OGC® DOCUMENT: 26-015R1

External identifier of this OGC® document: http://www.opengis.net/doc/guides/t21-D302

Open Geospatial Consortium

OGC TESTBED-21 DOCUMENTATION FOR THE GIMI CONFORMANCE TEST TOOL

USER GUIDE

PUBLISHED

Version: 0.9
Submission Date: 2026-05-08
Approval Date: 2026-06-04
Publication Date: 29026-08-31
Editor: Charles Heazel

Note

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.

CONTENTS

I ABSTRACT

This document presents a user guide for the GIMI 1.0 Conformance Test Tool, an Executable Test Suite (ETS) developed to verify implementations of the GEOINT Imagery Media for ISR (GIMI) 1.0 Standard. The document is not an OGC Standard, but a Testbed deliverable intended to support review, testing and implementation of the GIMI standard.

There are four major sections to this document.

1. Test Procedure: The Test Procedures section provides an overview of the test suite and instructions on how to execute it.
2. How to interpret the results: The Interpreting Results section provides a description of how the results are presented and what they represent.
3. Architecture: A general understanding of the architecture of both the GIMI format and the Executable Test Suite is valuable when interpreting the test results.
4. Installing the Test Suite: A general set of instructions for installing the Test Suite. More detailed instructions are provided in the Annexes.

II EXECUTIVE SUMMARY

GIMI integrates multiple advanced media standards, including ISO Base Media File Format (ISOBMFF) and the High Efficiency Image File Format (HEIF), to support still imagery, motion imagery, tiled imagery, audio, metadata, security markings, timing information, and content identification. Due to this breadth of capabilities, the availability of reliable conformance testing tools is critical to achieving interoperable implementations.

The GIMI 1.0 Executable Test Suite addresses this need through a structured, automated conformance testing workflow that combines industry-recognized media conformance testing with GIMI-specific validation. The test suite operates in three stages: verification of underlying MPEG/ISO standards using an external conformance tool; parsing of the GIMI box-based file structure into an accessible internal representation; and execution of requirement-level conformance tests implemented using the TestNG framework.

A distinguishing feature of the GIMI test approach is that conformance class selection is driven by the content of the file under test rather than user choice. Whenever content governed by a given group of requirements is present, that class is tested automatically. The test suite supports five primary conformance classes, covering audio content, security markings, HEIF and ISO compliance, metadata, and time-varying media.

The User Guide also provides guidance on installation, deployment, and integration, including Docker-based deployment and integration with the OGC TEAM Engine test harness. Architectural sections describe how the ETS processes imagery essence and metadata, manages extensibility, and supports future enhancements through modular design and helper applications.

Looking forward, the document emphasizes extensibility as a core design principle. Drawing on lessons from earlier imagery standards that evolved significantly over time, the GIMI Test Suite is designed to grow alongside the GIMI Standard itself. While some conformance areas remain under development, the framework provides a solid foundation for improving interoperability, supporting early adoption, and informing future OGC standardization efforts.

KEYWORDS

The following are keywords to be used by search engines and document catalogues.

conformance, gimi, team engine, compliance, testing

IV SECURITY CONSIDERATIONS

No security considerations have been made for this document.

V SUBMITTING ORGANIZATIONS

The following organizations submitted this Document to the Open Geospatial Consortium (OGC):

- Heazel Technologies
- Pixalytics
- Geomatys

VI CONTRIBUTORS

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

NAME	ORGANIZATION	ROLE
Charles Heazel	Heazel Technologies	Editor
Samantha Lavender	Pixalytics	Contributor
Martin Desruisseaux	Geomatys	Contributor
Gobe Hobona	OGC	Contributor

1 SCOPE

1 SCOPE

This User Guide describes how to use the GIMI 1.0 Executable Test Suite (ETS). The purpose of the ETS is to enable users to verify whether software products correctly implement the GIMI 1.0 Standard, which is specified in NGA document NGA.STND.0076-01_V1.0.0_GIMI. The User Guide provides guidance on installation, deployment, usage and integration of the ETS.

