- Open Vulnerability and Assessment Language - Element Dictionary

- Schema: Core Results
- Version: 5.3
- Release Date: 6/22/2007 11:18:49 AM

The following is a description of the elements, types, and attributes that compose the core schema for encoding Open Vulnerability and Assessment Language (OVAL) Results. Each of the elements, types, and attributes that make up the Core Results Schema are described in detail and should provide the information necessary to understand what each object represents. This document is intended for developers and assumes some familiarity with XML. A high level description of the interaction between these objects 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.

< oval_results >

The oval_results element is the root of an OVAL Results Document. Its purpose is to bind together the four major sections of a results file - generator, directives, oval_definitions, and results - which are the children of the root element. It must contain exactly one generator section, one directives section, and one results section.

Child Elements	Type	MinOccurs	MaxOccurs
generator	oval:GeneratorType	1	1
directives	oval-res:DirectivesType	1	1
oval-def:oval_definitions	n/a	0	1
results	oval-res:ResultsType	1	1
ds:Signature	n/a	0	1

== DirectivesType ==

The DirectivesType complex type presents flags describing what information has been included in the results file. There are six possible results (true, false, unknown, error, not evaluated, and not applicable) for an evaluation of an OVAL Definition. The directives state which of these results are being reported in the results file. For example, a results file dealing with vulnerabilities might only present to the user the definitions that returned a true result meaning the vulnerability exists. A different result file dealing with compliance definitions might want to report the results of all definitions except those not evaluated.

Child Elements	Type	MinOccurs	MaxOccurs
definition_true	oval-res:DirectiveType	1	1
definition_false	oval-res:DirectiveType	1	1
definition_unknown	oval-res:DirectiveType	1	1
definition_error	oval-res:DirectiveType	1	1
definition_not_evaluated	oval-res:DirectiveType	1	1
definition_not_applicable	oval-res:DirectiveType	1	1

== DirectiveType ==

Each directive determines whether or not certain results are included in the results file. The required reported attribute controls this by providing a true or false for the specific directive. The optional content attribute controls how much information about the specific result is provided. For example, thin conten would only be the id of the definition and the result, while a full content set would be the definition id with the result along with all the test ids and their results. Please refer to the contentEnumeration for details about the different content options.

Attributes:

- reported xsd:boolean (required)
- content oval-res:ContentEnumeration (optional -- default='full')

== ResultsType ==

The ResultsType complex type is a container for one or more system elements. Each system element defines the results associated with an individual system. Please refer to the description of SystemType for more information about an individual system element.

Child Elements	Type	MinOccurs	MaxOccurs
system	oval-res:SystemType	1	unbounded

== SystemType ==

The SystemType complex type holds the evaluation results of the definitions and tests, as well as a copy of the OVAL System Characteristics used to perform the evaluation. The definitions section holds the results of the definitions and the tests section holds the results of the tests. The oval_system_characteristics section is a copy of the system characteristics file used to perform the evaluation of the OVAL Definitions. Note that the oval_definitions part of the system characteristics file should be left out as the definition information has already been included at the top of the results file.

Child Elements	Type	MinOccurs	MaxOccurs
definitions	oval-res:DefinitionsType	0	1
tests	oval-res:TestsType	0	1
oval-sc:oval_system_characteristics	n/a	1	1

== DefinitionsType ==

The DefinitionsType complex type is a container for one or more definition elements. Each definition element holds the result of the evaluation of an OVAL Definition. Please refer to the description of DefinitionType for more information about an individual definition element.

Child Elements	Type	MinOccurs	MaxOccurs
definition	oval-res:DefinitionType	1	unbounded

== DefinitionType ==

The DefinitionType complex type holds the result of the evaluation of an OVAL Definition. The message element holds an error message or some other string that the analysis engine wishes to pass along. In addition, the optional criteria element provides the results of the individual pieces of the criteria. Please refer to the description of the CriteriaType for more information.

