Open Geospatial Consortium

Approval Date: 2013-01-18

Posted Date: 2013-01-13

Publication Date: 2013-06-18

Reference number of this document: OGC 12-152r1

Reference URL for this document: http://www.opengis.net/def/doc-type/per/ows9-cite-guide

Category: Public Engineering Report

Editors: Luis Bermudez

OGC® OWS-9 CITE Help Guide Engineering Report

Warning

This document is not an OGC Standard. This document presents a discussion of technology issues considered in an initiative of the OGC Interoperability Program. This document does not represent an official position of the OGC. It is subject to change without notice and may not be referred to as an OGC Standard. However, the discussions in this document could very well lead to the definition of an OGC Standard.

Document type:	OGC® Engineering Report
Document subtype:	NA
Document stage:	Approved for public release
Document language:	English

Abstract

This Engineering Report was created as a deliverable of the CITE Thread as part of the OGC Web Services, Phase 9 (OWS-9) initiative of the OGC Interoperability Program. This OGC® Engineering Report describes how to execute the CITE tests, how to select the conformance classes, how to access and download the TEAM Engine, Test Scripts and Reference Implementations, how to insert your own data and/or schemas along with a section that provides other community help resources.

This is not a normative document.

Keywords

ogcdoc, OGC document, cite, testing, TEAM, ows9, ows-9

What is OGC Web Services 9 (OWS-9)?

OWS-9 builds on the outcomes of prior OGC interoperability initiatives and is organized around the following threads:

- **Aviation**: Develop and demonstrate the use of the Aeronautical Information Exchange Model (AIXM) and the Weather Exchange Model (WXXM) in an OGC Web Services environment, focusing on support for several Single European Sky ATM Research (SESAR) project requirements as well as FAA (US Federal Aviation Administration) Aeronautical Information Management (AIM) and Aircraft Access to SWIM (System Wide Information Management) (AAtS) requirements.

- **Cross-Community Interoperability (CCI)**: Build on the CCI work accomplished in OWS–8 by increasing interoperability within communities sharing geospatial data, focusing on semantic mediation, query results delivery, data provenance and quality and Single Point of Entry Global Gazetteer.

- **Security and Services Interoperability (SSI)**: Investigate 5 main activities: Security Management, OGC Geography Markup Language (GML) Encoding Standard Application Schema UGAS (UML to GML Application Schema) Updates, Web Services Façade, Reference Architecture Profiling, and Bulk Data Transfer.

- **OWS Innovations**: Explore topics that represent either new areas of work for the Consortium (such as GPS and Mobile Applications), a desire for new approaches to existing technologies to solve new challenges (such as the OGC Web Coverage Service (WCS) work), or some combination of the two.

- **Compliance & Interoperability Testing & Evaluation (CITE)**: Develop a suite of compliance test scripts for testing and validation of products with interfaces implementing the following OGC standards: Web Map Service (WMS) 1.3 Interface Standard, Web Feature Service (WFS) 2.0 Interface Standard, Geography Markup Language (GML) 3.2.1 Encoding Standard, OWS Context 1.0 (candidate encoding standard), Sensor Web Enablement (SWE) standards, Web Coverage Service for Earth Observation (WCS-EO) 1.0 Interface Standard, and TEAM (Test, Evaluation, And Measurement) Engine Capabilities.

The OWS-9 sponsors are: AGC (Army Geospatial Center, US Army Corps of Engineers), CREAF-GeoViQua-EC, EUROCONTROL, FAA (US Federal Aviation Administration), GeoConnections - Natural Resources Canada, Lockheed Martin Corporation, NASA (US National Aeronautics and Space Administration), NGA (US National Geospatial-Intelligence Agency), USGS (US Geological Survey), UK DSTL (UK MoD Defence Science and Technology Laboratory).

License Agreement

Contents

Page

Figures

Page

OGC® OWS-9 CITE Help Guide Engineering Report

1 Introduction

1.1 Scope

This OGC® Engineering Report describes how to execute the CITE tests, how to select the conformance classes, how to access and download the TEAM Engine, Test Scripts and Reference Implementations, how to insert your own data and/or schemas along with a section that provides other community help resources.

All questions regarding this document should be directed to the editor or the contributors:

Name	Organization
Luis Bermudez (LB)	OGC
David Wesloh (DW)	NGA

1.2 Revision history

Date	Release	Editor	Primary clauses modified	Description
2012.11.30	1	LB	All	Initial Document
2012.12.20	2	DW	ALL	Comments on the sections an minor editing
2012.12.26	3	DW	ALL	Added more content and figures to sections an minor editing.
2013.01.13	4	LB	Minor	Minor edits and removed comments and tracked changes.

1.3 Future work

1.4 Foreword

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.

2 References

The following documents are referenced in this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

Compliance Test Language (CTL) Best OGC 06-126
Practicehttp://portal.opengeospatial.org/files/?artifact_id=33085

CITE Policies and Procedures: OGC 08-134r9
http://portal.opengeospatial.org/files/49237

3 Background

3.1 OGC CITE program

The OGC Compliance (CITE) Program provides the resources, procedures, and policies for improving software implementations' compliance with OGC standards. The Compliance Program provides an online free testing facility, a process for certification and branding of compliant products, and coordination of a vibrant community that develops and supports test scripts. The Compliance Program also runs plugfests, which are short term events for increasing interoperability among vendors' products.

3.2 Composition of a test suite and conformance classes

A Compliance Test Package is composed of an Abstract Test Suite (ATS) and an Executable Test Suite (ETS). An ATS is a set of testable assertions about the functionality of a standard, which an implementation must support in order to achieve compliance to the standard. An ATS is based on the conformance classes defined in the standard in accordance with the latest revision of “OGC 08-131r3 The Specification Model — A Standard for Modular specifications”. An ETS is a set of code (e.g. Java and CTL) that provides runtime tests for the assertions defined by the ATS. Test data required to do the tests are part of the ETS.

