, remove_all_elements, create_iterator,	mandatory	
F9		feature_type, get_geometry, property_exists, get_property, set_property, delete_property, get_property_sequence, get_property_iterator, destroy,	mandatory	
F10	FeatureCollectionFactory	create, createFromCollection, createFromSequence	mandatory	
F11	FeatureIterator	next, next_n, advance, current, get_geometry, get_property, get_property_sequence, get_property_iterator, reset, more, destroy,	mandatory	
F12	ContainerFeatureCollection	number_features, add_element, merge, insert_element_at, replace_element_at, remove_element_at, remove_all_elements, create_iterator,	mandatory	
F13		create_feature, create_features	mandatory	
F14		feature_type, get_geometry, property_exists, get_property, set_property, delete_property, get_properties_sequence, get_property_iterator, destroy,	mandatory	
F15	ContainerFeatureCollectionFactory	create, createFromCollection, createFromSequence, createFromFeatureData	mandatory	
F16	QueryEvaluator	ql_types, default_ql_type,	option	
F17		evaluate, query_by_properties	option	

F18	QueryableFeatureCollection	ql_types, default_ql_type	option	
F19		evaluate, query_by_properties	option	
F20		number_features, add_element, merge, insert_element_at, replace_element_at, remove_element_at, remove_all_elements, create_iterator,	option	
F21		feature_type, get_geometry, property_exists, get_property, set_property, delete_property, get_property_sequence, get_property_iterator, destroy,	option	
F22	QueryableContainerFeatureCollection	ql_types, default_ql_type	option	
F23		evaluate, query_by_properties	option	
F24		number_features, add_element, merge, insert_element_at, replace_element_at, remove_element_at, remove_all_elements, create_iterator,	option	
F25		create_feature, create_features	option	
F26		feature_type, get_geometry, property_exists, get_property, set_property, delete_property, get_property_sequence, get_property_iterator, destroy,	option	
F27	QueryResultSetIterator	get_metadata, advance, get_record, get_record_as_feature, get_property_by_index, get_property_by_name, get_string_by_index, get_string_by_name, get_float_by_index, get_float_by_name, get_double_by_index, get_double_by_name, get_long_by_index, get_long_by_name, get_short_by_index, get_short_by_name, get_boolean_by_index, get_boolean_by_name, get_decimal_by_index, get_decimal_by_name, get_byte_stream_by_index, get_byte_stream_by_name, get_geometry_by_index, get_geometry_by_name, destroy	option	
F28	QueryResultSetMetaData	get_property_count, get_schema_name, get_property_name, get_property_type, get_property_type_name, is_read_only, is_writable, is_case_sensitive, is_searchable, is_signed, is_currency	option	

4.2 Spatial Reference System

ID	Functionality Tested	Description		Result
S1	SpatialReferenceSyst	***Interfaces***	***Methods***	

	em Methods	AngularUnit	name, authority, code, alias, abbreviation, remarks, well_known_text, radians_per_unit	A “passed” or “Failed” message is displayed at the end of test.
		LinearUnit	name, authority, code, alias, abbreviation, remarks, well_known_text, meters_per_unit	
		Ellipsoid	name, authority, code, alias, abbreviation, remarks, well_known_text, semi_major_axis, semi_minor_axis, inverse_flattening, axis_unit	
		HorizontalDatum	name, authority, code, alias, abbreviation, remarks, well_known_text, base_ellipsoid	
		PrimeMeridian	name, authority, code, alias, abbreviation, remarks, well_known_text, longitude, angular_units	
		SpatialReferenceSystem	name, authority, code, alias, abbreviation, remarks, well_known_text	
		GeodeticSpatialReferenceSyst em	name, authority, code, alias, abbreviation, remarks, well_known_text	
		GeograhicCoordinateSystem	name, authority, code, alias, abbreviation, remarks, well_known_text, usage, horizontal_datum, angular_unit, prime_meridian	
		Parameter	name, authority, code, alias, abbreviation, remarks, well_known_text, units, value	
		ParamterList	name, authority, code, alias, abbreviation, remarks, well_known_text, number_parameters, get_default_parameters, set_parameters, get_parameters	