The primary focus of the User Guide is the ETS. For further information regarding the GIMI standard, the reader is referred to GIMI specification and the associated GIMI Handbook.

2 NORMATIVE REFERENCES

2 NORMATIVE REFERENCES

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

National Geospatial-Intelligence Agency: NGA.STND.0076-01_V1.0.0_GIMI — GEOINT Imagery Media for Intelligence, Surveillance, and Reconnaissance (ISR) (GIMI)*, 2024-10-23, https://nsgreg.nga.mil/doc/view?i=5650.

ISO/IEC 14496-12:2021 Information Technology — Coding of audio-visual objects — Part 12: ISO base media file format.

ISO/IEC 23001-17:2023 Information Technology — MPEG systems technologies — Part 17: Uncompressed video and images in ISO Base Media File Format.

ISO/IEC 23008-12:2022 Information Technology — MPEG systems technologies — Part 12: Image File Format.

For the purposes of this document, the terms and definitions given in ISO/IEC 14496-12. In addition, ISO and IEC maintain terminological databases for use in standardization at the following URIs:

1. ISO Online browsing platform: available at https://www.iso.org/obp.
2. IEC Electropedia: available at http://www.electropedia.org/.

3

GIMI 1.0 EXECUTABLE TEST SUITE

3 GIMI 1.0 EXECUTABLE TEST SUITE

3.1. Overview

The GIMI 1.0 Executable Test Suite (ETS) has been implemented as a module for the open source TEAM Engine test harness.

3.1.1. Organization

The Executable Test Suite has four stages:

1) MPEG conformance testing

The GIMI Standard is built on a collection of existing industry Standards. These standards are maintained by the Moving Picture Expert Group (MPEG), Motion Imagery Standards Board (MISB), the Joint Photographic Experts Group (JPEG), and the International Organization for Standardization (ISO). Since the OGC does not have authority to define conformance for these standards, the GIMI test-suite addresses these requirements using a widely accepted conformance test tool. ComplianceWarden was selected to fill that role.

2) GIMI File Parser

The first step in processing a GIMI file is to convert it from its' native box structure into an in-memory representation of that structure. This task is performed initially by the SIS GeoHEIF code from Apache. Since this code does not address all of the GIMI box types, it is augmented by ETS-specific parsing code. It is expected that this code will be incorporated, in some form, into SIS in the future.

3) Information Extraction

The GIMI format is built from boxes. A box can contain other boxes, or data, but not both. This results in a complex data model where it can be complicated to locate and extract a specific item of information. The GIMI ETS parser flattens that format into a Java class model. In addition, it pre-processes information elements, making them easily accessible by the conformance tests.

4) GIMI Tests

The GIMI test-suite uses TestNG as its testing language. TestNG is a Java based software testing framework. The GIMI test-suite consists of one package for each conformance class, and one test method for each requirement. Since the required information has been pre-processed by the GIMI parser, the test methods can focus on the requirements.

3.1.2. Conformance Classes

The GIMI 1.0 Standard is organized into five Conformance Classes. These are:

- Core — Foundational requirements including ISO and MPEG conformance
- Audio — Audio media
- GEO1 — IC/DoD Security markings, etc.
- Metadata — Still media and static content
- Movie — Time-varying media

The GIMI Standard is self describing. The file header contains a set of “brands”. Each Conformance Class corresponds to a brand (the opposite is not the case). If the content governed by a Conformance Class (brand) is present in any form, then conformance to that Conformance Class is required. As a result, this test suite does not allow the user to select Conformance classes. If the content and brand are present, then the Conformance Class will be tested.

3.2. How to run the tests

Running the GIMI test suite is very simple. Testers have control over one parameter; the name or URI of the file to be tested. Once the test is completed, the test results are displayed for the tester.

The options for running the suite are summarized below in the following sections.