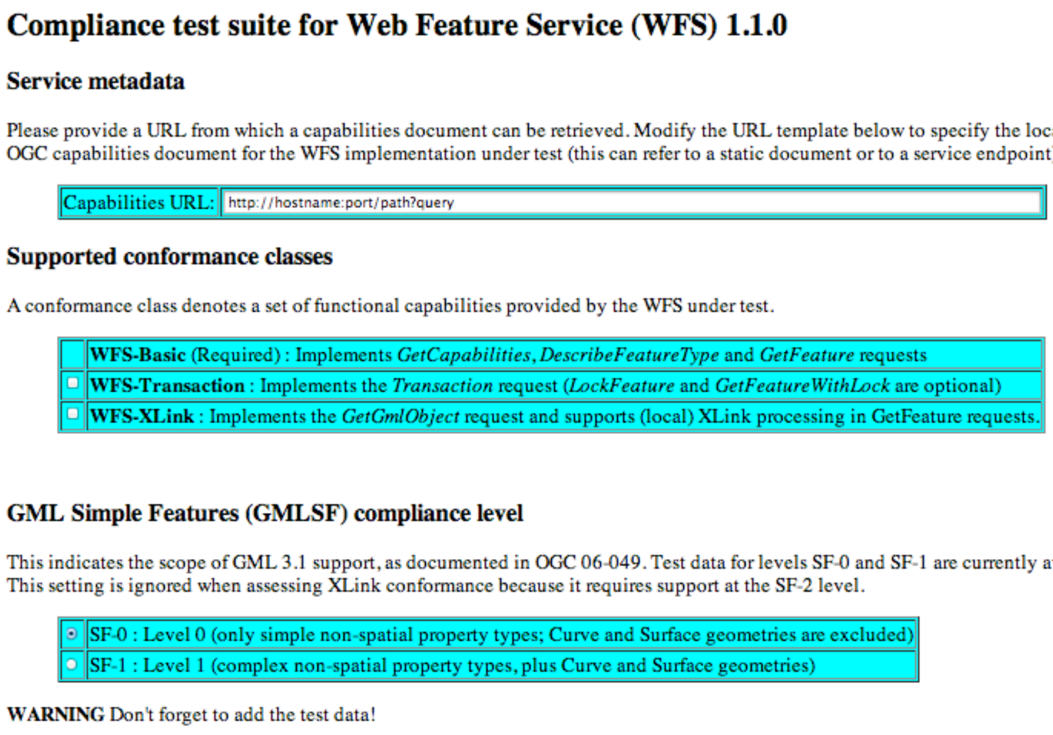
3.3 Introduction to TEAM Engine

The OGC's accepted Compliance Test Engine is an open source product called TEAM Engine. TEAM Engine (Test, Evaluation, And Measurement Engine) is a software program for testing web services, clients and encodings. It executes test scripts written in Compliance Test Language (CTL), TestNG and other languages. It is lightweight and easy to run as a command line or to setup as a service. It can be used to test any type of service or encoding, not only OGC. It is also the official tool used by the Open Geospatial Consortium (OGC) for compliance testing. It helps verify that a software implementation is conformant with a standard in order for the implementation to get certified. Once the software implementation gets certified, the organization owner of the software can use the compliance OGC logo in marketing materials.

Figure 1 – OGC Compliant logo.

3.4 Test scripts

The Test Scripts composition is explained in 3.2. It is a set of code that implements the ATS. The tests can be organized as a set of conformance classes or it can be presented with options that the user running the test can select. Figure 2 shows an example of possible selections for WFS 1.1.0.



Compliance test suite for Web Feature Service (WFS) 1.1.0

Service metadata

Please provide a URL from which a capabilities document can be retrieved. Modify the URL template below to specify the location of an OGC capabilities document for the WFS implementation under test (this can refer to a static document or to a service endpoint).

Capabilities URL: http://hostname:port/path?query

Supported conformance classes

A conformance class denotes a set of functional capabilities provided by the WFS under test.

	WFS-Basic (Required) : Implements *GetCapabilities*, *DescribeFeatureType* and *GetFeature* requests
□	**WFS-Transaction** : Implements the *Transaction* request (*LockFeature* and *GetFeatureWithLock* are optional)
□	**WFS-XLink** : Implements the *GetGmlObject* request and supports (local) XLink processing in GetFeature requests.

GML Simple Features (GMLSF) compliance level

This indicates the scope of GML 3.1 support, as documented in OGC 06-049. Test data for levels SF-0 and SF-1 are currently available. This setting is ignored when assessing XLink conformance because it requires support at the SF-2 level.

●	SF-0 : Level 0 (only simple non-spatial property types; Curve and Surface geometries are excluded)
○	SF-1 : Level 1 (complex non-spatial property types, plus Curve and Surface geometries)

WARNING Don't forget to add the test data!

Figure 2 – Example of Test Suite Options

The Figure shows that there are three conformance classes in the test: WFS-Basic, WFS-Transaction and WFS-Link. Note that the basic level cannot be pre-selected because is the required conformance class to get WFS 1.1.0 certification. The user can also select the level of GML.

Tests that ran in TEAM Engine can be encoded in CTL and TestNG. These are describe in the following sections.

3.4.1 CTL

Compliance Test Language or CTL is an XML grammar, based on XSLT, for documenting and scripting suites of tests for verifying that an implementation of a specification complies with the specification. A more complete description of CTL is available as an OGC Discussion Paper (OGC Document 06-126).

A test suite in CTL consists of a set of objects. The initial object is a suite, which identifies a starting test and may also include a form with instructions. The starting test is called using the values the user enters in the form as parameters and contains instructions that may call other tests or use functions and parsers.

A suite may optionally be extended by a profile to test functionality that is not mandatory in the base test suite. A profile identifies the base test suite it is extending and a starting profile test. When a profile is executed, the starting tests from both the base test suite and the profile are called, and both are passed the values from the base test suite's form as parameters. See Figure 3.

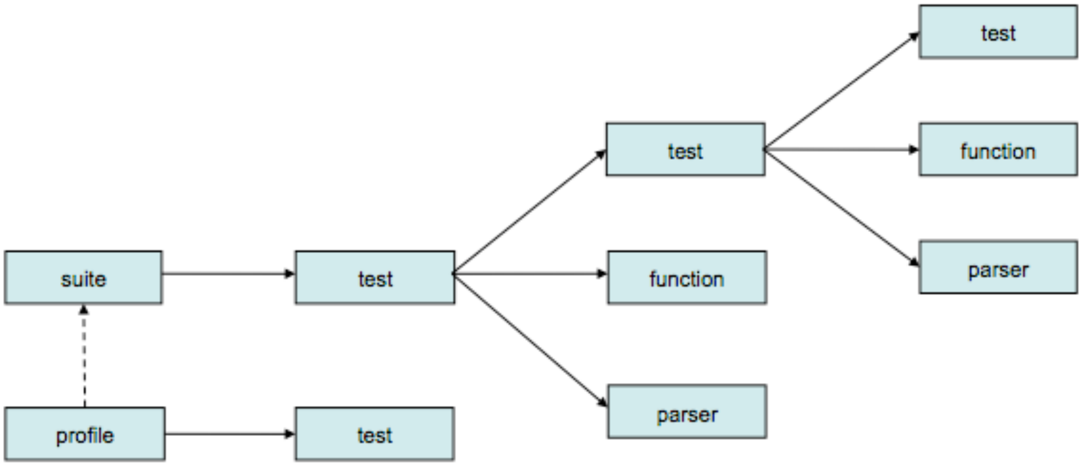


Figure 3 – Structure CTL script

A CTL file is an XML file that contains a CTL object or a package element as its root element. The package element is a container for multiple CTL objects. Each CTL object is identified by a unique, namespace qualified, name, so the set of objects in a test suite may span several files. Test objects contain programmatic code that consists of XSL instructions and/or CTL instructions. Some parser (e.g. SOAP) objects are built-in to CTL, and may be used without declaring a parser object.

