Open Geospatial Consortium, Inc.

Date: 2009-08-14

Reference number of this document: OGC 09-038r1

Version: 0.3.0

Category: Public Engineering Report

Editor: Clemens Portele

OGC® OWS-6 GML Profile Validation Tool ER

Warning

This document is not an OGC Standard. This document is an OGC Public Engineering Report created as a deliverable in an OGC Interoperability Initiative and is not an official position of the OGC membership. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard. Further, any OGC Engineering Report should not be referenced as required or mandatory technology in procurements.

Document type:	OpenGIS® Public Engineering Report
Document subtype:	NA
Document stage:	Approved for Public Release
Document language:	English

Preface

Suggested additions, changes, and comments on this draft report are welcome and encouraged. Such suggestions may be submitted by email message or by making suggested changes in an edited copy of this document.

The changes made in this document version, relative to the previous version, are tracked by Microsoft Word, and can be viewed if desired. If you choose to submit suggested changes by editing this document, please first accept all the current changes, and then make your suggested changes with change tracking on.

Forward

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 Inc. 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.

OWS-6 Testbed

OWS testbeds are part of OGC's Interoperability Program, a global, hands-on and collaborative prototyping program designed to rapidly develop, test and deliver Engineering Reports and Change Requests into the OGC Specification Program, where they are formalized for public release. In OGC's Interoperability Initiatives, international teams of technology providers work together to solve specific geoprocessing interoperability problems posed by the Initiative's sponsoring organizations. OGC Interoperability Initiatives include test beds, pilot projects, interoperability experiments and interoperability support services - all designed to encourage rapid development, testing, validation and adoption of OGC standards.

In April 2008, the OGC issued a call for sponsors for an OGC Web Services, Phase 6 (OWS-6) Testbed activity. The activity completed in June 2009. There is a series of on-line demonstrations available here: http://www.opengeospatial.org/pub/www/ows6/index.html The OWS-6 sponsors are organizations seeking open standards for their interoperability requirements. After analyzing their requirements, the OGC Interoperability Team recommended to the sponsors that the content of the OWS-6 initiative be organized around the following threads:

1. Sensor Web Enablement (SWE)
2. Geo Processing Workflow (GPW)
3. Aeronautical Information Management (AIM)
4. Decision Support Services (DSS)
5. Compliance Testing (CITE)

The OWS-6 sponsoring organizations were:

- U.S. National Geospatial-Intelligence Agency (NGA)
- Joint Program Executive Office for Chemical and Biological Defense (JPEO-CBD)
- GeoConnections - Natural Resources Canada
- U.S. Federal Aviation Agency (FAA)
- EUROCONTROL
- EADS Defence and Communications Systems
- US Geological Survey
- Lockheed Martin

- BAE Systems
- ERDAS, Inc.

The OWS-6 participating organizations were:
52North, AM Consult, Carbon Project, Charles Roswell, Compusult, con terra, CubeWerx, ESRI, FedEx, Galdos, Geomatys, GIS.FCU, Taiwan, GMU CSISS, Hitachi Ltd., Hitachi Advanced Systems Corp, Hitachi Software Engineering Co., Ltd., iGSI, GmbH, interactive instruments, lat/lon, GmbH, LISAsoft, Luciad, Lufthansa, NOAA MDL, Northrop Grumman TASC, OSS Nokalva, PCAvionics, Snowflake, Spot Image/ESA/Spacebel, STFC, UK, UAB CREAF, Univ Bonn Karto, Univ Bonn IGG, Univ Bunderswehr, Univ Muenster IfGI, Vightel, Yumetech.

Contents

Page

OGC® OWS-6 GML Profile Validation Tool ER

1 Introduction

1.1 Scope

This document outlines an approach for validating data accessed from a Web Feature Service. Two types of validation are supported:

- XML Schema validation against the GML application schema
- Validation of additional constraints encoded in Schematron

This report describes the validation tool, the types of constraints that have been tested and documents the results.

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 Inc. 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.

1.2 Document contributor contact points

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

Name	Organization
Clemens Portele	interactive instruments GmbH
Stefan Hansen	LISAsoft
Dave Wesloh	NGA
Paul Birkel	MITRE

1.3 Revision history

Date	Release	Editor	Primary clauses modified	Description
2009-04-19	0.0.1	C. Portele	all	

2009-05-10	0.0.2	C. Portele	all	Revision based on comments
2009-06-03	0.0.3	C. Portele P. Birkel	all	Minor edits for consistency
2009-08-13	0.3.0	C. Reed	Various	Prepare for publication

1.4 Future work

See Clauses 8.2.2 and 9.

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.

ISO/TS 19103:2005, Geographic Information – Conceptual Schema Language

ISO 19109:2004, Geographic Information – Rules for Application Schemas

ISO/IEC 19757-3:2006 Information technology — Document Schema Definition Languages (DSDL) — Part 3: Rule-based validation — Schematron

W3C XML Schema Part 1: Structures Second Edition. W3C Recommendation (28 October 2004)

W3C XML Schema Part 2: Datatypes Second Edition. W3C Recommendation (28 October 2004)

3 Terms and definitions

For the purposes of this report, the definitions specified in ISO/TS 19103 and ISO 19109 shall apply.

4 Conventions

4.1 Abbreviated terms

GML Geography Markup Language

GSIP GEOINT Structure Implementation Profile

ISO International Organization for Standardization

LoD Level of Detail

NGA	National Geospatial-Intelligence Agency
OCL	Object Constraint Language
OGC	Open Geospatial Consortium
OWS	OGC Web Services
UML	Unified Modeling Language
UCUM	Unified Code for Units of Measure
UTDS	Urban Topographic Data Store
WFS	Web Feature Service
XML	eXtended Markup Language

4.2 UML notation

Diagrams that appear in this standard are presented using the Unified Modeling Language (UML) static structure diagram, as described in ISO/TS 19103.