3.2.1. Integrated development environment (IDE)

Use a Java IDE such as Eclipse, NetBeans, or IntelliJ. Clone the repository and build the project. Set the main class to run: `org.opengis.cite.gimi1.TestNGController`

Arguments: The first argument must refer to an XML properties file containing the required test run arguments. If not specified, the default location at `${user.home}/test-run-props.xml` will be used.

You can modify the sample file in `src/main/config/test-run-props.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties version="1.0">
    <comment>Sample test run arguments</comment>
    <entry key="iut">file:///path/sample-image.heif</entry>
```

```
</properties>
```

Listing 1

The TestNG results file (`testng-results.xml`) will be written to a subdirectory in `${user.home}/testng/` having a UUID value as its name.

3.2.2. Command shell (console)

One of the build artifacts is an "all-in-one" JAR file that includes the test suite and all of its dependencies; this makes it very easy to execute the test suite in a command shell:

```
java -jar ets-gimi-1-0.2-SNAPSHOT-aio.jar  [-o|--outputDir $TMPDIR] [xml-file]
```

Listing 2

Where `xml-file` is the path to the properties XML file, e.g., `src/main/config/test-run-props.xml`.

The example below declares the `TE_BASE` environment variable, the output folder for the test report, and the input XML file that references the GIMI file to be tested.