3.4.2 TestNG

(From the http://testng.org/doc/index.html)

TestNG is a testing framework based on JUnit and NUnit. It provides the following new functionalities:

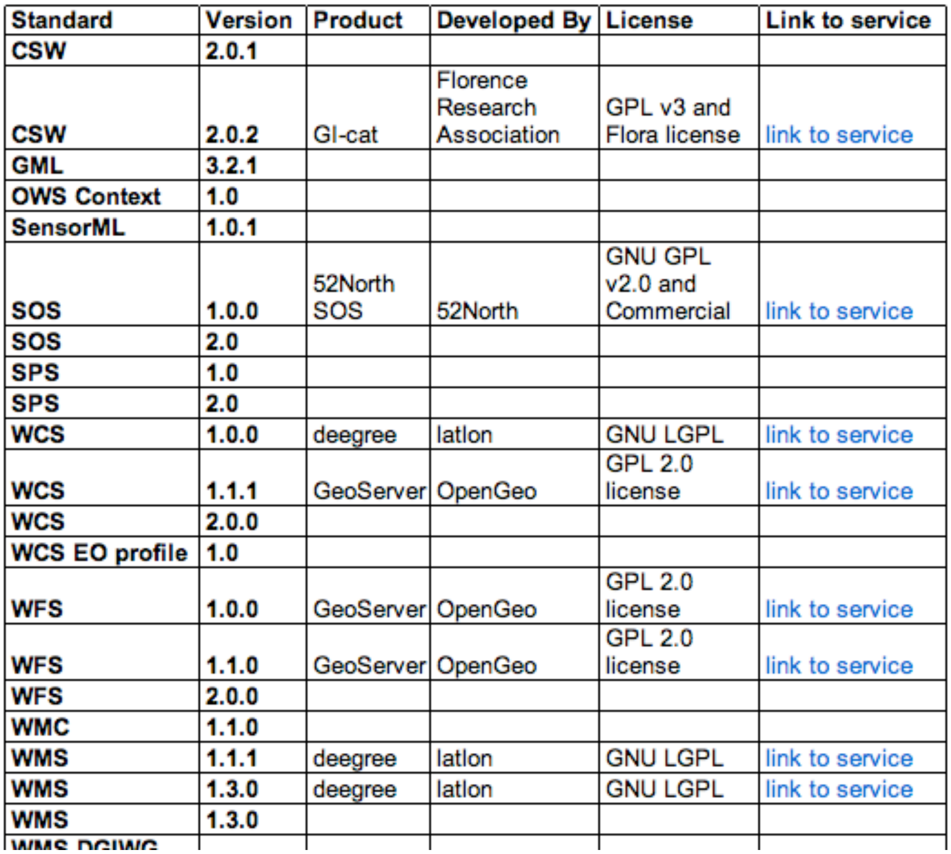
- Use of Annotations.
- Allow running tests in in arbitrarily big thread pools with various policies available (all methods in their own thread, one thread per test class).
- Tests are multithread safe.
- Support for data-driven testing.
- Support for parameters.

- Powerful execution model (no more TestSuite).
- Supported by a variety of tools and plug-ins (Eclipse, IDEA, Maven, etc.).
- Embeds BeanShell for further flexibility.
- Default JDK functions for runtime and logging (no dependencies).
- Dependent methods for application server testing.

3.5 Reference Implementations

A Reference Implementation (OGC 08-134r9) is a fully functional, licensed copy of a tested, branded software that has passes the test for an associated conformance class in a version of an Implementation Standard. It is free and publicly available for testing as a web service or via download. The Reference Implementation does not need to pass all the conformance classes within the standard. In most of the cases, the Reference Implementation will pass at least the core and possibly some number of extension conformance classes. Multiple reference implementations can exist for an associated version of an Implementation Standard. The coordinator after reviewing the results and checking the public interface of the software will determine if the implementation can be a reference implementation.

OGC will maintain a page that explains the scope of each reference implementation. Currently it is located at: http://cite.opengeospatial.org/reference. OGC will also provide this information in the public data base at the main OGC portal.



Standard	Version	Product	Developed By	License	Link to service
CSW	2.0.1				
CSW	2.0.2	GI-cat	Florence Research Association	GPL v3 and Flora license	link to service
GML	3.2.1				
OWS Context	1.0				
SensorML	1.0.1				
SOS	1.0.0	52North SOS	52North	GNU GPL v2.0 and Commercial	link to service
SOS	2.0				
SPS	1.0				
SPS	2.0				
WCS	1.0.0	deegree	latlon	GNU LGPL	link to service
WCS	1.1.1	GeoServer	OpenGeo	GPL 2.0 license	link to service
WCS	2.0.0				
WCS EO profile	1.0				
WFS	1.0.0	GeoServer	OpenGeo	GPL 2.0 license	link to service
WFS	1.1.0	GeoServer	OpenGeo	GPL 2.0 license	link to service
WFS	2.0.0				
WMC	1.1.0				
WMS	1.1.1	deegree	latlon	GNU LGPL	link to service
WMS	1.3.0	deegree	latlon	GNU LGPL	link to service
WMS	1.3.0				
WMS DGIWG					

Figure 4 – List of Reference Implementations as of Dec 2012

As of December 2012, all the listed reference implementations, fully implement all the tests. Meaning that the implementation passes all the advertised tests in TEAM Engine.

OGC will make its best effort to host Reference Implementations on an OGC server to help others in the community to develop compliant implementations.

4 How to setup TEAM Engine

TEAM Engine can be setup to run via a web server or be invoked via command line. For developers invoking TEAM Engine via command line is very convenient since the user can very fast test changes in the implementation code. Setting it up via a Web Service can help communities making available a central web test facility to test standards of interest to the community.

4.1 Requirements

TEAM engine requires JAVA 1.6. For web installation it is recommended a dedicated TOMCAT server. The installation server at OGC is as follows:

- 3GHz (4 core) Xeon processor
- 2 GB RAM
- Debian Linux
- 3 GB of data storage for 800 sessions. Each session is a test run stored as uncompressed xml files.

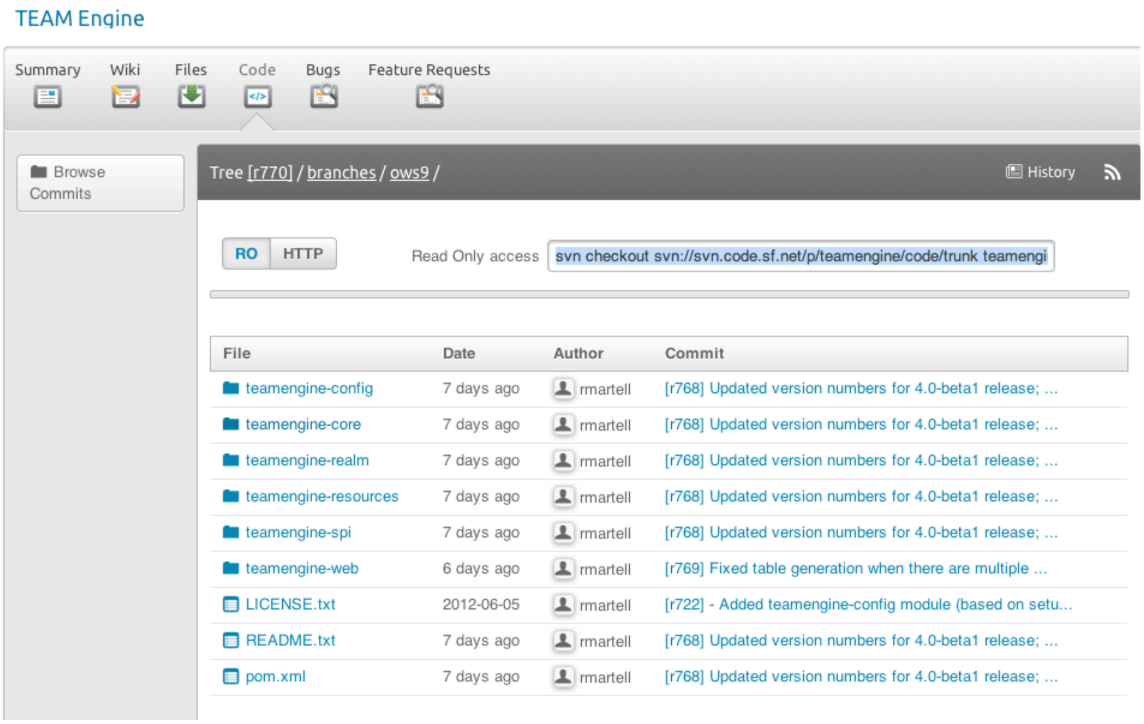
4.2 Setting up via MAVEN

TEAM Engine is available via MAVEN, which is easy for developers to integrate and easy to build. MAVEN is a software project management tool that helps to integrate distributed code. MAVEN provides a common framework to build and release code. Via MAVEN the executable TEAM Engine code is created. Apache Maven 3.0 is required to build the teamengine code base. MAVEN is supported by most Integration Development Environments (IDE) tools such as Eclipse. MAVEN can be downloaded at: http://maven.apache.org

TEAM Engine code is available at Sourceforge and can be downloaded via SVN http://sourceforge.net/projects/teamengine/.