5 Validation tool guidelines overview

Application schemas conforming to ISO 19109 - Rules for application schemas - include a description of the structural components of a geographic dataset, in particular a schema for the types and their properties, but they also include constraints on them. See Figure 1, which is taken from ISO 19109; here, GF_Constraint represents such constraints. For application schemas in UML, constraints are usually described in OCL and/or natural language.

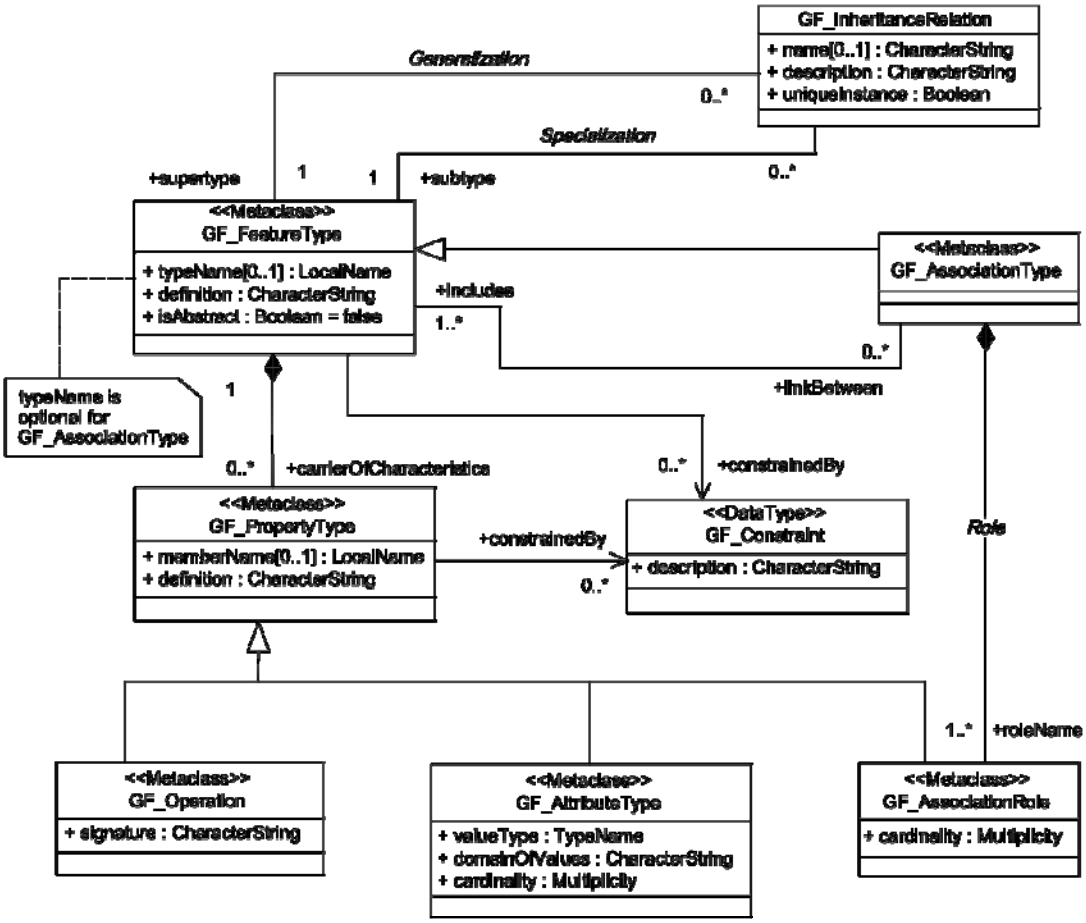

Figure 1 — Extract from the ISO 19109 General Feature Model [ISO 19109]

For the encoding of geographic data, the application schema is converted to an encoding schema specific to the language used for the encoding. For data encoded in GML, XML Schema is used to encode relevant information regarding the application schema; the result is called a GML application schema. Conformance of instance data according to the GML application schema can then be tested using schema validation, a capability offered by standard XML parsers.

However, only a subset of all requirements captured in the application schema can be represented in XML Schema. In particular, constraints (GF_Constraint in Figure 1) are usually not represented in the GML application schema and another language is needed to express such requirements in a way so that they can be tested. The typical approach for XML-based encodings is the use of Schematron (ISO/IEC 19757-3:2006[1]) as discussed in detail in the following OGC documents (see Bibliography): OWS-5 GSIP Schema Processing Engineering Report (08-078r1), and the Specification Model — Modular specifications RFC (08-131r2).

Schematron is already used by GML to express constraints that cannot be represented in XML Schema. It is currently considered the most appropriate language to express constraints on XML instance documents. Tools exist to process Schematron constraints and assert the complicance of an instance document with the specified constraints[2].

NOTE Schematron is based on XPath expressions. This has the effect that Schematron constraints are in practice limited to a single document[3]; in addition, Schematron is not Xlink-aware (without support by additional Xpath functions).

In OWS-5 the automated derivation of Schematron rules from an application schema in UML has been analysed, successfully implemented and demonstrated (see the OWS-5 GSIP Schema Processing Engineering Report and the “Application Schema Processing” video at http://www.opengeospatial.org/pub/www/ows5/index.html).

