- Open Vulnerability and Assessment Language - Element Dictionary

- Schema: Independent Definition
- Version: 4.2
- Release Date: 2 December 2005

The following is a description of the elements, types, and attributes that compose the tests found in Open Vulnerability and Assessment Language (OVAL) that are independent of a specific family or platform. Each test is described in detail and should provide the information necessary to understand what each element and attribute represents. This document is intended for developers and assumes some familiarity with XML. A high level description of the interaction between the different tests and their relationship to the Core Definition Schema is not outlined here.

The OVAL Schema is maintained by The Mitre Corporation and developed by the public OVAL Community. For more information, including how to get involved in the project and how to submit change requests, please visit the OVAL website at http://oval.mitre.org.

Elements

This section describes all the elements that are found within the schema, starting with the root element. Note that in the tables outlining possible attributes and child elements, square brackets [] means that the item is optional. All complex and simple types, along with attribute groups are described later in this document.

<compound_test>

A compound test allows multiple tests (including other compound tests) to be joined together by a logical operator. This provides flexibility in test creation and enables complex tests to be reused, serving as building blocks for future tests. The required operation attribute specifies how to logically combine the numerous subtests of a compound test. Possible values are: AND, OR, XOR. A value of AND means that each subtest must be true for the compound_test to return true. A value of OR means that only one subtest must be true for the compound_test to return true. A value of XOR means that one, and only one, subtest must be true for the compound_test to return true. A compound test extends the testType. Please refer to the "Complex Types" section of this document for a description of the testType.

Extends:	testType
Valid Sections:	[notes], subtest

```
<compound_testid="cmp-0"operation="AND"comment="an example compound test">
    <oval:notes>
```

```
            <oval:note>This is an example compound test. It ANDs together the results of
            three separate tests, one of which is negated.</oval:note>
        </oval:notes>
        <subtesttest_ref="wrt-0"/>
        <subtesttest_ref="wat-0"negate="true"/>
        <subtesttest_ref="cmp-1"/>
    </compound_test>
```

<subtest>

The subtest element specifies a particular test to be referenced. The required test_ref attribute accomplishes this by linking to a valid test id. The optional 'negate' attribute signifies that the result of an individual test should be negated during analysis. For example, consider a test that returns TRUE if a specific patch is installed. By negating this test, it now analyzes to TRUE if the patch is NOT installed.

Parent Test:	Compound Test
Cardinality:	1-n
Content:	none

Text File Content Test

<textfilecontent_test>

The textfilecontent test looks at the contents of a text file (aka a configuration file) by looking at individual lines.

Extends:	standardTestType
Valid Sections:	notes, object, data

```
    <textfilecontent_testid="tft-0"check="all"comment="the enable parameter in helpctr.txt is set to
        <oval:notes>
            <oval:note>This is an example test. It is meant to give a short overview of the test a
            every possible child element.</oval:note>
        </oval:notes>
        <object>
            <path>
                <componenttype="registry_value">HKEY_LOCAL_MACHINE\SOFTWAR
                NT\CurrentVersion\SystemRoot</component>
                <componenttype="literal">\system32\helpctr.txt</component>
            </path>
```

```
        <lineoperator="pattern match">enable = (true|false)</line>
    </object>
    <dataoperation="AND">
        <subexpressionoperator="equals">true</subexpression>
    </data>
</textfilecontent_test>
```

object section

<path>

Specifies the absolute path to a file on the machine. This path can be created from multiple components that are added together. When a pattern match operator is used, the corresponding regular expression is matched against the set of absolute path strings. These string would not include the '.' and '..' notations. This means that a '.*' component of a regular expression will not only match all files in the specified directories, but all subdirectories, their subdirectories, etc.

Parent Test:	Text File Content Test
Cardinality:	1
Content:	none
Valid Datatypes:	component
Valid Operators:	equals, not equal, pattern match

<line>

The line element represents a line in the file and is represented using a regular expression.