The code is available when selecting the **Code** option or typing the following

http://sourceforge.net/p/teamengine/code/



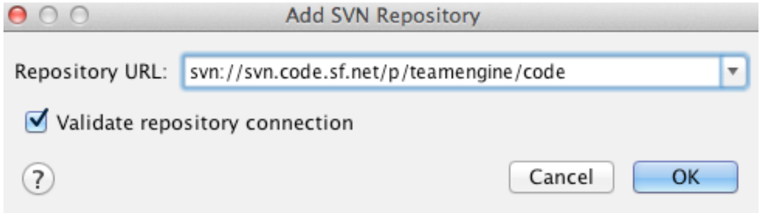
Tree [r770] / branches / ows9 /

Read Only access svn checkout svn://svn.code.sf.net/p/teamengine/code/trunk teamengi

File	Date	Author	Commit
teamengine-config	7 days ago	rmartell	[r768] Updated version numbers for 4.0-beta1 release; ...
teamengine-core	7 days ago	rmartell	[r768] Updated version numbers for 4.0-beta1 release; ...
teamengine-realm	7 days ago	rmartell	[r768] Updated version numbers for 4.0-beta1 release; ...
teamengine-resources	7 days ago	rmartell	[r768] Updated version numbers for 4.0-beta1 release; ...
teamengine-spi	7 days ago	rmartell	[r768] Updated version numbers for 4.0-beta1 release; ...
teamengine-web	6 days ago	rmartell	[r769] Fixed table generation when there are multiple ...
LICENSE.txt	2012-06-05	rmartell	[r722] - Added teamengine-config module (based on setu...
README.txt	7 days ago	rmartell	[r768] Updated version numbers for 4.0-beta1 release; ...
pom.xml	7 days ago	rmartell	[r768] Updated version numbers for 4.0-beta1 release; ...

To connect via an SVN client you can use the following URL

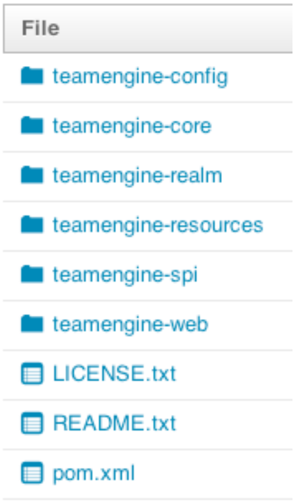
svn://svn.code.sf.net/p/teamengine/code/trunk



Repository URL: svn://svn.code.sf.net/p/teamengine/code

☑ Validate repository connection

When opening the root the following folder and files will be shown.



File

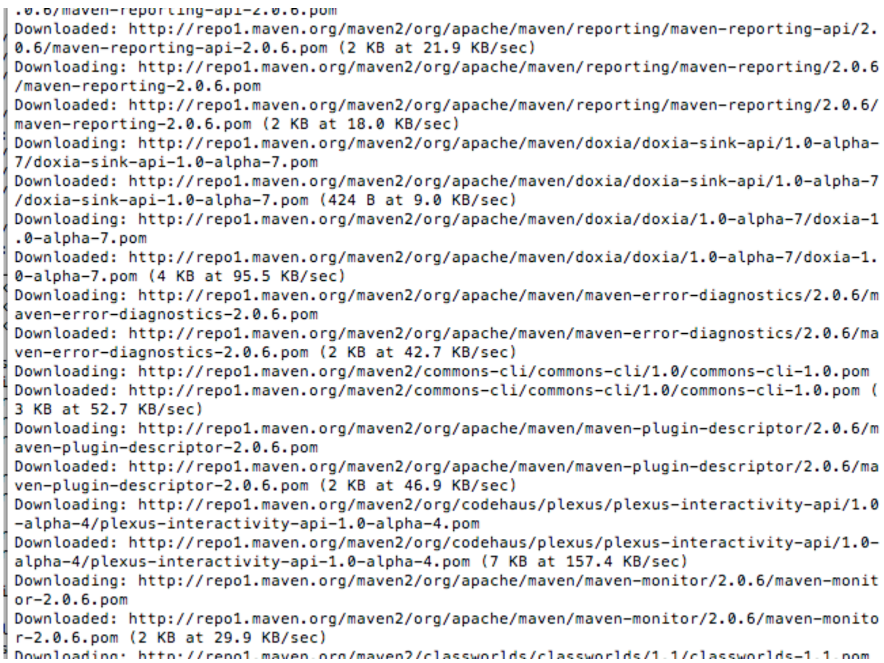
- teamengine-config
- teamengine-core
- teamengine-realm
- teamengine-resources
- teamengine-spi
- teamengine-web
- LICENSE.txt
- README.txt
- pom.xml

Figure 5 – TEAM Engine Code structure at Sourceforge

TEAM Engine consists in the following MAVEN modules:

- teamengine-core: Main CTL script processor
- teamengine-web: Web interface
- teamengine-resources: Shared resources (stylesheets, schemas, etc.)
- teamengine-config: Configuration utility
- teamengine-realm: Custom Tomcat user realm
- teamengine-spi: Extensibility framework and REST API