The required definition_id attribute is the OVAL id of the definition. The required version attribute is the specific version of the OVAL Definition used during analysis. The optional variable_instance attrbitue is a unique id that differentiates every unique instance (based on the value of variables used) of a definition in the OVAL Results file. Languages that include OVAL might reference the same definition multiple times. Each time a different set of values is supplied for the variables, a new instance of the definition result must be created. (definitions that do not use variables can only have one unique instance) The inclusion of a unique instance identifier will allow the OVAL results file to report the correct result of a definition for each combination of supplied values. The required result attribute holds the result of the evaluation. Please refer to the description of the resultEnumeration for details about the different result values.

Attributes:

- definition_id oval:DefinitionIDPattern (required)
- version xsd:nonNegativeInteger (required)
- variable_instance xsd:nonNegativeInteger (optional -- default='1')
- result oval-res:ResultEnumeration (required)

Child Elements	Type	MinOccurs	MaxOccurs
message	oval:MessageType	0	unbounded

criteria	oval-res:CriteriaType	0	1

== CriteriaType ==

The CriteriaType complex type describes the high level container for all the tests and represents the meat of the definition. Each criteria can contain other criteria elements in a recursive structure allowing complex logical trees to be constructed. Each referenced test is represented by a criterion element. Please refer to the description of the CriterionType for more information about and individual criterion element. The optional extend_definition element allows existing definitions to be included in the criteria. Refer to the description of the ExtendDefinitionType for more information.

The required operator attribute provides the logical operator that binds the different statements inside a criteria together. The optional negate attribute signifies that the result of an extended definition should be negated during analysis. For example, consider a definition that evaluates TRUE if a certain software is installed. By negating the definition, it now evaluates to TRUE if the software is NOT installed. The required result attribute holds the result of the evaluation of the criteria. Note that this would be after any negation operation has been applied. Please refer to the description of the resultEnumeration for details about the different result values.

Attributes:

- operator oval:OperatorEnumeration (required)
- negate xsd:boolean (optional -- default='false')
- result oval-res:ResultEnumeration (required)

Child Elements	Type	MinOccurs	MaxOccurs
criteria	oval-res:CriteriaType		
criterion	oval-res:CriterionType		
extend_definition	oval-res:ExtendDefinitionType		

== CriterionType ==

The CriterionType complex type identifies a specific test that is included in the definition's criteria.

The required test_id attribute is the actual id of the included test. The required version attribute is the specific version of the OVAL Test used during analysis. The optional variable_instance attribute differentiates between unique instances of a test. This can happen when a test includes a variable reference and different values for that variable are used by different definitions. The optional negate attribute signifies that the result of an individual test should be negated during analysis. For example, consider a test that evaluates to TRUE if a specific patch is installed. By negating this test, it now evaluates to TRUE if the patch is NOT installed. The required result attribute holds the result of the evaluation. Please refer to the description of the resultEnumeration for details about the different result values.

Attributes:

- test_ref oval:TestIDPattern (required)
- version xsd:nonNegativeInteger (required)
- variable_instance xsd:nonNegativeInteger (optional -- default='1')

- negate xsd:boolean (optional -- default='false')
- result oval-res:ResultEnumeration (required)

== ExtendDefinitionType ==

The ExtendDefinitionType complex type identifies a specific definition that has been extended by the criteria.

The required definition_id attribute is the actual id of the extended definition. The required version attribute is the specific version of the OVAL Definition used during analysis. The optional variable_instance identifier is a unique id that differentiates every unique instance of a definition in the OVAL Results file based on the combination of variable values used. Languages that include OVAL might reference the same definition multiple times. Each time a different set of values is supplied for the variables, a new instance of the definition must be created. (definitions that do not use variables can only have one unique instance) The inclusion of a unique instance identifier will allow the OVAL results file to report the correct result of a definition for each combination of supplied values. The optional negate attribute signifies that the result of an extended definition should be negated during analysis. For example, consider a definition that evaluates TRUE if a certain software is installed. By negating the definition, it now evaluates to TRUE if the software is NOT installed. The required result attribute holds the result of the evaluation. Please refer to the description of the resultEnumeration for details about the different result values.