The focus of the activity in OWS-6 has been to develop a tool that could access a WFS directly, execute XML Schema and Schematron validation and report the results.

Clause 6 describes the Validation Tool, DuckHawk, used in OWS-6.

Clause 7 describes the types of constraints used in the OWS-6 initiative and provides an example for each type.

Clause 8 illustrates the execution of the validation tool and the reported results.

6 GML Profile Validation Tool

NOTE The text in this clause is partially taken from the DuckHawk User Guide (see Bibliography).

[1] http://standards.iso.org/ittf/PubliclyAvailableStandards/c040833_ISO_IEC_19757-3_2006(E).zip

[2] for example: http://www.schematron.com/implementation.html

[3] ISO/IEC 19757-4 Namespace-based Validation Dispatching Language (NVDL) might be used in the future to overcome this constraint.

6.1 Overview

DuckHawk is an open source testing framework that enables the development of automated reliability, load, performance, stress, error-handling and conformance tests. It can be used to automatically test any web service or application.

DuckHawk is written in Java and built upon JUnit 3. The tests themselves are written in Java and use a design similar to JUnit 3 tests. DuckHawk allows configuring the test parameters through configuration files.

In OWS-6, DuckHawk is used to perform conformance tests. Tests using the conformance testing module send requests to the target server and validate the returned XML response documents against an XML Schema and against Schematron rules. The support for Schematron is an OWS-6 extension to the tool.

DuckHawk collects all test results and passes them on to test listeners, which can perform processing, analysis, and generate reports. At the moment two main implementations of test listeners exist; an XML-report generator and a HTML-report generator.

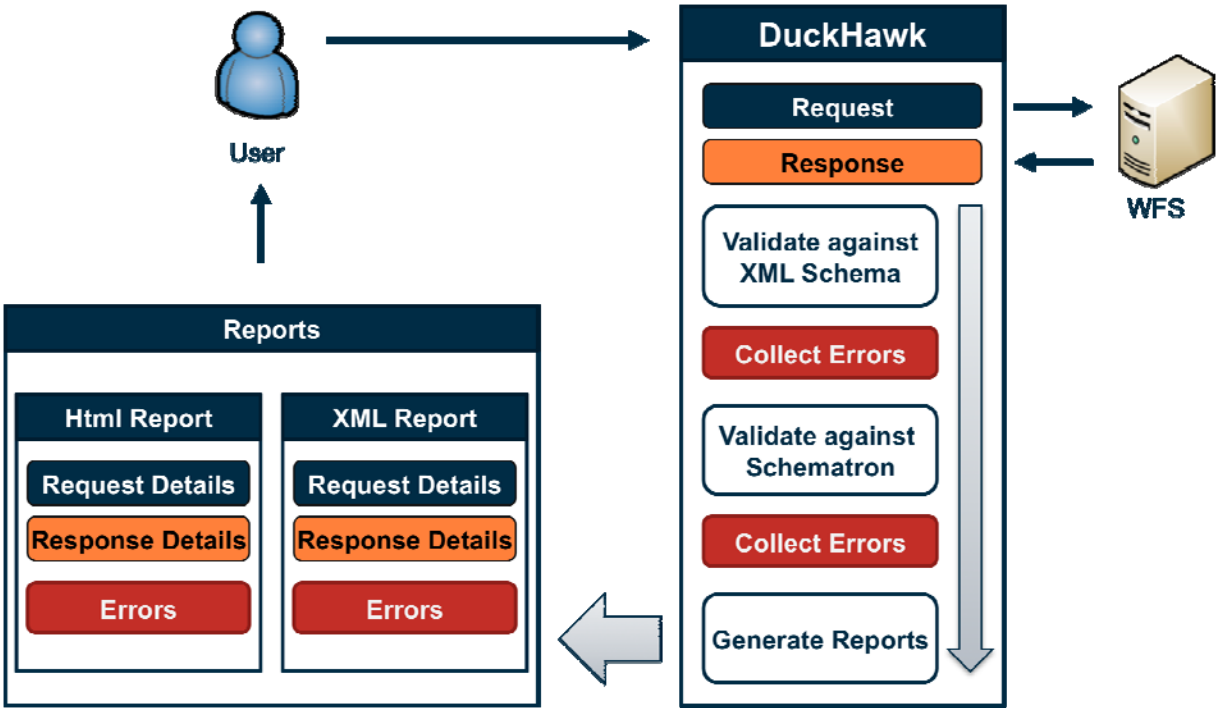

Figure 2 — DuckHawk information flow in OWS-6

More information can be found at the webpage of the DuckHawk project: http://xircles.codehaus.org/projects/duckhawk.

6.2 Installing the Test Suite

The test suite comes packed in a ZIP-archive. Extract this archive to a folder on the systems from which the tests will be run.

The ZIP-archive contains the following files:

Name		**Purpose**
log4j.properties		Logging properties
ows6.properties		Test suit properties
jar	ows6-1.0-SNAPSHOT-jar-with-dependencies.jar	Java application
reports	xml	Folder for XML reports
	html	Folder for HTML reports
Schematron	*file*.sch	Folder for Schematron rules
	iso_schematron_skeleton_for_saxon.xml	XSL transformation from Schematron to XSLT
tests	OWS6GenericTest.csv	Test parameters

6.3 Configuring the Test Suite Context

The file ows6.properties specifies the context in which the tests will be executed. It is located in the root-directory of the installation.