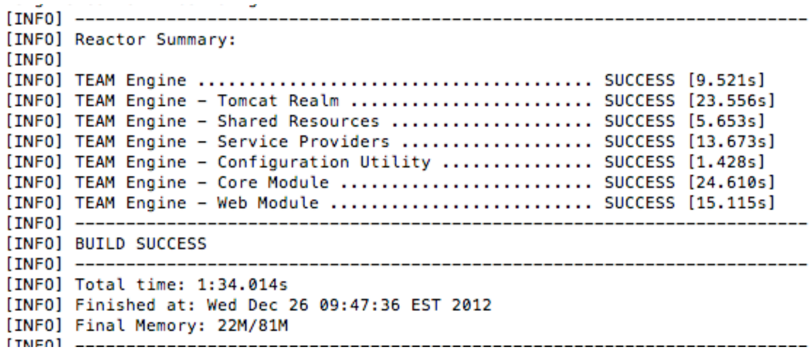
Once downloaded, simply run **mvn package** in the root project directory to generate all build artifacts, or mvn install to add the artifacts to the local repository. When installing, all needed code is downloaded from various MAVEN repositories. Depending on the Internet connection this process my take 1-5 minutes.



```
.0.6/maven-reporting-api-2.0.6.pom
Downloaded: http://repo1.maven.org/maven2/org/apache/maven/reporting/maven-reporting-api/2.
0.6/maven-reporting-api-2.0.6.pom (2 KB at 21.9 KB/sec)
Downloading: http://repo1.maven.org/maven2/org/apache/maven/reporting/maven-reporting/2.0.6
/maven-reporting-2.0.6.pom
Downloaded: http://repo1.maven.org/maven2/org/apache/maven/reporting/maven-reporting/2.0.6/
maven-reporting-2.0.6.pom (2 KB at 18.0 KB/sec)
Downloading: http://repo1.maven.org/maven2/org/apache/maven/doxia/doxia-sink-api/1.0-alpha-
7/doxia-sink-api-1.0-alpha-7.pom
Downloaded: http://repo1.maven.org/maven2/org/apache/maven/doxia/doxia-sink-api/1.0-alpha-7
/doxia-sink-api-1.0-alpha-7.pom (424 B at 9.0 KB/sec)
Downloading: http://repo1.maven.org/maven2/org/apache/maven/doxia/doxia/1.0-alpha-7/doxia-1
.0-alpha-7.pom
Downloaded: http://repo1.maven.org/maven2/org/apache/maven/doxia/doxia/1.0-alpha-7/doxia-1.
0-alpha-7.pom (4 KB at 95.5 KB/sec)
Downloading: http://repo1.maven.org/maven2/org/apache/maven/maven-error-diagnostics/2.0.6/m
aven-error-diagnostics-2.0.6.pom
Downloaded: http://repo1.maven.org/maven2/org/apache/maven/maven-error-diagnostics/2.0.6/ma
ven-error-diagnostics-2.0.6.pom (2 KB at 42.7 KB/sec)
Downloading: http://repo1.maven.org/maven2/commons-cli/commons-cli/1.0/commons-cli-1.0.pom
Downloaded: http://repo1.maven.org/maven2/commons-cli/commons-cli/1.0/commons-cli-1.0.pom (
3 KB at 52.7 KB/sec)
Downloading: http://repo1.maven.org/maven2/org/apache/maven/maven-plugin-descriptor/2.0.6/m
aven-plugin-descriptor-2.0.6.pom
Downloaded: http://repo1.maven.org/maven2/org/apache/maven/maven-plugin-descriptor/2.0.6/ma
ven-plugin-descriptor-2.0.6.pom (2 KB at 46.9 KB/sec)
Downloading: http://repo1.maven.org/maven2/org/codehaus/plexus/plexus-interactivity-api/1.0
-alpha-4/plexus-interactivity-api-1.0-alpha-4.pom
Downloaded: http://repo1.maven.org/maven2/org/codehaus/plexus/plexus-interactivity-api/1.0-
alpha-4/plexus-interactivity-api-1.0-alpha-4.pom (7 KB at 157.4 KB/sec)
Downloading: http://repo1.maven.org/maven2/org/apache/maven/maven-monitor/2.0.6/maven-monit
or-2.0.6.pom
Downloaded: http://repo1.maven.org/maven2/org/apache/maven/maven-monitor/2.0.6/maven-monito
r-2.0.6.pom (2 KB at 29.9 KB/sec)
Downloading: http://repo1.maven.org/maven2/classworlds/classworlds/1.1/classworlds-1.1.pom
```

Figure 6 – Example MAVEN artifacts being downloaded

Once all the coded is downloaded you should see a BUILD SUCCESS message as shown in Figure 7.



```
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary:
[INFO]
[INFO] TEAM Engine ....................................... SUCCESS [9.521s]
[INFO] TEAM Engine - Tomcat Realm ........................ SUCCESS [23.556s]
[INFO] TEAM Engine - Shared Resources .................... SUCCESS [5.653s]
[INFO] TEAM Engine - Service Providers ................... SUCCESS [13.673s]
[INFO] TEAM Engine - Configuration Utility ............... SUCCESS [1.428s]
[INFO] TEAM Engine - Core Module ......................... SUCCESS [24.610s]
[INFO] TEAM Engine - Web Module .......................... SUCCESS [15.115s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 1:34.014s
[INFO] Finished at: Wed Dec 26 09:47:36 EST 2012
[INFO] Final Memory: 22M/81M
[INFO] ------------------------------------------------------------------------
```

Figure 7 – MAVEN Build Success Message

The main build artifacts include:

teamengine-core-${version}-distribution.zip

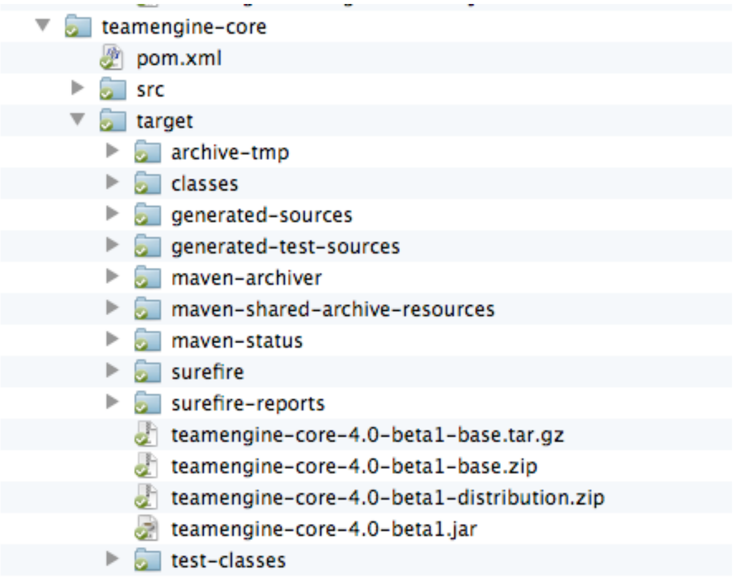
The core binary distribution for command line usage.

teamengine-core-${version}-base.[zip | tar.gz]

Content of the main configuration directory (TE_BASE).

teamengine.war
   The JEE web (Servlet) application.
teamengine-common-libs.[zip | tar.gz]
   Common runtime dependencies (e.g. JAX-RS 1.1).

The generated files (after running mvn package) are found under the target folder. For example under the teamengine-core the were various directories and zip files created as shown in Figure 8.