Attributes:

- definition_ref oval:DefinitionIDPattern (required)
- version xsd:nonNegativeInteger (required)
- variable_instance xsd:nonNegativeInteger (optional -- default='1')
- negate xsd:boolean (optional -- default='false')
- result oval-res:ResultEnumeration (required)

== TestsType ==

The TestsType complex type is a container for one or more test elements. Each test element holds the result of the evaluation of an OVAL Test. Please refer to the description of TestType for more information about an individual test element.

Child Elements	Type	MinOccurs	MaxOccurs
test	oval-res:TestType	1	unbounded

== TestType ==

The TestType complex type provides a reference to every item that matched the object section of the original test as well as providing an overall test result based on these items. The optional message element holds an error message or some other string that the analysis engine wishes to pass along. The optional tested_variable elements hold the value of each variable used by the test during evaluation. This includes the values used in both OVAL Objects and OVAL States. If a variable represents an array of values, then multiple tested_variable elements would exist with the same variable_id attribute. Please refer to the description of TestedVariableType

for more information.

The required test_id attribute identifies the test, and must conform to the format specified by the testidPattern simple type. The required version attribute is the specific version of the OVAL Test used during analysis. The optional variable_instance attribute differentiates between unique instances of a test. This can happen when a test includes a variable reference and different values for that variable are used by different definitions. The required check_existance attribute is used in determining the overall result by signifying how many matching items must exist. The optional check_state attribute is also used in determining the overall result and is used to define how many of the matching items must meet the supplied OVAL State. (For example: Should the test check that all files match a specified version or that at least one file matches the specified version?) The valid values for both the check_existance and check_state attribute are explained in the simple type declarations found in the common schema.

The required result attribute holds the result of the evaluation. Please refer to the description of the resultEnumeration for details about the different result values. The overall result of the test is determined by the results of each matching item and the different check attributes. If you are using an OVAL System Characteristics file, then the following rules apply: If a collected_object is not found the result for the OVAL Test should be "unknown". When the flag attribute of the collected_object is “error” the result for the OVAL Test should be set to “error”. When the flag attribute is “complete” the result of the test is determined by first evaluating the check_existance attribute on the test and then evaluating the check_state attribute. The check_state attribute only needs to be considered if the result of evaluating the check_existance attribute is “true”. When the flag attribute is “does not exist” the result of the test is determined by examining the check_existance attribute’s value, if the check_existance attribute is “none_exist” or “any_exist” the Test should evaluate to “true”, for all other values of the check_existance attribute the Test should evaluate to “false”. When the flag attribute is “not collected” the result of the test should be set to “unknown”. When the flag attribute is “not applicable” the result for the Test should be set to “not applicable”. When the flag attribute is “incomplete” it may not be possible to determine a result other than “unknown” for the test. However, in some cases it will be possible to determine a result. These cases are: 1) when the check_existance attribute on a test is set to “none_exist” and the collected object has 1 or more item references with a status of "exists" a result of “false” should be reported 2) when the check_existance attribute is set to “only_one_exists” the collected object has more than 1 item reference with a status of "exists" a result of “false” should be reported 3) if after evaluating the check_existance attribute a non "true" result has not been determined the check_state attribute must be considered. As follows: 3a) if the check_state attribute evaluation results in "false" then the OVAL Test result should be set to "false" 3b) if the check_state attribute is set to "at_least_one_satisfies" and its evaluation results in "true" the OVAL Test result should be set to "true". For all other cases when the collected_object flag is "incompete" the OVAL Test result should be set to "unknown".

