Testbed-12 CITE User Guide - Profiles

Table of Contents

Publication Date: 2017-06-16

Approval Date: 2017-03-23

Posted Date: 2016-11-16

Reference number of this document: OGC 16-076r1

Reference URL for this document: http://www.opengis.net/doc/bp/t12-A088

Category: User Guide

Editor: R. Martell

Title: Testbed-12 : CITE User Guide - Profiles

COPYRIGHT

Copyright © 2017 Open Geospatial Consortium. To obtain additional rights of use, visit http://www.opengeospatial.org/

IMPORTANT	Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights. Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.
NOTE	This document is a user guide created as a deliverable of an initiative from the OGC Innovation Program (formerly OGC Interoperability Program). It is not an OGC standard. It is not an official position of the OGC membership. There may be additional valid approaches beyond what is described in this user guide.

POINTS OF CONTACT

Name	Organization
R. Martell (ed.)	Galdos Systems, Inc.
Luis Bermudez	OGC

Chapter 1. Introduction

This document offers guidance about how to develop and execute a test suite that verifies an implementation of some OGC application profile that is based on one or more existing standards. It is primarily intended for developers and other technical professionals who seek to test implementations that claim conformance to an OGC profile. For more general information about how to develop OGC conformance tests, see the TEAM Engine documentation site.

In ISO 19106, a profile is defined as a

> set of one or more base standards or subsets of base standards, and, where applicable, the identification of chosen clauses, classes, options and parameters of those base standards, that are necessary for accomplishing a particular function.
>
> — ISO 19106:2004 cl. 4.5

In practice, a profile imposes additional constraints on an implementation such as mandating support for an optional capability. Or it may define extensions where permitted by the base standards. However, a profile can **never** contradict any base standard, such that conformance to a profile implies conformance to *all* of the base standards from which it is derived.

Various user communities have developed profiles based on OGC or ISO standards. For example:

- DGIWG - Web Feature Service Profile (DGIWG 122)
- DGIWG - Web Map Service Profile (DGIWG 112)
- NSG OGC Web Map Service 1.3 Interoperability Standard (Ed.)

The DGIWG WMS profile is based on the OGC WMS 1.3 specification, and the NSG WMS profile is in turn based on the DGIWG profile. A profile always has one or more dependencies that must be heeded when developing an implementation or a conformance test suite. As indicated in WMS profile dependencies, a test suite for any WMS profile will include tests covering one or more conformance classes in the OGC WMS conformance test suite. The DGIWG profile requires conformance at the "Queryable WMS" level; the associated test suite also includes tests that cover the specific requirements of that profile.

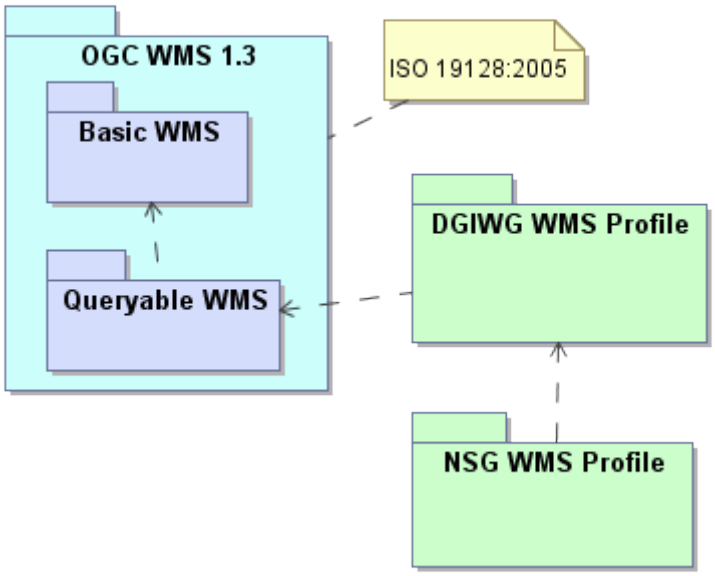


Figure 1. WMS profile dependencies

A conformance class is essentially a container for a coherent set of requirements that address some functional capability or area of concern. A conformance level is a kind of conformance class in the which the requirements of a higher level contain all the requirements of the lower levels (see Conformance classes).