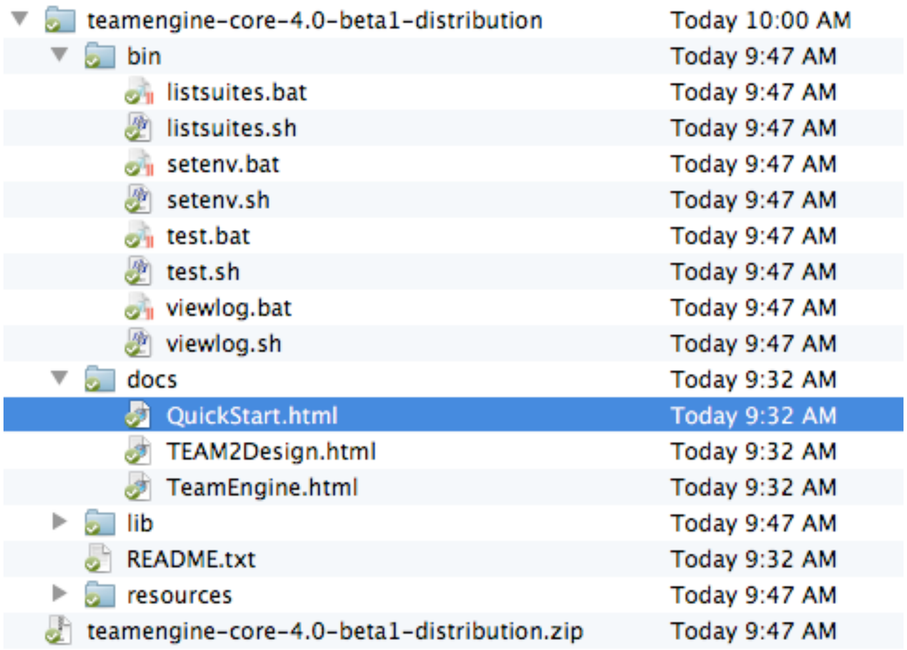
```
teamengine-core
    pom.xml
    src
    target
        archive-tmp
        classes
        generated-sources
        generated-test-sources
        maven-archiver
        maven-shared-archive-resources
        maven-status
        surefire
        surefire-reports
        teamengine-core-4.0-beta1-base.tar.gz
        teamengine-core-4.0-beta1-base.zip
        teamengine-core-4.0-beta1-distribution.zip
        teamengine-core-4.0-beta1.jar
        test-classes
```

Figure 8 – teamengine-core build

4.3 Installation and Configuration

To run via command line you need the teamengine-core-xxx-distribution.zip generated in the previous step (See Figure 8).

Unzip the file (e.g. teamengine-core-4.0-beta1-distribution.zip) and you will be ready to run team engine. The unzip directory should like Figure 9



```
▼ teamengine-core-4.0-beta1-distribution          Today 10:00 AM
  ▼ bin                                            Today 9:47 AM
      listsuites.bat                               Today 9:47 AM
      listsuites.sh                                Today 9:47 AM
      setenv.bat                                   Today 9:47 AM
      setenv.sh                                    Today 9:47 AM
      test.bat                                     Today 9:47 AM
      test.sh                                      Today 9:47 AM
      viewlog.bat                                  Today 9:47 AM
      viewlog.sh                                   Today 9:47 AM
  ▼ docs                                           Today 9:32 AM
      QuickStart.html                              Today 9:32 AM
      TEAM2Design.html                             Today 9:32 AM
      TeamEngine.html                              Today 9:32 AM
  ▶ lib                                            Today 9:47 AM
    README.txt                                     Today 9:32 AM
  ▶ resources                                      Today 9:47 AM
  teamengine-core-4.0-beta1-distribution.zip       Today 9:47 AM
```

Figure 9 – teamengine-core-distribution unzipped

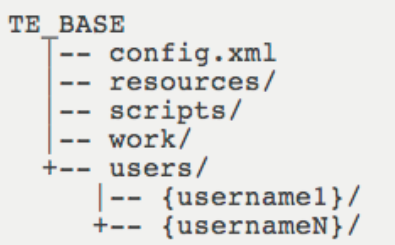
The bin folder contains all the command executable. The docs folder contains documentation, and the lib and resources folders contain the code needed to run TEAM Engine via command line.

The next step is to setup a directory where all the tests, session and temporary files will be stored. This is called TE_BASE (Figure 10). The value of the TE_BASE system property or environment variable specifies the location of the main configuration directory that contains several essential sub-directories. The process includes:

□ Creation of a TE_BASE directory in the server (e.g. /srv/teamengine)

□ Unpacking the contents of the teamengine-core-*-base archive into this location.

□ Ensuring that users (including the Tomcat user) have write access.



```
TE_BASE
  |-- config.xml
  |-- resources/
  |-- scripts/
  |-- work/
  +-- users/
      |-- {username1}/
      +-- {usernameN}/
```

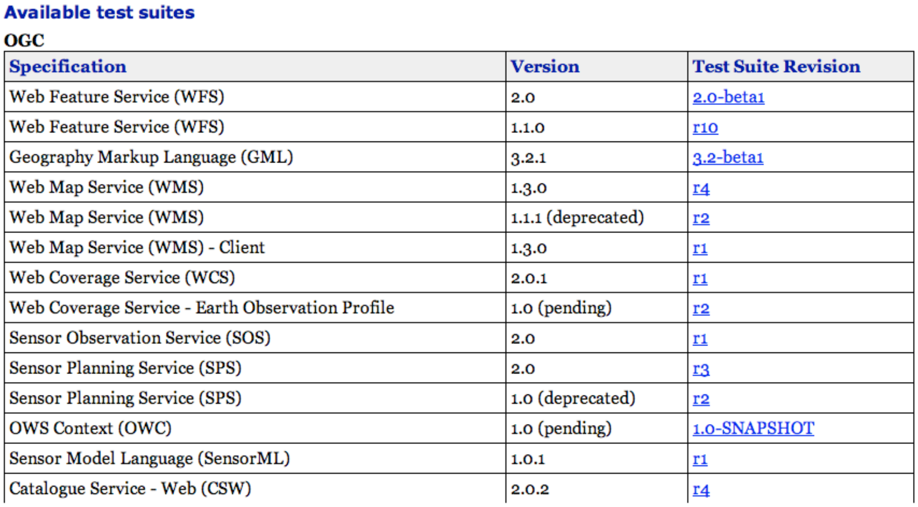
Figure 10 – TE BASE directory

CTL test scripts are placed in the TE_BASE/scripts directory and the TE_BASE/config.xml file is updated accordingly.

The scripts are available at the OGC SVN repository, which has anonymous read access

https://svn.opengeospatial.org/ogc-projects/cite/scripts/

Figure 11 shows the tests available by Dec 2012 in the TEAM Engine beta 4.



Available test suites

OGC

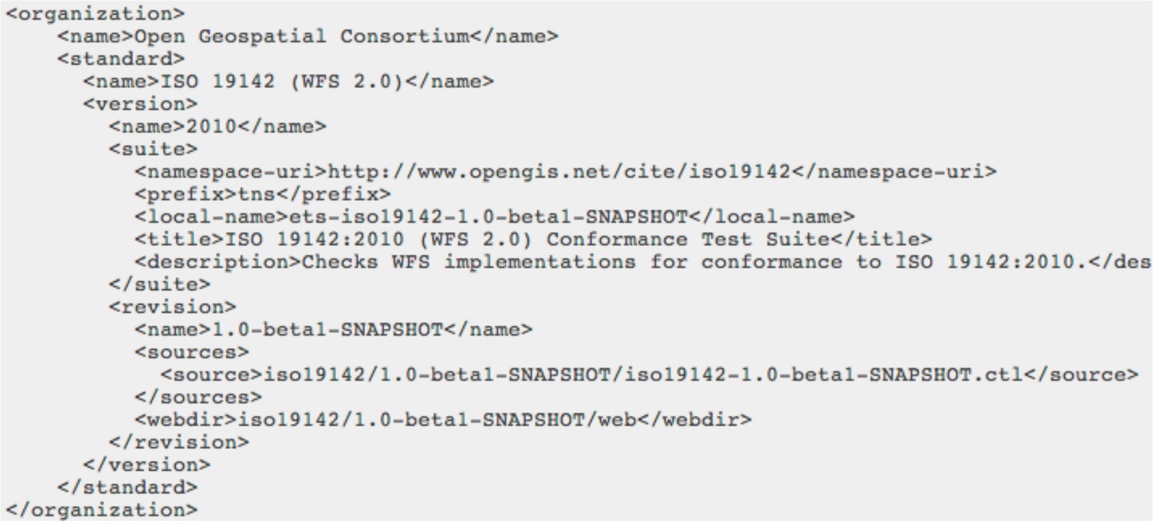
Specification	Version	Test Suite Revision
Web Feature Service (WFS)	2.0	2.0-beta1
Web Feature Service (WFS)	1.1.0	r10
Geography Markup Language (GML)	3.2.1	3.2-beta1
Web Map Service (WMS)	1.3.0	r4
Web Map Service (WMS)	1.1.1 (deprecated)	r2
Web Map Service (WMS) - Client	1.3.0	r1
Web Coverage Service (WCS)	2.0.1	r1
Web Coverage Service - Earth Observation Profile	1.0 (pending)	r2
Sensor Observation Service (SOS)	2.0	r1
Sensor Planning Service (SPS)	2.0	r3
Sensor Planning Service (SPS)	1.0 (deprecated)	r2
OWS Context (OWC)	1.0 (pending)	1.0-SNAPSHOT
Sensor Model Language (SensorML)	1.0.1	r1
Catalogue Service - Web (CSW)	2.0.2	r4