```
java -jar -DTE_BASE=/cli-testing/te_base  ets-gimi-1-0.2-SNAPSHOT-aio.jar  -o /cli-testing/output  /cli-testing/test-run-props.xml
```

Listing 3

3.2.3. OGC test harness

Use TEAM Engine, the official OGC test harness. The latest Beta versions of executable test suites are usually available at the beta testing facility. However, by the end of Testbed-21 the GIMI ETS was still at Alpha stage, therefore the ETS was only available from OGC's Gitlab Server. It is envisaged that after validation and approval by the Sponsor, the ETS can be moved to the Beta testing facility or a similar environment. This may occur in subsequent Testbeds. Nonetheless, a user can also [build and deploy](https://github.com/opengeospatial/teamengine) the test harness and use a local installation.

4 INTERPRETING RESULTS

4 INTERPRETING RESULTS

The GIMI ETS provides a visual representation of the test results. These results are color-coded as follows:

Green = the test was successful. No additional information is supplied.

Grey = the test was skipped. If the pre-conditions for a test have not been met, then that test is considered not applicable and is skipped. Reporting for these cases includes the text of the requirement skipped and any additional information as to why it was skipped.

Red = the test failed. In this case the output includes the text of the requirement that failed as well as detailed information about where and why it failed. Since a test can fail in many places, the report includes information about every instance where a failure occurred. It can be somewhat verbose.

Figure 1 — Sample Results

5 GIMI-ARCHITECTURE

5 GIMI-ARCHITECTURE

5.1. Approach

The GIMI Conformance Testing Tool was implemented as a Java-based application for integration into TEAM Engine — an open source test harness maintained by the OGC. The conformance testing tool uses the TestNG framework for handling test assertions, and makes use of the following open source Apache software libraries:

- Apache SIS for some HEIF file reading
- Apache Jena for reading RDF
- Apache Xerces for XML validation

In addition to the aforementioned Apache software libraries, it also makes use of the open source jMISB library for reading MISB KLV binary. As illustrated in Figure 2, the GIMI conformance testing tool delegates the testing of ISOBMFF and HEIF conformance testing to an instance of ComplianceWarden through a separate process. The following section describes of the components of the architecture.

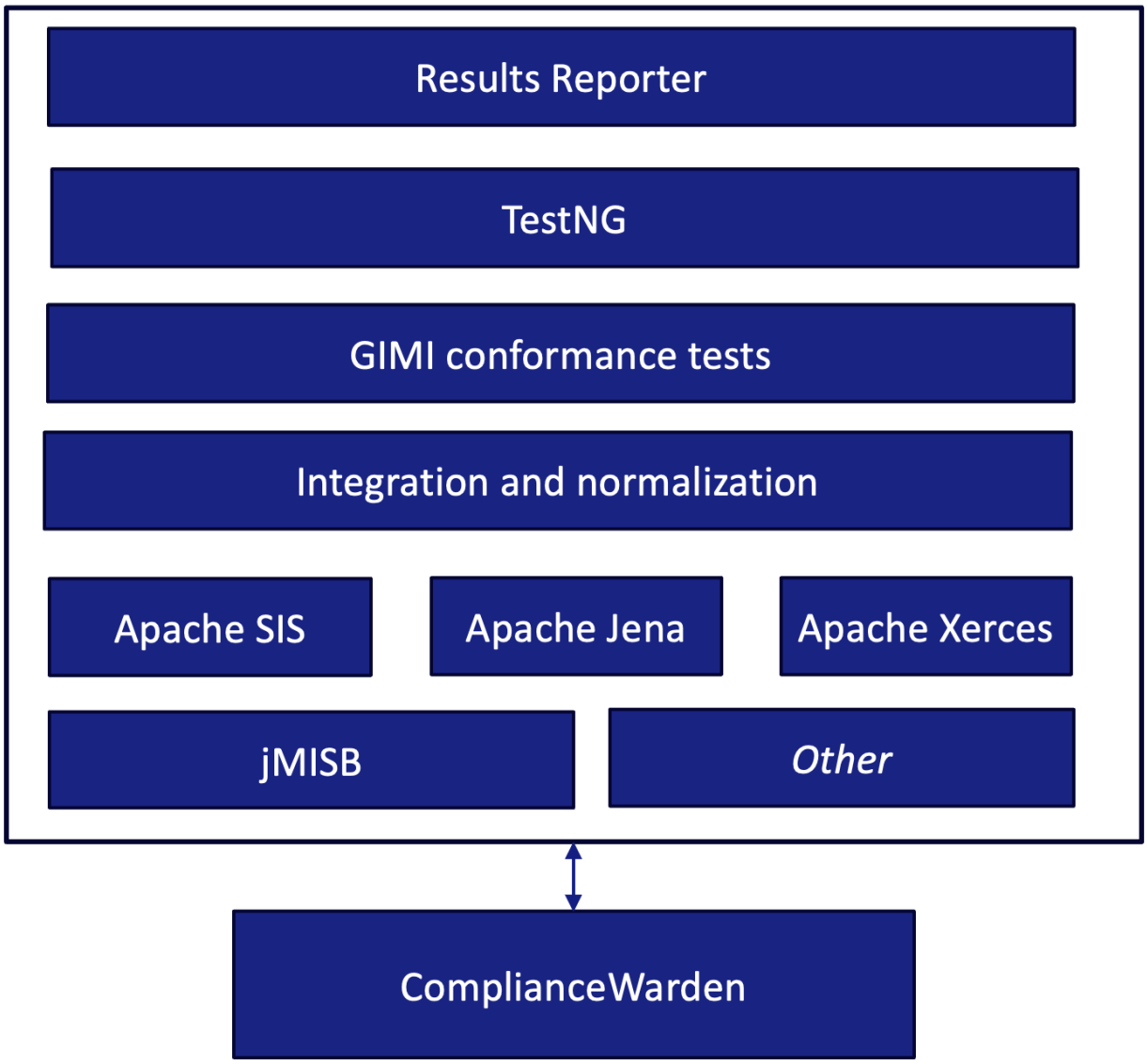

Figure 2 — Architecture of the GIMI Conformance Testing Tool

GIMI files are constructed of "boxes". Each "box" is identified by a box type, and a definition of what content can reside in that box. Boxes can be nested, with boxes within a box.

The GIMI Executable Test Suite will begin by reading the test file, box by box, and building a tree of the box data. The box tree is then processed to produce a Java object tree. The object tree is optimized to provide the test scripts easy access to the information then need. The conformance tests will then be executed against that information.

Boxes fall into two categories; essence and metadata. Essence boxes contain the imagery or sensor data to be provided through the file. Exploitation of essence box contents is accomplished using a CODEC. Metadata boxes contain information for understanding and accessing the essence. Metadata will be provided in three encodings; XML, RDF, or KLV. These encodings are pre-processed in the Object tree stage, relieving the test code of the need to process the raw data.

5.2. ETS Packages

The GIMI1 package consists of two major subpackages; the Content package and the Conformance package. The Content package receives the parsed boxes from SIS and processes them into an object-based data model. The Conformance package contains the TestNG tests. These tests leverage the information contained in the object-based data model to determine if a test criteria is met or not.

5.2.1. Conformance Package

The Conformance package contains five sub-packages, one for each Conformance Class. Each sub-package contains one or more Lava classes, each containing test methods with similar information requirements. Each java class contains an initialize() method which runs before any of the test methods. This method extracts information from the object model and prepares it for use by the test methods. Each test method addresses one and only one requirement.