The parameters listed below have to be set within this file:

Parameter	Example	Explanation
host	host = services.interactive-instruments.de	URL of the host.
port	Port = 80	Port of the server.
serverPath	serverPath = ows6/cgi-bin/ows6city-wfs.exe or serverPath = ows6/cgi-bin/ows6airport-wfs.exe	Path to service interface.
testsConfigDir	testsConfigDir = tests/	Location of the test parameters.
testsConfigurationFile	testsConfigurationFile = OWS6GenericTest.csv	Name of the file containing the test parameters.
schematronFolder	schematronFolder = schematron/	Location of Schematron files.

schematronFilesExtension	schematronFilesExtension = sch	Files extension of files containing Schematron rules.
schematronTransformer	schematronTransformer = iso_schematron_skeleton_for_saxon.xml	XSLT file for Schematron transformation.
reportXmlDir	reportXmlDir = ./reports/xml	Folder for XML reports.
reportHtmlDir	reportHtmlDir = ./reports/html	Folder for HTML reports.

6.4 Running the Test Suite

To start the test suite the command

java -jar jar/ows6-1.0-SNAPSHOT-jar-with-dependencies.jar

has to be executed in the root directory of the installation. After entering the command the tests will be executed. DuckHawk will generate log-files and reports, but also will output internal information on the screen. After the execution DuckHawk will return to the command-line.

NOTE DuckHawk's internal process uses exceptions to report any validation errors. Therefore, any validation error will result in the output of an exception message in the console. This does not indicate an error within DuckHawk.

6.5 Output

6.5.1 Reports

Duckhawk's OWS-6 module produces two types of reports. All of the data collected during a test run is stored in XML reports. Each time a test suite is executed DuckHawk generates a summary report and a report for each of the executed test classes.

As the main purpose of the XML reports is to provide a machine-readable representation of the test data, DuckHawk also produces HTML reports. HTML reports include a summary report as well as reports for each test class. The reports for the test classes are linked from summary report.

6.5.2 Log File

During execution of the OWS6 test suite a log file is generated. By default it is stored under the name ows6.log in the directory in which the test suite had been started. The default configuration is to backup and reset this log file once it reaches 200KB in size. The name and the maximum size of the log file can be adjusted in the configuration file log4j.properties, which is located in the same directory.

6.6 Adding and Adjusting Tests

6.6.1 Adjusting Existing Tests

Existing tests can be adjusted by editing the configuration file for that test. The configuration file for each test defines the parameters specific to that test.

The only parameter required for a test in the OWS-6 module is a WFS-request. Each line of the file, except of the first line which defines the name of the parameters, contains one request. The request must be in one single line without any line breaks.

EXAMPLE The test request used in OWS-6 is

```
<wfs:GetFeature service="WFS" version="1.1.0" xmlns:utds="http://www.opengis.net/ows-6/utds/0.3" xmlns:wfs="http://www.opengis.net/wfs" xmlns:ogc="http://www.opengis.net/ogc" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.opengis.net/wfs  http://schemas.opengis.net/wfs/1.1.0/wfs.xsd http://www.opengis.net/ows-6/utds/0.3 http://services.interactive-instruments.de/ows6/data/schema/UTDS-CityGML.xsd"><wfs:Query typeName="utds:Building"/></wfs:GetFeature>
```

This request returns all buildings. It could be changed to a different query, see Clause 8 for examples.

6.6.2 Adding New Tests

New tests can be added by extending OWS6GenericTest.csv. A new, alternative configuration file can also be created. The parameter *testsConfigurationFile* in the ows6.properties controls which configuration file to use.

EXAMPLE New tests are added by adding new lines with a single WFS request each to OWS6GenericTest.csv. See Clause 8 for examples of such requests. For each request, all Schematron rules are assessed.

6.6.3 Adding Schematron Rules

The test suite verifies each response to a request against the Schematron rules defined in all files with the extension defined by the parameter *schematronFilesExtension* in the Schematron folder defined by the parameter *schematronFolder*.

To add new rules simply add a file containing the rules with the correct extension to the Schematron folder. Responses will be verified against these new rules in the next test run.

EXAMPLE In the configuration used in OWS-6, if additional schematron constraints have been specified, put these in a new file with the extension ".sch" in the folder "schematron".

7 Constraint types used in OWS-6

7.1 Overview

This clause describes the types of constraints used in the OWS-6 initiative and provides an example for each type.

The Schematron fragments use the following XML namespace prefixes:

```
<ns prefix="sch" uri="http://purl.oclc.org/dsdl/schematron"/>

<ns prefix="b" uri="http://www.opengis.net/citygml/building/1.0"/>

<ns prefix="c" uri="http://www.opengis.net/citygml/1.0"/>

<ns prefix="icism" uri="urn:us:gov:ic:ism:v2"/>

<ns prefix="u" uri="http://www.opengis.net/ows-6/utds/0.3"/>

<ns prefix="gml" uri="http://www.opengis.net/gml" />
```

The Schematron tests are run in addition to the XML Schema validation against the GML application schema.

The application schema and data used are described in the OWS-6 UTDS-CityGML Implementation Profile (see Bibliography).

7.2 Valid code list values

This test validates that only allowed code list values are used in properties that represent code list values. We distinguish two types:

- constraints requiring fixed values
- constraints requiring the existence of the value in the code list

For fixed values, a Schematron rule checking for explicit values is specified. In the UTDS-CityGML such constraints have to be used for code list values that are specified by CityGML due to a lack of explicit code list references in the CityGML encoding.

EXAMPLE 1 The allowed building classes are: administration, security, traffic, schools, education, research, industry, church institution, storage, or habitation.