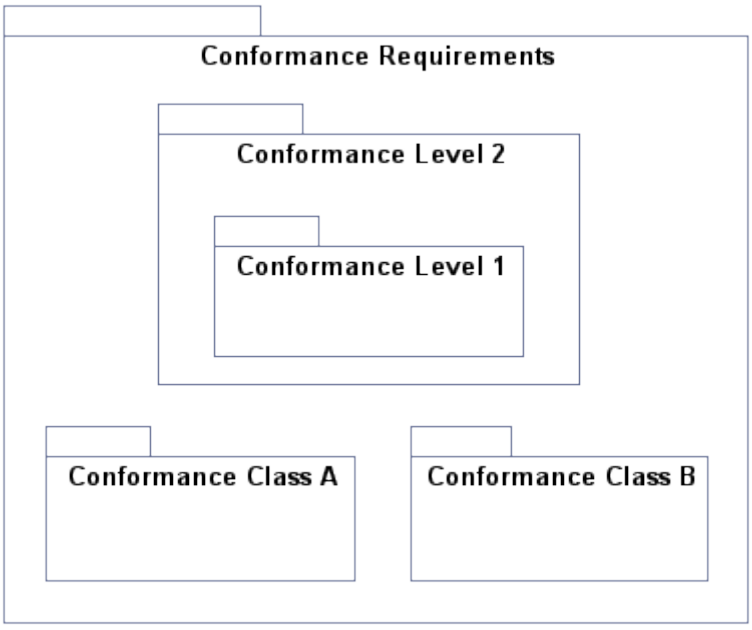
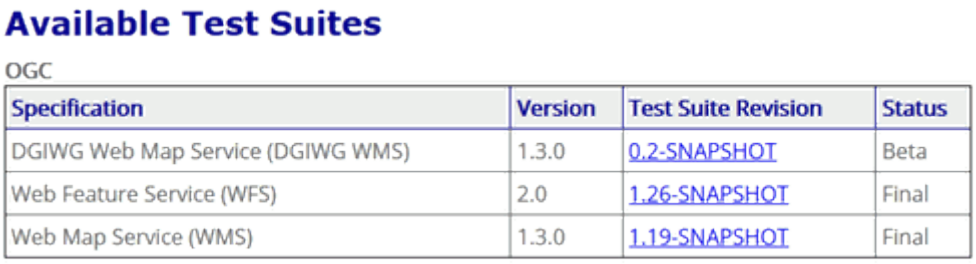


Figure 2. Conformance classes

A test suite for a profile generally selects one or more conformance classes (or levels) from the relevant set of base specifications; these tests then implicitly become part of the dependent test suite. The base tests are invoked in the course of running the profile-specific tests. However, the manner in which this is done depends on how a test suite is implemented.

A test suite that covers the requirements of an application profile is accessed and executed just like any other test suite. It appears in the listing of available suites, and it can be selected and run in the same manner. See the following figure: Profile-specific test suites.



Available Test Suites

OGC

Specification	Version	Test Suite Revision	Status
DGIWG Web Map Service (DGIWG WMS)	1.3.0	0.2-SNAPSHOT	Beta
Web Feature Service (WFS)	2.0	1.26-SNAPSHOT	Final
Web Map Service (WMS)	1.3.0	1.19-SNAPSHOT	Final

Figure 3. Profile-specific test suites

Chapter 2. TestNG

Most of the OGC conformance test suites developed in recent years are based on the TestNG framework. The framework provides a simple declarative mechanism for incorporating tests from other suites in a test suite definition, which is an XML document that conforms to this document type definition.

NOTE	While an XML file (conventionally named testng.xml) is often used to define a test suite, it is also possible to assemble the runtime definition programmatically using the TestNG API. The XmlSuite class is the entry point for doing this (it is correlated with the top-level <suite> tag).