Parent Test:	Text File Content Test
Cardinality:	1
Content:	string
Valid Datatypes:	string
Valid Operators:	pattern match

data section

<subexpression>

Each subexpression in the regular expression of the line element is then tested against the value specified in the subexpression element.

Parent Test:	Text File Content Test
Cardinality:	0-n
Content:	string
Valid Datatypes:	string
Valid Operators:	equals, not equal, pattern match

<unknown_test>

An unknown test acts as a placeholder for tests whose implementation is unknown. Any information that is known about the test should be held in the notes child element that is available through the extension of the abstract test element. An unknown test extends the testType. Please refer to the "Complex Types" section of this document for a description of the testType.

Extends:	TestType
Valid Sections:	[notes]

```
<unknown_testid="ukn-0"comment="an example unknown test">
        <oval:notes>
                <oval:note>This is an example test. A description about the desired test would
                go here including what is unknown about it.</oval:note>
        </oval:notes>
</unknown_test>
```

<variable_test>

A variable test allows the value of a variable to be compared to a defined value. An example use would be to validate that a variable being passed in from an external source falls within a specified range.

Extends:	TestType
Valid Sections:	[notes], item

```
<variable_testid="vct-0"operation="AND"comment="an example variable test">
```

```
        <itemvariable="var-3"datatype="int"operator="greater than">6</item>
        <itemvariable="var-3"datatype="int"operator="less than"var_ref="var-6"/>
</variable_test>
```

XML File Content Test

<xmlfilecontent_test>

The xmlfilecontent test uses Xpath to explore the contents of an xml file. The value element checks the value of the nodes found.

Extends:	standardTestType
Valid Sections:	notes, object, data

```
<xmlfilecontent_testid="xft-0"check="none exist"comment="there does not exists an
Andrew object in fred.xml">
        <oval:notes>
                <oval:note>This is an example test. It is meant to give a short overview of the
                test and might not contain every possible child element.</oval:note>
        </oval:notes>
        <object>
                <path>
                        <componenttype="literal">c:\fred.xml</component>
                </path>
                <xpath>/people/name</xpath>
        </object>
        <dataoperation="AND">
                <value_ofoperator="equals">Andrew</value_of>
        </data>
</xmlfilecontent_test>
```

object section

<path>

Specifies the absolute path to a file on the machine. This path can be created from multiple components that are added together. When a pattern match operator is used, the corresponding regular expression is matched against the set of absolute path strings. These string would not include the '.' and '..' notations.

This means that a '.*' component of a regular expression will not only match all files in the specified directories, but all subdirectories, their subdirectories, etc.

Parent Test:	XML File Content Test
Cardinality:	1
Content:	none
Valid Datatypes:	component
Valid Operators:	equals, not equal, pattern match

<xpath>

Specifies an Xpath expression describing the nodes to look at.

Parent Test:	XML File Content Test
Cardinality:	1
Content:	string
Valid Datatypes:	string
Valid Operators:	equals, not equal, pattern match

data section

<value_of>

The value element checks the value of the nodes found.

Parent Test:	XML File Content Test
Cardinality:	0-1
Content:	string
Valid Datatypes:	string
Valid Operators:	equals, not equal, pattern match

Complex Types

This section describes any global complex types defined in the schema. These types can be instantiated by elements in this schema as well as elements in other schemas. Note that in the tables outlining possible attributes and child elements, square brackets [] means that the item is optional.

-- componentType --

The componentType allows a value to be obtained by combining pieces from different sources. Each string defined by the different component elements is concatenated together to form the final string used. Each child component element has an attribute called type. The value of this attribute determines where to get the string used to build the file path. A type of literal means to use the value of the child component element as is, and to just concatenated it to the other strings. If a pattern match operator has been specified with a componentType, then the final string should be thought of as the pattern to test. As of Version 4 of the OVAL schema, pattern match can not be specified for the idividual components.

Extends:	oval:subtestBaseType
Attributes:	(includes oval:subtestAttributes)
Content:	none
Child Elements:	component