		Projection	name, authority, code, alias, abbreviation, remarks, well_known_text, usage, classification, forward, inverse, parameters, angular_units, linear_units, base_ellipsoid	
		ProjectedCoordinateSystem	name, authority, code, alias, abbreviation, remarks, well_known_text, usage, geographic_coordinate_system, linear_units, base_projection, parameters, forward, inverse	
S2	GeographicTransform Interface (option)	GeographicTransform	name, authority, code, alias, abbreviation, remarks, well_known_text, source_gcs, target_gcs, forward, inverse,	
S3	SpatialReferenceSystemFactory Interface	A SpatialReferenceSystem is created from WKT by using creat_from_WKT method.		
S4	SpatialReferenceComponentFactory Interface	Each SpatialReferenceComponent are created from EPSG ID using following methods. authority, create_projected_coordinate_system, create_geographic_coordinate_system, create_projection, create_geographic_transform, create_horizontal_datum, create_ellipsoid, create_prime_meridian, create_linear_unit, create_angular_unit		A "passed" or "Failed" message is displayed at the end of test.

4.3 Geometry

ID	**Functionality Tested**	**Description**		**Result**
G1	Geometry Interface (Mandatory)	The following methods are tested:		A "passed" or "Failed" message is displayed at the end of test.
		Interfaces	***Methods***	
		Point	copy, dimension, range_envelope, spatial_reference_system, is_empty, is_simple, is_closed, export, destroy, coordinates	

		MultiPoint	copy, dimension, range_envelope, spatial_reference_system, is_empty, is_simple, is_closed, export, destroy, number_elements, add_element, merge, insert_element_at, replace_element_at, remove_element_at, remove_all_elements, retrieve_element_at, create_iterator	test.
		LineString	copy, dimension, range_envelope, spatial_reference_system, is_empty, is_simple, is_closed, export, destroy, num_points, get_point_by_index, get_point_by_index_as_WKS, set_point_by_index, set_point_by_index_as_WKS, insert_point_by_index, insert_point_by_index_as_WKS, append_point, append_point_with_WKS, delete_point_by_index, length, start_point, end_point, start_point_as_WKS, end_point_as_WKS, is_planar, value, value_as_WKS	
		MultiLineString	copy, dimension, range_envelope, spatial_reference_system, is_empty, is_simple, is_closed, export, destroy, length, number_elements, add_element, merge, insert_element_at, replace_element_at, remove_element_at, remove_all_elements, retrieve_element_at, create_iterator	
		LinearPolygon	copy, dimension, range_envelope, spatial_reference_system, is_empty, is_simple, is_closed, export, destroy, area, centroid, centroid_as_WKS, point_on_surface, point_on_surface_as_WKS, is_planar, exterior_ring, exterior_ring_as_WKS, interior_rings, interior_rings_as_WKS	

		MultiLinearPolygon	copy, dimension, range_envelope, spatial_reference_system, is_empty, is_simple, is_closed, export, destroy, area, number_elements, add_element, merge, insert_element_at, replace_element_at, remove_element_at, remove_all_elements, retrieve_element_at, create_iterator	
G2	Geometry Methods (option)	Point	export_WKBGeometry	
		LineString	export_WKBGeometry	
		LinearPolygon	export_WKBGeometry	
		MultiPoint	export_WKBGeometry	
		MultiLineString	export_WKBGeometry	
		MultiLinearPolygon	export_WKBGeometry	
G3	Geometry Methods (option)	<SpatialOperator>	boundary, buffer, covex_hull, distance, difference, intersection, symmetric_difference, union_op	
		<SpatialRelation>	contains, crosses, disjoint, equals, intersects, overlaps, touches, within	
		<SpatialRelation2>	relate	
G4	GeometryFactory Methods (mandatry)	GeometryFactory	create, create_from_WKS	
		PointFactory	create_from_Point, create_from_WKSPoint	
		LineStringFactory	create_from_LineString, create_from_WKSLineString	
		LinearPolygonFactory	create_from_LinearPolygon, create_from_WKSLinearPolygon	
		MultiPointFactory	create_from_MultiPoint, create_from_WKSMultiPoint	
		MultiLineStringFactory	create_from_MultiLineString, create_from_WKSMultiLineString	
		MultiLinearPolygonFactory	create_from_MultiLinearPolygon, create_from_WKSMultiLinearPolygon	
G5	GeometryFactory Methods (option)	GeometryFactory	create_from_WKB	
		PointFactory	create_from_WKBPoint	
		LineStringFactory	create_from_WKBLineString	
		LinearPolygonFactory	create_from_WKBLinearPolygon	
		MultiPointFactory	create_from_WKBMultiPoint	
		MultiLineStringFactory	create_from_WKBLineString	
		MultiLinearPolygonFactory	create_from_WKBMultiLinearPolygon	