5.2.2. Content Package

The Content package contains seven sub-packages. These sub-packages are organized based on central concepts from the GIMI standard.

5.2.2.1. Metadata Package

The Metadata Package supports most of the content which is not temporal in nature. That includes security markings, still imagery, and supporting RDF metadata.

5.2.2.2. Media Package

Addresses the Media and the Identified Media boxes. This is where most of the essence and some of the metadata content is stored. Currently there is not much processing taking place in this package since there are no requirements specific to this content.

5.2.2.3. Movie Package

The movie package organizes the temporal media and its supporting metadata. A Movie is constructed from one or more tracks. It can also be divided into multiple parts (fragments). In that case, the movie is constructed upon playback from the content of the Movie class and the associated Movie Fragment classes.

5.2.2.4. Track Package

A Movie contains multiple tracks. Each track addresses a specific media type. Tracks are synchronized by time stamps associated with “Sync” samples in each track. Like Movies, a track can be composed of multiple Track Fragments.

5.2.2.5. Sample Package

A Sample is roughly equivalent to a single pixel (a sample) collected by a sensor. Not all samples need to be described. It is sufficient to know which samples are the “sync” samples, the number of samples between “sync” samples, and the properties of those samples.

5.2.2.6. Group package

GIMI provides a rich capability to associate media in groups. The logic implementing these capabilities reside in this package.

5.2.2.7. Data Package

There are some classes which are used across multiple packages. In most cases these represent content extracted from the media packages.

5.3. Supporting Libraries and Tools

5.3.1. Apache SIS

Apache Spatial Information System (Apache SIS) is an open-source Java library designed to support the development of geospatial applications. It provides core data structures for representing geographic features, spatial metadata, and coordinate reference systems, and it implements the GeoAPI interfaces to ensure interoperability with industry standards.

A key strength of Apache SIS lies in its adherence to OGC and ISO geospatial standards. Its referencing module supports coordinate reference system definitions and transformations between them. The library also provides storage modules capable of reading and writing a variety of geospatial data formats, including GeoTIFF, ASCII grids, and GIMI/GeoHEIF files.

Within the GIMI ETS, the Apache SIS library is used to support the parsing and reading of ISOBMFF and HEIF boxes. This includes code specifically designed for reading GIMI

specific content in those boxes. Much of the source code developed in this Testbed for GIMI conformance testing was also contributed back into Apache SIS to improve the box parsing.

GIMI files are input to the Executable Test Suite using the GeoHeif software developed for Apache SIS. This software parses the GIMI file and generates one Java class for each box. While currently included in the GIME ETS project, they are expected to become a part of the Apache SIS platform in the future.

5.3.2. Apache Jena

Apache Jena is a free, open-source Java framework for building Semantic Web and Linked Data applications. It provides developers with tools and APIs to create, manipulate, and query RDF (Resource Description Framework) data, which forms the foundation of many knowledge graph systems. The support for RDF includes the ability to write or read many of the common RDF encoding formats, e.g. RDF/XML, RDF Turtle, and JSON-LD.

A central feature of Apache Jena is its support for querying and storage of RDF data. Through its ARQ engine, Jena provides a full implementation of the SPARQL 1.1 query language, allowing users to perform complex queries. The framework includes TDB, a high-performance native triple store for persistent storage, as well as Fuseki, a server component. Jena's ontology API supports vocabularies such as RDFS, GeoSPARQL and OWL, thereby making it ideal for integration into the GIMI Conformance Testing Tool.