In OCL this constraint could be expressed as

```
context Building inv:

class->isEmpty() or class=BuildingClassType::1020 or
class=BuildingClassType::1140 or class=BuildingClassType::1170 or
```

class=BuildingClassType::1100 or class=BuildingClassType::1120 or
class=BuildingClassType::1160 or class=BuildingClassType::1080 or
class=BuildingClassType::1150 or class=BuildingClassType::1000

NOTE The BuildingClass code list values specified in CityGML are:

BuildingClassType			
Code list derived from German authoritative standards ALKIS/ATKIS (www.adv-online.de)			
1000	**habitation**	**1100**	**schools, education, research**
1010	**sanitation**	**1110**	**maintainence and waste management**
1020	**administration**	**1120**	**healthcare**
1030	**business, trade**	**1130**	**communicating**
1040	**catering**	**1140**	**security**
1050	**recreation**	**1150**	**storage**
1060	**sport**	**1160**	**industry**
1070	**culture**	**1170**	**traffic**
1080	**church institution**	**1180**	**function**
1090	**agriculture, forestry**		

The Schematron encoding would be

```
<rule context="//u:Building">

    <assert test=" count(b:class)=0 or b:class='1020' or b:class='1140' or
    b:class='1170' or b:class='1100' or b:class='1120' or b:class='1160' or
    b:class='1080' or b:class='1150' or b:class='1000'">The building with id '<value-
    of select="@gml:id"/>' is of a CityGML building class that is not allowed in
    UTDS.</assert>

</rule>
```

EXAMPLE 2 The value used for the condition of a building has to be in the ConditionOfFacility code list used for the property value.

There is no need to express the constraints in OCL as this constraint is already part of the code list concept.

In Schematron, the constraint could be expressed as

```
<rule context="//u:Building">

    <let name="code" value="u:conditionOfFacility"/>

    <let name="codeSpaceDoc"
    value="document(u:conditionOfFacility/@codeSpace)/gml:Dictionary"/>

    <assert test="$codeSpaceDoc">Unable to find the condition-of-facility code list
    dictionary.</assert>
```

```
        <assert
        test="count($codeSpaceDoc/gml:dictionaryEntry/gml:Definition[gml:name=$co
        de])=1">The building with id '<value-of select="@gml:id"/>' has a value for the
        condition of the facility (<value-of select="$code"/>) that is not part of the
        associated codelist.</assert>

    </rule>
```

EXAMPLE 3 In UTDS-CityGML, code lists are not feature-type-specific. However, in UTDS, usually only a subset of all values of a code list are valid for a specific feature type. This can be assessed using constraints. In this example, we check that the values the vertical construction material of an aircraft hangar are valid.

```
    <rule context="//u:AircraftHangar">

        <assert test="count(u:verticalConstructionMaterial[1])=0 or
        u:verticalConstructionMaterial[1]='1' or u:verticalConstructionMaterial[1]='3'
        or u:verticalConstructionMaterial[1]='8' or
        u:verticalConstructionMaterial[1]='9' or u:verticalConstructionMaterial[1]='12'
        or u:verticalConstructionMaterial[1]='13' or
        u:verticalConstructionMaterial[1]='15' or
        u:verticalConstructionMaterial[1]='18'">The aircraft hangar with id '<value-of
        select="@gml:id"/>' has a vertical construction material '<value-of
        select="u:verticalConstructionMaterial[1]"/>' that is not allowed in
        UTDS.</assert>

    </rule>

    <rule context="//u:AircraftHangar">

        <assert test="count(u:verticalConstructionMaterial[2])=0 or
        u:verticalConstructionMaterial[2]='1' or u:verticalConstructionMaterial[2]='3'
        or u:verticalConstructionMaterial[2]='8' or
        u:verticalConstructionMaterial[2]='9' or u:verticalConstructionMaterial[2]='12'
        or u:verticalConstructionMaterial[2]='13' or
        u:verticalConstructionMaterial[2]='15' or
        u:verticalConstructionMaterial[2]='18'">The aircraft hangar with id '<value-of
        select="@gml:id"/>' has a vertical construction material '<value-of
        select="u:verticalConstructionMaterial[2]"/>' that is not allowed in
        UTDS.</assert>

    </rule>

    <rule context="//u:AircraftHangar">

        <assert test="count(u:verticalConstructionMaterial[3])=0 or
        u:verticalConstructionMaterial[3]='1' or u:verticalConstructionMaterial[3]='3'
        or u:verticalConstructionMaterial[3]='8' or
        u:verticalConstructionMaterial[3]='9' or u:verticalConstructionMaterial[3]='12'
```

```
or u:verticalConstructionMaterial[3]='13' or
u:verticalConstructionMaterial[3]='15' or
u:verticalConstructionMaterial[3]='18'">The aircraft hangar with id '<value-of
select="@gml:id"/>' has a vertical construction material '<value-of
select="u:verticalConstructionMaterial[3]"/>' that is not allowed in
UTDS.</assert>

</rule>
```

7.3 Valid coordinate reference system

Let's assume that an application schema requires the use of a specific coordinate reference system in geometries. Then the reference could check that the correct coordinate reference system is used in instance data.

EXAMPLE UTDS-CityGML data has to be provided by default in WGS84 (geographic 3D). Using the EPSG register of coordinate reference systems this constraint could be expressed in Schematron as

```
<rule context="//u:Building">

    <assert test="b:lod1Solid/gml:Solid/@srsName='urn:ogc:def:crs:EPSG::4979' or
    ../../gml:boundedBy/gml:Envelope/@srsName='urn:ogc:def:crs:EPSG::4979'">
    The building with id '<value-of select="@gml:id"/>' has a geometry that used an
    invalid coordinate reference system.</assert>

</rule>
```

NOTE If a CRS dictionary would be available, the existence of the CRS in that dictionary could also be validated, similar to the code list case.