Figure 11 – Test Suites with revisions available at TEAM Engine Beta 4 (stage 2)

The table shows the specification name, the version and the test suite revision. For example, WFS 1.1.0 has had 10 revisions. To run the latest revision of WFS 1.1.0 the test that needs to be downloaded and copied into the TE_BASE/scripts is: https://svn.opengeospatial.org/ogc-projects/cite/scripts/wfs/1.1.0/tags/r10/

4.4 Template Configuration File for New Tests

A new TEAM Engine installation contains an entry for a sample test script ("XML Note"). Every OGC test suite should include a configuration file. If a new test suite is to be create this file can be copied and edited. An entry for the WFS 2.0 test suite is shown below.



```
<organization>
    <name>Open Geospatial Consortium</name>
    <standard>
      <name>ISO 19142 (WFS 2.0)</name>
      <version>
        <name>2010</name>
        <suite>
          <namespace-uri>http://www.opengis.net/cite/iso19142</namespace-uri>
          <prefix>tns</prefix>
          <local-name>ets-iso19142-1.0-beta1-SNAPSHOT</local-name>
          <title>ISO 19142:2010 (WFS 2.0) Conformance Test Suite</title>
          <description>Checks WFS implementations for conformance to ISO 19142:2010.</des
        </suite>
        <revision>
          <name>1.0-beta1-SNAPSHOT</name>
          <sources>
            <source>iso19142/1.0-beta1-SNAPSHOT/iso19142-1.0-beta1-SNAPSHOT.ctl</source>
          </sources>
          <webdir>iso19142/1.0-beta1-SNAPSHOT/web</webdir>
        </revision>
      </version>
    </standard>
</organization>
```

4.5 Running as a web server

Apache Tomcat 7.0 (with JDK 6 or later) is a supported servlet container. It is strongly recommended that a dedicated Tomcat instance be created to host the teamengine application. Create one as suggested below.

4.5.1 Create a CATALINA_BASE directory

Under Windows do the following:

```
> mkdir base-1 & cd base-1
> xcopy %CATALINA_HOME%\conf conf\
> mkdir lib logs temp webapps work
```

Under GNU/Linux do the following

```
$ sudo mkdir -p /srv/tomcat/base-1; cd /srv/tomcat/base-1
$ sudo cp -r $CATALINA_HOME/conf .
$ sudo mkdir lib logs temp webapps work
```

4.5.2 Configure Java Virtual Machine

These JVM options are recommended for the Tomcat instance:

```
CATALINA_OPTS="-server -Xmx1024m -XX:MaxPermSize=128m -DTE_BASE=$TE_BASE -
Dderby.system.home=$DERBY_DATA"
```

4.5.3 Create contexts

Unpack the contents of the teamengine-common-libs archive into the CATALINA_BASE/lib directory. Add the following contexts for static web content to `CATALINA_BASE/conf/Catalina/localhost/`.

docs.xml

```
<Context path="/docs" docBase="${TE_BASE}/resources/docs" />
```

reports.xml

```
<Context path="/reports" docBase="${TE_BASE}/users" />
```

about.xml

```
<Context path="/about" docBase="${TE_BASE}/scripts" />
```

4.5.4 Deploy the teamengine.war component and start the Tomcat instance.

Once the war is installed, start Tomcat. The following URIs provide starting points for discovering and executing test suites:

- /teamengine - Home page for selecting and running CTL test suites
- /teamengine/rest/suites - Shows a listing of available (TestNG) test suites. See CiteTestNG for details.

5 Downloading OGC Test Scripts

5.1 Accessing and installing OGC Tests

OGC Tests are available via OGC SVN:

https://svn.opengeospatial.org/ogc-projects/cite/scripts/

Download the scripts you want to test and copy them under the scripts directory (See section 4.3). Then follow the instructions in 4.3.

5.2 Running OGC Test via command line

Once TEAM Engine and tests are correctly installed, following instructions in section 4.2 and 4.3, the command line can be used to run the tests. For example:

```
C:\teamengine2.0>bin\test -source=hello.ctl
```

6 How to test profiles

As of December 2012 only profiles related to GML can be tested. The test verifies conformance of GML data and application schemas with respect to ISO 19136:2007 (GML 3.2.1). As shown in Figure 12 , a conforming GML data instance must refer to the relevant GML application schema, which in turn imports the complete GML schema.