Attributes:

- test_id — oval:TestIDPattern — (required)
- version — xsd:nonNegativeInteger — (required)
- variable_instance — xsd:nonNegativeInteger — (optional -- default='1')
- check_existence — oval:ExistenceEnumeration — (optional -- default='at_least_one_exists')
- check — oval:CheckEnumeration — (required)
- result — oval-res:ResultEnumeration — (required)

Child Elements	Type	MinOccurs	MaxOccurs
message	oval:MessageType	0	unbounded
tested_item	oval-res:TestedItemType	0	unbounded
tested_variable	oval-res:TestedVariableType	0	unbounded

== TestedItemType ==

The TestedItemType complex type holds a reference to each system characteristic item that matches the object specified in a test. Details of the item can be found in the oval_system_characteristics section of the OVAL Results file by using the required item_id. The optional message element holds an error message or some other string that the analysis engine wishes to pass along. The required result attribute holds the result of the evaluation of the individual item as it relates to the state specified by the test. Please refer to the description of the resultEnumeration for details about the different result values.

Attributes:

- item_id oval:ItemIDPattern (required)
- result oval-res:ResultEnumeration (required)

Child Elements	Type	MinOccurs	MaxOccurs
message	oval:MessageType	0	unbounded

== TestedVariableType ==

The TestedVariableType complex type holds the value to a variable used during the evaluation of a test. Of special importance are the values of any external variables used since these values are not captured in either the definition or system characteristic files. If a variable is represented by an array of values, then multiple elements of TestedVariableType, each with the same variable_id attribute, would exist. The required variable_id attribute is the unique id of the variable that was used.

Attributes:

- variable_id oval:VariableIDPattern (required)

Simple Content	xsd:anySimpleType

-- ContentEnumeration --

Defines the valid values for the directives controlling the expected content of the results file. The specific content that is expected with each value is defined by a style sheet that complements the OVAL Results Schema. Please refer to these style sheets for more information.

Value	Description
thin	A value of 'thin' means only the minimal amount of information will be provided. This is the id associated with an evaluated OVAL Definition and the result of the evaluation. The criteria child element of a definition should not be present when providing thin results. In

	addition, system characteristic information for the objects used by the given definition should not be presented.
full	A value of 'full' means that very detailed information will be provided allowing in-depth reports to be generated from the results. In addition to the results of the evaluated definition, the results of all extended definitions and tests included in the criteria as well as the actual information collected off the system will be presented.

-- ResultEnumeration --

Define acceptable result values for the evaluation of an OVAL Definition or an OVAL Test.

Value	Description
true	When evaluating a definition or test, a result value of 'true' means that the characteristics being evaluated match the information represented in the system characteristic file.
false	When evaluating a definition or test, a result value of 'false' means that the characteristics being evaluated do not match the information represented in the system characteristic file.
unknown	When evaluating a definition or test, a result value of 'unknown' means that the characteristics being evaluated can not be found in the system characteristic file. (or the characteristics can be found but collected object flag is 'not collected') For example, assume you have a definition that tests a file, but when you look at the system characteristic file, data pertaining to that file can not be found. The lack of an object (in the collected_object section) for this file in the SC file means that no attempt was made to even try and collect information about the file. So you do not know what the result would be if it was collected. Note that finding a collected_object element in the system characteristic file is not the same as finding a matching element of the system. When evaluating an OVAL Test, the lack of a matching object on a system (for example, file not found) does not mean an unknown result since part of a test in OVAL is about existence. In this case the result would be 'false'.
error	When evaluating a definition or test, a result value of 'error' means that the characteristics being evaluated exist in the system characteristic file but there was an error either collecting information or in performing anaylsis. For example, if there was an error returned by an api

	when trying to determine if an object exists on a system. Another example would be: xsi:nil might be set on an object entity, but then the entity is compared to a state entity with a value, thus producing an error.
not evaluated	When evaluating a definition or test, a result value of 'not evaluated' means that a choice was made not to evaluate the given definition or test. The actual result is in essence unknown since if evaluation had occurred it could have been either true or false.
not applicable	When evaluating a definition or test, a result value of 'not applicable' means that the definition or test being evaluated is not valid on the given platform. For example, trying to collect Linux RPM information on a Windows system.