Within the GIMI ETS, the Apache Jena library is used for parsing and reading RDF-encoded metadata retrieved from a GIMI file. By using Jena for this purpose, the ETS benefits from the ability of Jena to both parse and query an RDF graph. It is envisaged that future extensions of the ETS will include the ability to run SPARQL queries and apply SHACL shapes during validation.

5.3.3. Apache Xerces

Apache Xerces is an open-source software project from the Apache Software Foundation that provides a family of XML parsers and related tools. The project includes implementations in multiple programming languages, notably Java, C++, and Perl. The parsers are optimized to handle large and complex XML documents while maintaining efficiency and reliability. The modular architecture enables developers to extend and customize parsing behavior for advanced use cases.

A core strength of Apache Xerces is its comprehensive support for XML standards and APIs. Xerces parsers are designed to be fully compliant with specifications such as XML 1.0 and 1.1, namespaces, and XML Schema, ensuring accurate validation and interoperability. The project supports widely used programming interfaces including DOM, SAX, JAXP, and StAX, providing flexibility for different styles of XML processing.

Within the GIMI ETS, the Apache Xerces library is used for parsing XML-encoded content such as the Security labeling included in a GIMI file. Not only is the ETS able to parse the XML-encoded Security metadata, but it is also able to validate the metadata to confirm that it is

structured according to the schema defined by the Intelligence Community Information Security Marking (IC ISM) standard.

5.3.4. ComplianceWarden

ComplianceWarden is an open-source pluggable compliance checker developed within the GPAC ecosystem to validate media files against a variety of formal specifications, particularly those related to modern multimedia container and codec standards. It serves as conformance testing tool for key specifications such as MPEG MIAF, AOM AVIF, and AV1 HDR10+, and is designed to extend to other formats including ISOBMFF, HEIF, MP4, and CMAF. The project is written in modern C++ and distributed under a permissive BSD-3 license, making it suitable for both research and production environments where standards compliance is critical.

A defining architectural feature of ComplianceWarden is its separation of parsing and validation. The tool first parses an input file into a generic abstract syntax tree (AST), which represents the structure of the media content independent of any specific rule set. It then applies validation rules attached to particular specifications on this AST, enabling flexible and extensible compliance checking across multiple standards. This design allows new specifications and rules to be added incrementally, supporting evolving industry standards while maintaining a consistent processing pipeline.

Within the GIMI ETS, the ComplianceWarden tool is used for verifying that a file conforms to the ISOBMFF and HEIF standards. ComplianceWarden is called by the GIMI ETS through an external process. This is because ComplianceWarden is mainly implemented in C/C++ whereas the GIMI ETS is a Java-based application. Nonetheless, the results of the tests executed within ComplianceWarden are reported back through the GIMI ETS and TEAM Engine. Therefore, the user only ever needs to interact with the entire GIMI ETS through a TEAM Engine.

6 INSTALLATION

6 INSTALLATION

The conformance test tool is implemented as an Executable Test Suite for deployment in version 6 of TEAM Engine.

6.1. Installation of dependencies

The following dependencies must be installed on the machine before TEAM Engine and the Executable Test Suites can be built. Additional dependencies are handled through Maven and the pom.xml file.

- Java 17: Download the Java Development Kit (JDK) 17. OpenJDK is recommended.
- Apache Maven: Download the latest release from https://maven.apache.org/download.cgi.
- Apache Tomcat 10.1: TEAM Engine requires Tomcat 10.1.

6.2. Installation through Docker

The quickest way to install TEAM Engine is to create a docker image and then deploy the image in a running container.

An example of how to build the project is:

```
mvn clean install site -Dmaven.javadoc.skip=true -Pintegration-tests,docker
```

Listing 4

Then to run the built docker image, run this command:

```
docker run -p 8081:8080 ogccite/ets-gimi-1:latest
```

Listing 5

Verify that TEAM Engine is running by visiting this address in a web browser `http://localhost:8081/teamengine`.

If you would like to transfer the docker image to a different server, then save the docker image to a tar file

```
docker save --output ets-gimi-1.tar ogccite/ets-gimi-1:latest
```

Listing 6

Transfer it to the new server, and then upload it to the new server with this command:

```
docker load --input ets-gimi-1.tar
```

Listing 7

6.3. Direct installation on host machine

To install TEAM Engine v6 enter the following commands in a terminal.

NOTE: These instructions assume a Linux or MacOS environment, with Java 17 and Maven 3.9 pre-installed.

```
cd t61

git clone https://github.com/opengeospatial/teamengine.git

curl -L https://dlcdn.apache.org/maven/maven-3/3.9.11/binaries/apache-maven-3.9.11-bin.zip > apache-maven-3.9.11-bin.zip

unzip apache-maven-3.9.11-bin.zip

cd teamengine

../apache-maven-3.9.11/bin/mvn clean package

cd ..

curl -O https://archive.apache.org/dist/tomcat/tomcat-10/v10.1.9/bin/apache-tomcat-10.1.9.zip

unzip apache-tomcat-10.1.9.zip && cp -R apache-tomcat-10.1.9 tomcat10 && rm -R apache-tomcat-10.1.9

chmod 777 -R ./tomcat10/bin/

unzip /t61/teamengine/teamengine-web/target/teamengine.war -d /t61/tomcat10/webapps/teamengine

unzip /t61/teamengine/teamengine-web/target/teamengine-common-libs.zip -d /t61/tomcat10/lib

unzip /t61/teamengine/teamengine-console/target/teamengine-console-6.1.0-SNAPSHOT-base.zip -d /t61/tomcat10/te_base

export JAVA_OPTS="-Xms1024m -Xmx6144m -DTE_BASE=/t61/tomcat10/te_base"

export CATALINA_OPTS="$CATALINA_OPTS -Dlog4j2.formatMsgNoLookups=true"
```

```
./tomcat10/bin/startup.sh
```

Listing 8

Now verify that TEAM Engine is running by visiting this address in a web browser `http://localhost:8081/teamengine`

6.4. Download and build from source

6.4.1. Get the source code

The source code is available from GitHub. Use Git to clone the repository and checkout a branch or a tagged release as indicated below:

git clone https://github.com/opengeospatial/teamengine.git git checkout ${project.version} (git checkout 6.0.0)

Apache Maven 3.9 is required to build the TEAM Engine code base, which currently consists of the following modules:

- teamengine-core: Main CTL script processor
- teamengine-resources: Includes shared resources such as stylesheets and schemas
- teamengine-spi: Provides an extensibility framework and a REST-like API for test execution
- teamengine-spi-ctl: Enables the execution of legacy CTL test suites using a RESTful API.
- teamengine-realm: A custom Tomcat user realm for file-based authentication
- teamengine-web: A web application for executing test suites and browsing test results
- teamengine-console: A console application that provides a command-line interface for executing test suites in Unix and Windows environments.
- teamengine-virtualization: Enables the automatic creation of virtualization images using Packer.

6.4.2. Build with Maven

Simply run `mvn package` in the root project directory to generate all build artifacts, or run `mvn install` to also install them into the local Maven repository. The main build artifacts are listed below.

- teamengine-console-${project.version}-bin.[zip|tar.gz]: Archive containing the console application (command-line usage)
- teamengine-console-${project.version}-base.[zip|tar.gz]: Archive containing the initial contents of the teamengine instance directory (TE_BASE)
- teamengine.war: The JEE (Servlet) web application
- teamengine-common-libs.[zip|tar.gz]: Archive containing common runtime dependencies (e.g. JAX-RS 1.1, Apache Derby)

The site documentation can be generated by simply executing the 'mvn site' phase against the top-level POM. An aggregate PDF document is also created and placed in the target/pdf directory.

6.4.3. Set up an instance directory (TE_BASE)

The value of the TE_BASE system property or environment variable specifies the location of the instance directory that contains several essential sub-directories. Create the TE_BASE directory (e.g. at /srv/teamengine) and unpack the contents of the teamengine-console-${project.version}-base archive into this location; make sure that users (including the Tomcat user) have write access. The structure of the TE_BASE directory is shown below.