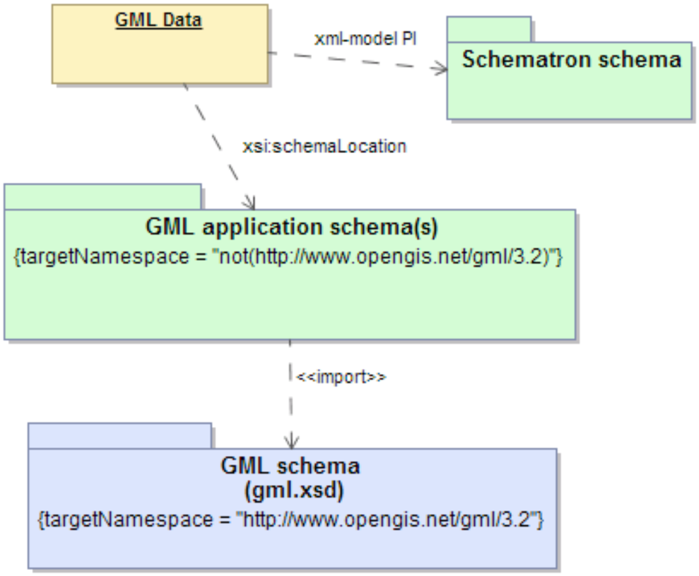
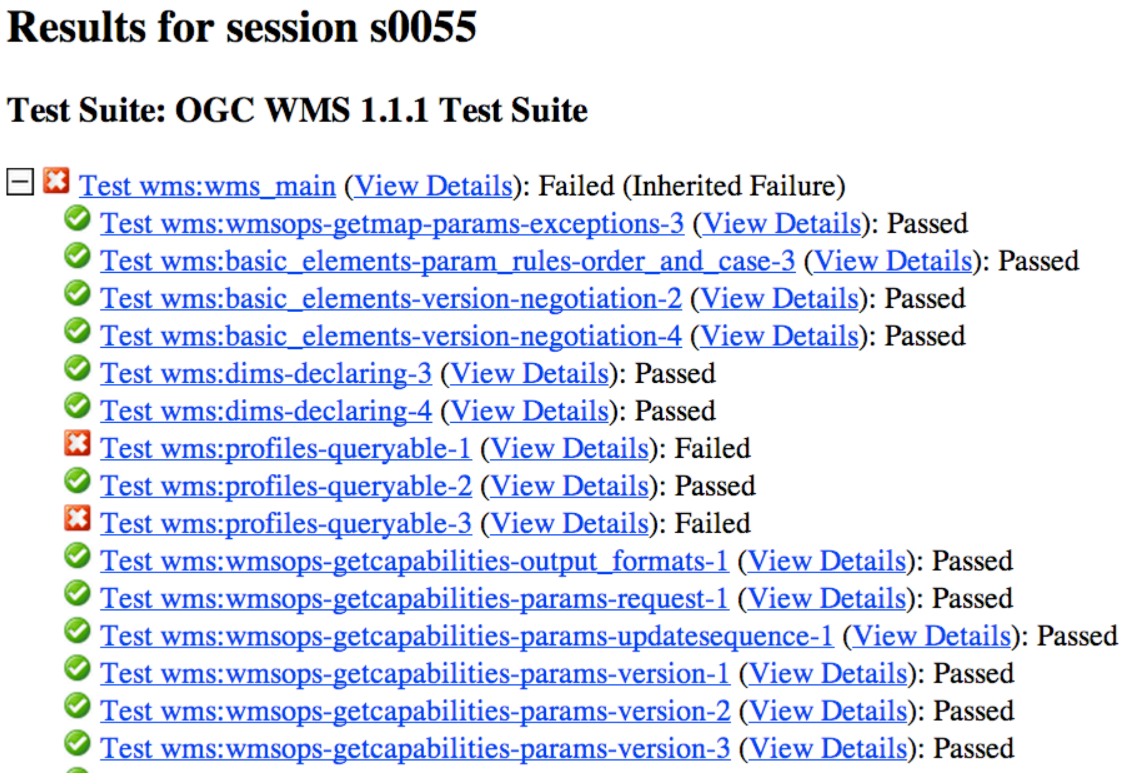

GML Data
xml-model PI
Schematron schema
xsi:schemaLocation
GML application schema(s)
{targetNamespace = "not(http://www.opengis.net/gml/3.2)"}
<<import>>
GML schema
(gml.xsd)
{targetNamespace = "http://www.opengis.net/gml/3.2"}

Figure 12 – Scope GML 3.2 Test

7 How to test your own data

Test Suites to test encodings will be available in 2013. This will include GML, SensorML and KML. The procedure is the same as testing a server. TEAM Engine will provide n HTML form to capture the information about the location of the file. Then the file will be uploaded to the engine and tested.

8 How to access test reports



Results for session s0055

Test Suite: OGC WMS 1.1.1 Test Suite

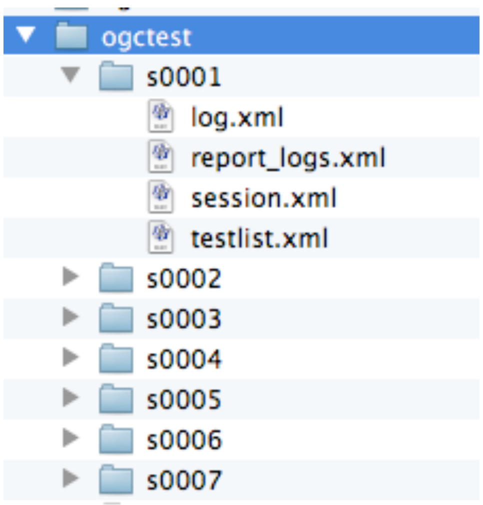
⊟ ☒ Test wms:wms_main (View Details): Failed (Inherited Failure)

- ✓ Test wms:wmsops-getmap-params-exceptions-3 (View Details): Passed
- ✓ Test wms:basic_elements-param_rules-order_and_case-3 (View Details): Passed
- ✓ Test wms:basic_elements-version-negotiation-2 (View Details): Passed
- ✓ Test wms:basic_elements-version-negotiation-4 (View Details): Passed
- ✓ Test wms:dims-declaring-3 (View Details): Passed
- ✓ Test wms:dims-declaring-4 (View Details): Passed
- ☒ Test wms:profiles-queryable-1 (View Details): Failed
- ✓ Test wms:profiles-queryable-2 (View Details): Passed
- ☒ Test wms:profiles-queryable-3 (View Details): Failed
- ✓ Test wms:wmsops-getcapabilities-output_formats-1 (View Details): Passed
- ✓ Test wms:wmsops-getcapabilities-params-request-1 (View Details): Passed
- ✓ Test wms:wmsops-getcapabilities-params-updatesequence-1 (View Details): Passed
- ✓ Test wms:wmsops-getcapabilities-params-version-1 (View Details): Passed
- ✓ Test wms:wmsops-getcapabilities-params-version-2 (View Details): Passed
- ✓ Test wms:wmsops-getcapabilities-params-version-3 (View Details): Passed

Figure 13 – Test Report

The example test report shows the result of a OWS WMS 1.1.1 test run. It shows that the main test failed, because sub-tests failed (e.g. Test wms:profiles-queryable-1). The user has the option to view the details of what was tested en each step.

The reports are stored under the **users** folder in **TE_BASE.** An example of the structure of the folder for the user **ogctest** is presented in Figure 14.



```
▼ ogctest
    ▼ s0001
        log.xml
        report_logs.xml
        session.xml
        testlist.xml
    ▶ s0002
    ▶ s0003
    ▶ s0004
    ▶ s0005
    ▶ s0006
    ▶ s0007
```

Figure 14 - User Test Folder

For each session there is folder created. Under each folder there are xml files that provide the information of the run. The log.xml has information about the test that was run, information about the implementation (e.g. URL of capabilities document) and contains an xml tag endtest, where if result =0, it means it passed the test.

<endtest result="0"/>.

9 Support and additional help resources

There are various resources provided by the CITE program to support testers, the TEAM Engine community and test developers. These are as follows:

9.1.1 Wiki

The wiki provides public information and its being updated constantly with information about how better use TEAM Engine. Link is here:

http://cite.opengeospatial.org

9.1.2 CITE Forum

The CITE forum link is as follows:

https://lists.opengeospatial.org/mailman/listinfo/cite-forum

The cite-forum public mailing list provides CITE and OGC compliant software developers the means to discuss issues and solutions related to OGC tests. This list is used for:

- Discussions of problems found when using TEAM Engine or the test scripts (for example apparently inexplicable failing tests)

- Discussions of new features for TEAM Engine

- Discussions of new tests enhancements

- Announcing of new beta and production releases of TEAM Engine

- Submission of bugs when using TEAM Engine or test scripts

The CITE Forum is supported by Test Leads, which are individuals from organizations that have an agreement with OGC to maintain the test. This agreement is via in-kind contribution, as an exchange for membership or paid via the OGC interoperability program activities. The Leads make their best effort to respond to any posted message within 24 hours.