The root <suite> element contains one or more child <test> elements that correspond to conformance classes. These conformance classes may be drawn from other test suites. For example, the WFS 2.0 test suite uses two conformance classes from the GML 3.2 test suite. The following listing shows how this dependency is declared.

Listing 1. Importing conformance classes from other test suites

```
<suite name="wfs20-1.22" verbose="0" configfailurepolicy="continue">
  <parameter name="wfs"  value=""/>
  <parameter name="fid"  value=""/>

  <test name="All GML application schemas">
    <classes>
      <class name="org.opengis.cite.iso19136.general.XMLSchemaTests" />
      <class name="org.opengis.cite.iso19136.general.GeneralSchemaTests" />
      <class name="org.opengis.cite.iso19136.general.ModelAndSyntaxTests" />
      <class name="org.opengis.cite.iso19136.general.ComplexPropertyTests" />
    </classes>
  </test>
  <test name="GML application schemas defining features and feature collections">
    <classes>
      <class name="org.opengis.cite.iso19136.components.FeatureComponentTests" />
    </classes>
  </test>
  <test name="Simple WFS">
    <packages>
      <package name="org.opengis.cite.iso19142.simple" />
    </packages>
  </test>
  <!-- remaining WFS tests -->
</suite>
```

The <test> sets are run in document order by default, which is usually the preferred behavior. If any tests fail in a base conformance class, this can cause profile-specific tests to be skipped since there is already evidence of non-conformance.

Chapter 3. CTL

The OGC Compliance Test Language (CTL) specifies an XML grammar for defining a test suite. Some older OGC test suites were implemented using CTL scripts. In recent years test suites have been developed using the TestNG framework, and this is the recommended approach for new test suites. But some current profiles are based on older OGC standards for which only CTL-based test suites exist.

Tests may be organized into separate packages denoted by the ctl:package element; the main package must contain a ctl:suite element that identifies the starting test. The ctl:profile element may be used to formally define a test group that corresponds to a conformance class. A profile must refer to its base test group (usually the main test suite), and it must also identify the starting test. The sample listing below displays a profile definition for the GET method binding in the WMTS test suite.

Listing 2. Profile for the GET method binding in WMTS 1.0

```
<profile xmlns="http://www.occamlab.com/ctl" name="wmts:server.profile.kvp.get">
  <title>WMTS 1.0 Server Compliance Test Profile for KVP GET binding</title>
  <description>Verifies that a WMTS 1.0 server implementation complies with
conformance classes for KVP GET binding.</description>
  <defaultResult>Pass</defaultResult>
  <base>wmts:server.suite.base</base>
  <starting-test>wmts:server.profile.kvp.get.main</starting-test>
</profile>
```

The <base> element refers to the suite or profile (by name) that this profile depends on. This establishes an implicit ordering, such that the root suite is run first and test execution follows the dependency graph.

Chapter 4. Invoking test suites using a REST API

A RESTful API was introduced in order to provide a common mechanism for invoking a test suite regardless of its implementation. In effect, a test suite becomes a kind of "microservice" that can be accessed as indicated in the table below.

Table 1. Test execution endpoints

Path	Resource	Method(s)	Output format
/rest/suites	List of available test suites	GET	XHTML
/rest/suites/{ets-code}/{ets-version}	Test suite summary	GET	XHTML
/rest/suites/{ets-code}/{ets-version}/run	Test run controller	GET, POST	XML, RDF/XML

The URI parameters are delimited by curly brackets; these are explained below.

ets-code

The ETS code (example: "wfs20")

ets-version

The ETS version (example: "1.25")

The list of available test suites is presented as a brief HTML document (XHTML syntax) that contains links to the deployed test suites. While the document can be displayed in a web browser for human viewers, it can also be consumed and parsed by other software applications in order to facilitate test execution. For example, a service description in a registry could be automatically annotated with information about its conformance status by running a test suite and inspecting the results.