```
TE_BASE |-- config.xml # main configuration file |-- resources/ #
test suite resources |-- site # site-specific HTML (welcome, title,
etc.) |-- lib # Java-based test suites and supporting libraries |--
scripts/ # CTL test scripts |-- work/ # teamengine work directory |--
users/ |-- {user1}/ # user account information and test run results
|-- {user2}/ +-- ...
```

Figure 3

CTL test scripts must be placed in the TE_BASE/scripts directory. Java-based test suites along with their supporting libraries are placed in the TE_BASE/resources/lib directory.

6.4.4. Create a dedicated Tomcat instance

Apache Tomcat 10.1 is a supported servlet container. It is strongly recommended that a dedicated Tomcat instance be created to host the teamengine application. That is, keep the location of the instance-specific data (referred to as CATALINA_BASE in the Tomcat documentation) separate from the Tomcat software installation (CATALINA_HOME). Create it as outlined below for your operating system.

To create a CATALINA_BASE directory in Windows:

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

Listing 9

In a Unix-like environment:

```
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
```

Listing 10

The recommended JVM options for the Tomcat instance are shown below. Note that the maximum memory heap size (-Xmx) may need to be increased as the number of concurrent users increases.

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

Listing 11

Unpack the contents of the teamengine-common-libs archive into the CATALINA_BASE/lib directory. Deploy the teamengine.war component by either copying it into the CATALINA_BASE/webapps directory or using the Tomcat Manager application. Start the Tomcat instance.

The URIs listed below provide starting points for discovering and executing test suites. Modify the path if the name of the WAR file (and hence the context path) has been changed from the default value: “teamengine”.

- /teamengine: Home page for selecting and running test suites using the web interface
- /teamengine/rest/suites: Presents a listing of available test suites that expose a RESTful API, with links to test suite documentation

6.5. Install test suite in TEAM Engine

Next you need to install the Executable Test Suite inside the instance of TEAM Engine.

```
./tomcat10/bin/shutdown.sh

git clone https://gitlab.ogc.org/ogc/T21-ets-gimi-1.git

cd T21-ets-gimi-1

mvn clean install site -Dmaven.javadoc.skip=true

unzip ./ets-gimi-1/target/ets-gimi-1-0.2-SNAPSHOT-ctl.zip -d ./tomcat10/te_
base/scripts/

unzip ./ets-gimi-1/target/ets-gimi-1-0.2-SNAPSHOT-deps.zip -d ./tomcat10/
webapps/teamengine/WEB-INF/lib/
```

```
./tomcat10/bin/startup.sh
```

Listing 12

Now verify that the GIMI Executable Test Suite is listed on the landing page of the TEAM Engine instance at `http://localhost:8081/teamengine`

6.6. Customize the welcome page

Several elements of the welcome page may be customized if desired: the logo in the header, the main text, and the content in the footer. The files located in the TE_BASE/resources/site directory can be modified to provide site-specific content:

- site/logo.png: The logo that appears in the header (at top left). The size of the default image is 127×58 px.
- site/welcome.txt: The main text.
- site/footer.txt: Footer content.
- site/title.html: Title of the web application