7.4 Valid units of measurement

Let's assume that an application schema requires the use of a specific unit of measure for a certain property. Then the reference could check that the correct unit is used in instance data.

EXAMPLE UTDS-CityGML data has to use metre as the unit for height values. Using the Unified Code for Units of Measure (UCUM) dictionary of measurement units this constraint could be expressed in Schematron as

```
<rule context="//u:Building">

    <assert test="b:measuredHeight/@uom='m'">The building with id '<value-of
    select="@gml:id"/>' has a measured height value that is not in metre, but in unit
    '<value-of select="b:measuredHeight/@uom"/>'.</assert>

</rule>
```

NOTE If a units of measure dictionary was available, the existence of the unit in that dictionary could also be validated, similar to the code list case.

7.5 Existence of conditional property values

In the application schema, the minimum multiplicity of a property may be “0”, but under certain conditions, the property must have a value.

EXAMPLE The CityGML schema specifies some properties with a minimum multiplicity of “0”, but they are required in UTDS-CityGML. For example, every building has to have a solid geometry on level-of-detail 1. In Schematron this could be expressed as

```
<rule context="//u:Building">

    <assert test="count(b:lod1Solid)=1">The building with id '<value-of
    select="@gml:id"/>' has no solid geometry on LoD 1.</assert>

 </rule>
```

7.6 Other conditions on property values

EXAMPLE 1 In UTDS-CityGML, a building of class “habitation” has to have a function “residential building” and no usage value (see mapping of UTDS feature function “detached house” in the OWS-6 UTDS-CityGML Implementation Profile ER). In Schematron this could be expressed as

```
<rule context="//u:Building">

    <assert test="(count(b:class)=0 or b:class!='1000' or (b:class='1000' and
    b:function='1000' and count(b:usage)=0))">The building with id '<value-of
    select="@gml:id"/>' of class 1000 has invalid function or usage values.</assert>

</rule>
```

EXAMPLE 2 In UTDS-CityGML, the solid geometry on level-of-detail 1 has to be a gml:Solid. In Schematron this could be expressed as

```
<rule context="//u:Building">

    <assert test="count(b:lod1Solid/gml:Solid)=1">The building with id '<value-of
    select="@gml:id"/>' has a solid geometry on LoD 1 which is not a
    gml:Solid.</assert>

</rule>
```

8 Validation

8.1 Validation in OWS-6

This clause illustrates the execution of the validation tool and the reported results.

The WFS calls are entered in OWS6GenericTest.csv. To test the building-related constraints on all features in the dataset, the following line is added to the file:

<wfs:GetFeature service="WFS" version="1.1.0" xmlns:utds="http://www.opengis.net/ows-6/utds/0.3" xmlns:wfs="http://www.opengis.net/wfs" xmlns:ogc="http://www.opengis.net/ogc" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.opengis.net/wfs http://schemas.opengis.net/wfs/1.1.0/wfs.xsd http://www.opengis.net/ows-6/utds/0.3 http://services.interactive-instruments.de/ows6/data/schema/UTDS-CityGML.xsd"><wfs:Query typeName="utds:Building"/></wfs:GetFeature>

The Schematron rules specified in the examples in clause 7 are added to a file with the extension ".sch" in the "schematron" folder.

The tests are then executed as described in 6.4. The following test report is created, identifying no errors:

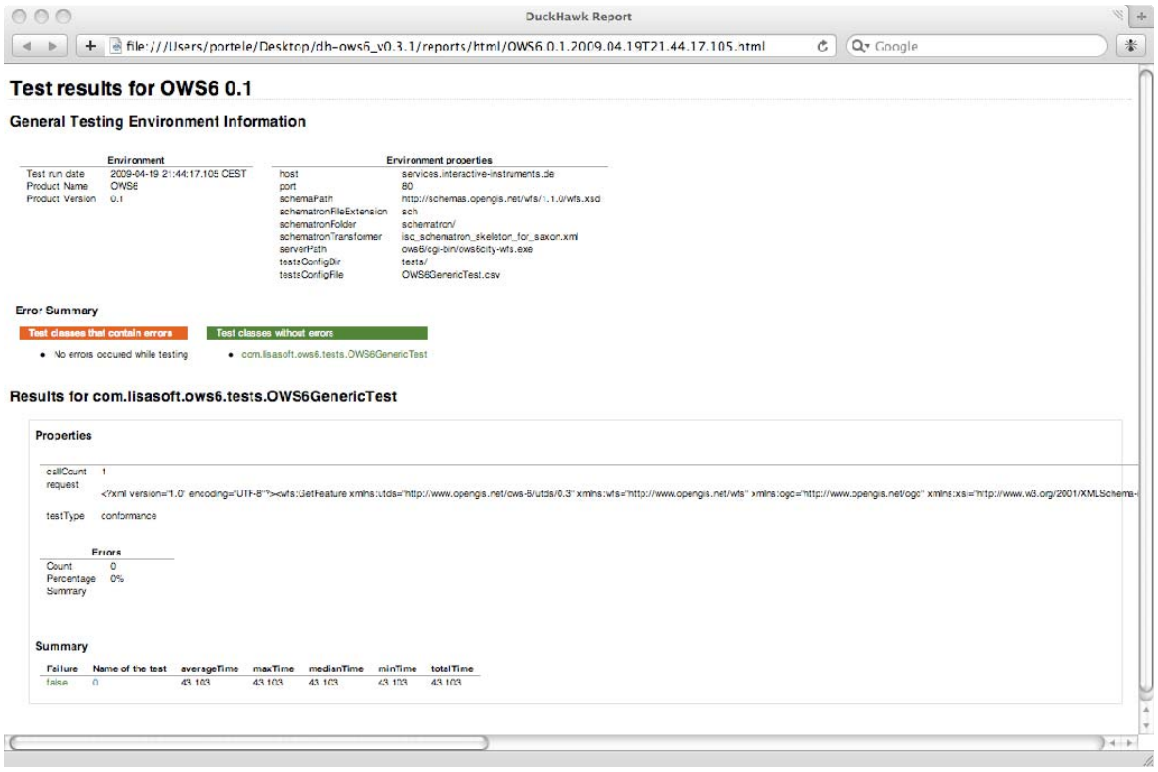

Figure 3 — DuckHawk report (1)

If we modify the Schematron rules to disallow buildings of class “administration” (b:class!=’1020’) and rerun the tests, errors are reported:

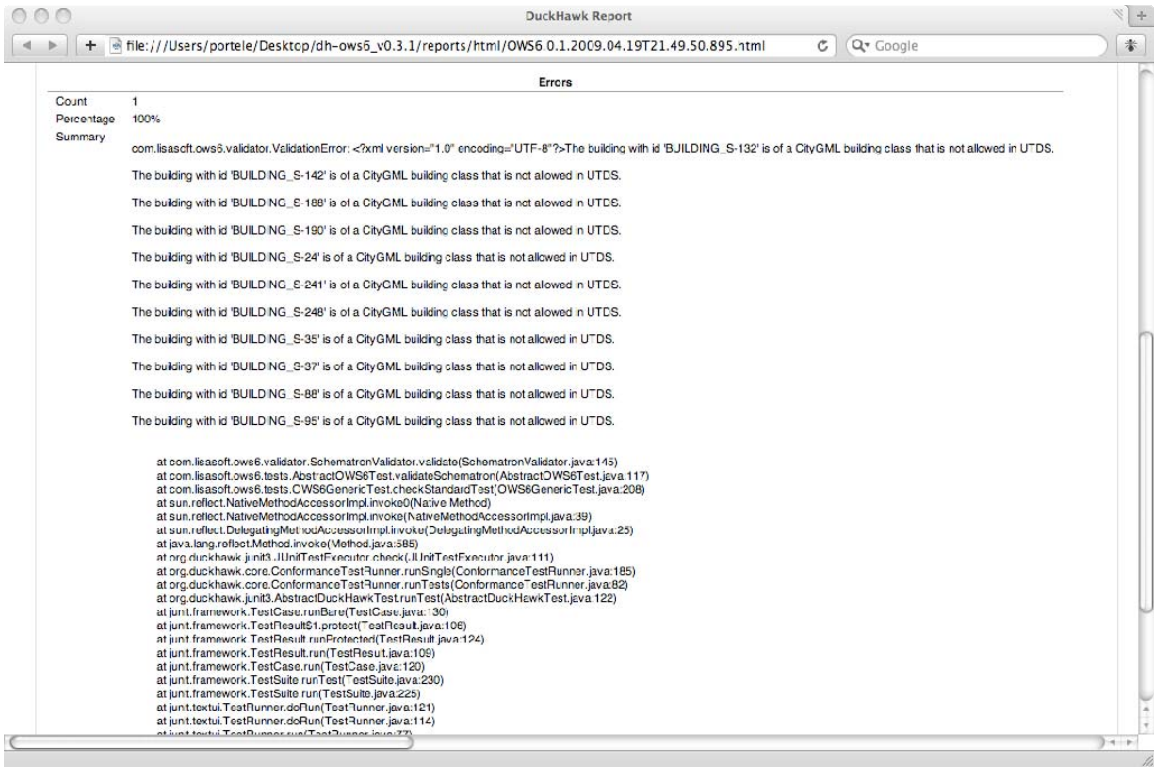

Figure 4 — DuckHawk report (2)

NOTE DuckHawk uses exceptions to signal events during validation and reports stack dumps in these cases. Although this looks like an issue in the software, this is a “feature” of the tool and not a bug.

8.2 Validation in general

8.2.1 General use case

The underlying general use case for this tool beyond its usage in OWS-6 is validation of data according to a data content specification which includes a predefined set of rules/constraints for that particular "product" and where these constraints are stored and published in a registry. An example for such a product is UTDS-CityGML as used above.

As data (identified for example as UTDS compliant) is received, either from a contractor, by a field operation or by access from a Web Feature Service, the recipient of the data (NGA analyst) would access the constraints in the registry and use them in the validation tool to validate whether in fact the data is actually compliant to the data content specification.

8.2.2 Additional tool requirements

To address the use case, a future version of a validation tool should be enabled to access Schematron rules from a registry depending on the “flavor” of the data (urban, local, etc.).

Furthermore, in addition to accessing data directly from a WFS, access to data in a local file would need to be supported, too.

8.2.3 Additional types of WFS requests

The WFS request used in the OWS-6 tests is shown in 8.1. This requests downloads all building UTDS-CityGML features. However, in other situations other kinds of requests will be required.

The general rules for GetFeature requests and the use of Filter Encoding apply here and a user of the tools is currently required to be familiar with these OGC standards. This sub-clause provides some examples, but do not provide a systematic introduction to creating such requests.

NOTE All examples split the requests over multiple lines for better readability. However, for the tool, all requests must be on a single line, i.e. without any line breaks.