Listing 3. List of deployed test suites (XHTML syntax)

```
<ul>
  <li><a href="suites/wfs20/1.25/" id="wfs20-1.25" type="text/html">WFS 2.0 (ISO
19142:2010) Conformance Test Suite</a></li>
  <li><a href="suites/gml32/1.24/" id="gml32-1.24" type="text/html">GML (ISO
19136:2007) Conformance Test Suite, Version 3.2.1</a></li>
  <!-- other available test suites -->
</ul>
```

When the link for a particular test suite is dereferenced a summary document is obtained. This document briefly describes the test suite and contains a table of test run arguments. Each input argument is a separate entry in the body of an HTML table as shown in the listing below.

Listing 4. Test run arguments for the WFS 2.0 test suite (raw HTML)

```
<tbody>
  <tr id="wfs">
    <td>wfs</td>
    <td>URI</td>
    <td>M</td>
    <td>A URI that refers to a representation of the service capabilities document.
    This document does not need to be obtained from the service under test (SUT),
    but it must describe the SUT. Ampersand ('&amp;') characters appearing within
    a query parameter value must be percent-encoded as %26.</td>
  </tr>
  <tr id="fid">
    <td>fid</td>
    <td>NCName</td>
    <td>O</td>
    <td>An identifier that matches the @gml:id attribute value of an available feature
    instance (may be omitted for "Basic WFS" implementations).</td>
  </tr>
</tbody>
```

Test run arguments

Name	Value domain	Obligation	Description
wfs	URI	M	A URI that refers to a representation of the service capabilities document. This document does not need to be obtained from the service under test (SUT), but it must describe the SUT. Ampersand ('&') characters appearing within a query parameter value must be percent-encoded as %26.
fid	NCName	O	An identifier that matches the @gml:id attribute value of an available feature instance (may be omitted for "Basic WFS" implementations).

Figure 4. Description of test run arguments presented in a web browser

A test run is initiated by submitting a request to the test run controller. The summary description lists the test run arguments (mandatory, conditional, optional) that are recognized by the controller. A test suite can be invoked using a simple GET request in most cases. For example, to test a WFS 2.0 implementation the target URI is constructed as follows (replace localhost:8080 with the actual host name and port number of an available teamengine installation):

```
http://localhost:8080/teamengine/rest/suites/wfs20/1.25/run?wfs={wfs-capabilities-url}
```

where `{wfs-capabilities-url}` is the URL to retrieve the capabilities document for the implementation under test (IUT). Note that this need not be obtained directly from the IUT—it could be fetched from elsewhere (e.g. a service registry), as long as it describes the same service.

With TEAM Engine 4.9 or later it is also possible to invoke a CTL test suite in this manner. However, a controller must be available in order to do this. The WMS 1.3 test suite contains a CtlController class that serves as an example of how to enable this capability in other CTL test suites.

As a concrete example, consider the DGIWG WMS profile which is based on the OGC WMS 1.3 standard (also published as ISO 19128:2005). The profile requires implementation of the **Queryable**

WMS conformance class as defined in the base standard. To verify this, the OGC test suite can be invoked using the REST API by submitting a GET request with the following query parameters (the target URI has been abbreviated to emphasize the query component):

```
/rest/suites/wms/1.19/run?capabilities-url={wms-capabilities-url}&queryable=queryable
```

A successful response contains an XML entity that represents the test results. The root element contains a <log> child element for each test that was run. The first log entry indicates the overall verdict; the value of the endtest/@result attribute is an integer code that signifies a test verdict (see table below).

Table 2. CTL test verdicts

Code	Result
1	Passed
2	Not Tested
3	Skipped
4	Warning
5	Inherited Failure
6	Failed

If a constituent test failed, the overall verdict is set as **Inherited Failure** (5). In general, a failed subtest will "taint" all of its ancestor tests in this manner.