EXAMPLE 1 Query restricted to a bounding box

```
<wfs:GetFeature service="WFS" version="1.1.0" xmlns:utds="http://www.opengis.net/ows-6/utds/0.3" xmlns:wfs="http://www.opengis.net/wfs" xmlns:ogc="http://www.opengis.net/ogc" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.opengis.net/wfs  http://schemas.opengis.net/wfs/1.1.0/wfs.xsd http://www.opengis.net/ows-6/utds/0.3 http://services.interactive-instruments.de/ows6/data/schema/UTDS-CityGML.xsd">

<wfs:Query typeName="utds:Building">

    <ogc:Filter>

        <ogc:BBOX>

            <ogc:PropertyName>gml:boundedBy</ogc:PropertyName>

            <gml:Envelope srsName="urn:ogc:def:crs:EPSG::4326">

                <gml:lowerCorner>29.8915 -90.0290</gml:lowerCorner>

                <gml:upperCorner>29.9075 -89.9742</gml:upperCorner>

            </gml:Envelope>

        </ogc:BBOX>

    </ogc:Filter>

</wfs:Query>
```

```
</wfs:GetFeature>
```

EXAMPLE 2 Query restricted to a specific building class

```
<wfs:GetFeature service="WFS" version="1.1.0" xmlns:utds="http://www.opengis.net/ows-
6/utds/0.3" xmlns:wfs="http://www.opengis.net/wfs" xmlns:ogc="http://www.opengis.net/ogc"
xmlns:build="http://www.opengis.net/citygml/building/1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/wfs  http://schemas.opengis.net/wfs/1.1.0/wfs.xsd
http://www.opengis.net/ows-6/utds/0.3 http://services.interactive-
instruments.de/ows6/data/schema/UTDS-CityGML.xsd">

<wfs:Query typeName="utds:Building">

    <ogc:Filter>

        <ogc:PropertyIsEqualTo>

            <ogc:PropertyName>build:class</ogc:PropertyName>

            <ogc:Literal>1020</ogc:Literal>

        </ogc:PropertyIsEqualTo>

    </ogc:Filter>

</wfs:Query>

</wfs:GetFeature>
```

EXAMPLE 3 Query restricted to buildings with a width > 100m

```
<wfs:GetFeature service="WFS" version="1.1.0" xmlns:utds="http://www.opengis.net/ows-
6/utds/0.3" xmlns:wfs="http://www.opengis.net/wfs" xmlns:ogc="http://www.opengis.net/ogc"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/wfs  http://schemas.opengis.net/wfs/1.1.0/wfs.xsd
http://www.opengis.net/ows-6/utds/0.3 http://services.interactive-
instruments.de/ows6/data/schema/UTDS-CityGML.xsd">

<wfs:Query typeName="utds:Building">

    <ogc:Filter>

        <ogc:PropertyIsGreaterThan>

            <ogc:PropertyName>utds:width</ogc:PropertyName>

            <ogc:Literal>100</ogc:Literal>

        </ogc:PropertyIsGreaterThan>

    </ogc:Filter>

</wfs:Query>
```

```
</wfs:GetFeature>
```

EXAMPLE 4 Query restricted to another feature type

```
<wfs:GetFeature service="WFS" version="1.1.0" xmlns:utds="http://www.opengis.net/ows-
6/utds/0.3" xmlns:wfs="http://www.opengis.net/wfs" xmlns:ogc="http://www.opengis.net/ogc"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/wfs  http://schemas.opengis.net/wfs/1.1.0/wfs.xsd
http://www.opengis.net/ows-6/utds/0.3 http://services.interactive-
instruments.de/ows6/data/schema/UTDS-CityGML.xsd">

<wfs:Query typeName="utds:Road"/>

</wfs:GetFeature>
```

EXAMPLE 5 Query restricted to another GML application schema

```
<wfs:GetFeature service="WFS" version="1.1.0" xmlns:app="http://www.example.org/ns"
xmlns:wfs="http://www.opengis.net/wfs" xmlns:ogc="http://www.opengis.net/ogc"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/wfs  http://schemas.opengis.net/wfs/1.1.0/wfs.xsd
http://www.example.org/ns http://wwww.example.org/app.xsd">

<wfs:Query typeName="app:MyFeatureType"/>

</wfs:GetFeature>
```

9 Conclusions

As a result of OWS-6 a validation tool that is able to access data directly from a WFS and validate Schematron rules is available. The tool provides a framework where one can plug in queries and constraints and execute the tests. In addition, a range of constraint types that are relevant for UTDS-CityGML data has been analyzed and documented in this ER.

The general approach to validation in OWS-6 seems to be feasible in general and allows to automatically check schema/business rules beyond XML Schema validation. This includes constraints which require on-demand access to dictionaries, e.g. code list or units dictionaries.

However, there are also limits to such constraints. For example, topology checks would require additional validation software.

More work is needed to identify the constraint types that are sufficient to cover most of the constraints in typical application schemas. The work from OWS-5 and OWS-6 is a basis and includes an initial analysis, but only analyzed at a limited set of application schemas (two application schemas, both work-in-progress).

Bibliography

[1] DuckHawk User Guide v0.3

[2] OGC® OWS-5 GSIP Schema Processing Engineering Report, OGC document 08-078r1 (OGC discussion paper)

[3] OGC® The Specification Model — Modular specifications, OGC document 08-131r2 (OGC Request for Comment)

[4] OGC® OWS-6 UTDS-CityGML Implementation Profile Engineering Report, OGC document 09-037 (OGC internal document)

[5] OGC® Web Feature Service, version 1.1.0, OGC document 04-094 (OGC standard)