Chapter 5. Using the W3C EARL vocabulary

The default format of the test results is framework-specific: for TestNG, this is an XML representation having <testng-results> as the document element. The results of running a CTL test suite also produce XML output, with <execution> as the document element. Support for the W3C Evaluation and Report Language (EARL) 1.0 Schema has been introduced. The specification (currently a late stage working draft) defines an RDF vocabulary for describing test results:

- Evaluation and Report Language (EARL) 1.0 Schema
- Developer Guide for EARL 1.0
- HTTP Vocabulary in RDF 1.0
- Representing Content in RDF 1.0

The following listing shows how conformance classes are described using the EARL vocabulary. An `earl:TestRequirement` instance represents a conformance class; it has one or more constituent tests (`earl:TestCase`). Furthermore, a dependency may be expressed using the *dct:requires* property. In this example, **Conformance level 2** is based on **Conformance level 1** and thus establishes a higher level of conformance.

Listing 5. Conformance classes in EARL results (RDF/XML)

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:earl="http://www.w3.org/ns/earl#"
         xmlns:dct="http://purl.org/dc/terms/">

  <earl:TestRequirement rdf:about="http://www.opengis.net/spec/KML/2.3/conf/level-1">
    <dct:title xml:lang="en">KML 2.3 - Conformance Level 1</dct:title>
    <dct:description xml:lang="en">Conformance Level 1 includes test cases that
address
    absolute requirements. A KML document must satisfy all assertions at this level to
    achieve minimal conformance</dct:description>
    <dct:isPartOf rdf:resource="http://docs.opengeospatial.org/ts/14-068r2/14-
068r2.html"/>
    <dct:hasPart>
      <earl:TestCase rdf:about="http://www.opengis.net/spec/KML/2.3/conf/level-1/atc-
101">
        <dct:description>Verify that the root element of the document has [local name]
= "kml"
        and [namespace name] = "http://www.opengis.net/kml/2.3".</dct:description>
        <dct:title>Document element</dct:title>
      </earl:TestCase>
    </dct:hasPart>
    <!-- other constituent test cases omitted -->
  </earl:TestRequirement>

  <earl:TestRequirement rdf:about="http://www.opengis.net/spec/KML/2.3/conf/level-2">
    <dct:title xml:lang="en">KML 2.3 - Conformance Level 2</dct:title>
    <dct:description xml:lang="en">Includes all tests in Level 1, plus test cases
covering
    requirements that should be satisfied by a KML document. Non-conformance at this
    level may hinder the utility, portability, or interoperability of the
document.</dct:description>
    <dct:requires rdf:resource="http://www.opengis.net/spec/KML/2.3/conf/level-1"/>
    <!-- constituent test cases omitted -->
  </earl:TestRequirement>

</rd:RDF>
```

The EARL vocabulary does not define any terms that pertain to a test run by itself. A custom vocabulary was introduced for this purpose. A `cite:TestRun` resource provides basic summary information about a test run, including the input arguments and an overall tally of test verdicts. Standard Dublin Core metadata terms are employed where appropriate. For example, the dct:extent property reports the temporal extent of the test run; that is, its total duration represented using the XML Schema duration datatype.

Listing 6. A TestRun resource

```
<cite:TestRun xmlns:cite="http://cite.opengeospatial.org/">
  <dct:extent
rdf:datatype="http://www.w3.org/2001/XMLSchema#duration">PT6M30.204S</dct:extent>
  <dct:title>wfs20-1.25</dct:title>
  <cite:testsSkipped
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</cite:testsSkipped>
  <cite:testsPassed
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">298</cite:testsPassed>
  <cite:testsFailed
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">46</cite:testsFailed>
  <dct:created>2016-10-25T17:33:31.290Z</dct:created>
  <cite:inputs>
    <rdf:Bag>
      <rdf:li rdf:parseType="Resource">
        <dct:title>wfs</dct:title>

<dct:description>http://example.org/services/wfs?service=WFS&amp;request=GetCapabiliti
es</dct:description>
      </rdf:li>
      <rdf:li rdf:parseType="Resource">
        <dct:title>xsd</dct:title>

<dct:description>http://example.org/services/wfs?service=WFS&amp;version=2.0.0&amp;req
uest=DescribeFeatureType</dct:description>
      </rdf:li>
    </rdf:Bag>
  </cite:inputs>
  <dct:identifier>8ed93bd8-b366-4d4f-b868-c8e5aeccfbaa</dct:identifier>
</cite:TestRun>
```