- Open Vulnerability and Assessment Language - Element Dictionary

- Schema: Core System Characteristics
- Version: 5.1
- Release Date: 6 November 2006

The following is a description of the elements, types, and attributes that compose the core schema for encoding Open Vulnerability and Assessment Language (OVAL) System Characteristics. The Core System Characteristics Schema defines all operating system independent objects. These objects are extended and enhanced by individual family schemas, which are described in separate documents. Each of the elements, types, and attributes that make up the Core System Characteristics 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_system_characteristics >

The system_characteristics element is the root of an OVAL System Characteristics Document, and must occur exactly once. Its purpose is to bind together the four major sections of a system characteristics file - generator, system_info, collected_objects, and system_data - which are the children of the oval_system_characteristics element. The generator section must be present and provides information about when the system characteristics file was compiled and under what version. The required system_info element is used to record information about the system being described. The collected_objects section provides a listing of all the objects used to generate this system characteristics file. The optional collected_objects section is used to associated the ids of the OVAL Objects collected with the system characteristics items that have been defined. The optional system_data section defines the specific characteristics that have been collected from the system. The optional Signature element allows an XML Signature as defined by the W3C to be attached to the document. This allows authentication and data integrity to be provided to the user. Enveloped signatures are supported. More information about the official W3C Recommendation regarding XML digital signatures can be found at http://www.w3.org/TR/xmldsig-core/.

Child Elements	Type	MinOccurs	MaxOccurs
generator	oval:GeneratorType	1	1
system_info	oval-sc:SystemInfoType	1	1
collected_objects	oval-sc:CollectedObjectsType	0	1
system_data	oval-sc:SystemDataType	0	1
ds:Signature	n/a	0	1

== SystemInfoType ==

The SystemInfoType complex type specifies general information about the system that data was collected from, including information that can be used to identify the system. The required os_name and os_version elements describe the operating system of the machine the data was collected from. The required architecture element describes the hardware architecture type of the system data was collected from. The required primary_host_name element is the primary host name of the machine the data was collected from. And the required interfaces element outlines the network interfaces that exist on the system. See the description of the InterfacesType complex type for more information. Note that the high level interfaces is required due to the inclusion of the xsd:any tag that follows it. The interfaces tag can be empty if no single interface is present.

Additional system information is also allowed although it is not part of the official OVAL Schema. Individual organizations can place system information that they feel is important and these will be skipped during the validation. All OVAL really cares about is that the required system information items are there.

Child Elements	Type	MinOccurs	MaxOccurs
os_name	xsd:string	1	1
os_version	xsd:string	1	1
architecture	xsd:string	1	1
primary_host_name	xsd:string	1	1
interfaces	oval-sc:InterfacesType	1	1

== InterfacesType ==

The InterfacesType complex type is a container for zero or more interface elements. Each interface element is used to describe an existing network interface on the system. Please refer to the description of the InterfaceType for more information.

Child Elements	Type	MinOccurs	MaxOccurs
interface	oval-sc:InterfaceType	0	unbounded

== InterfaceType ==

The InterfaceType complex type is used to describe an existing network interface on the system. This information can help identify a specific system on a given network. The required interface_name element is the name of the interface while the required ip_address and mac-address elements hold the respective addresses for the specific interface.

Child Elements	Type	MinOccurs	MaxOccurs
interface_name	xsd:string	1	1
ip_address	xsd:string	1	1

mac_address	xsd:string	1	1

== CollectedObjectsType ==

The CollectedObjectsType complex type states all the objects that have been collected by the system characteristics file. The details of each object are defined by the global OVAL object that is identified by the id.

Child Elements	Type	MinOccurs	MaxOccurs
object	oval-sc:ObjectType	1	unbounded

== ObjectType ==

The ObjectType complex type provides a reference between items collected and a related global OVAL Object. The message element holds an error message or some other string that the data collection engine wishes to pass along. The optional variable_value elements define the actual value(s) used during data collection of any variable referenced by the object (as well as any object referenced via a set element). An OVAL Object that includes a variable maybe have a different set of matching items depending on the value given assigned to the variable. A tool that is given an OVAL System Characteristics file in order to analyze an OVAL Definition needs to be able to determine the exact instance of an object to use based on the variable values supplied. If a variable represents an array of values, then multiple variable_value elements would exist with the same variable_id attribute. Each reference elements link the collected item found by the data collection engine and the global OVAL Object. A global OVAL Object my have multiple matching items on a system. For example a global file object that is a pattern match might match 10 different files on a specific system. In this case, there would be 10 reference elements, one for each of the files found on the system.

If an OVAL Object does not exist on the system, then an object element is still provided but with the flag attribute set to 'does not exist' and with no reference child elements. This shows that the object was looked for but not found on the system. If no object element is written in this case, users of the system characteristics file will not know whether the object was not found or no attempt was made to collect it.

The required id attribute is the id of the global OVAL Object. The required version attribute is the specific version of the global OVAL Object that was used by the data collection engine. The version is necessary so that analysis using a system characteristics file knows exactly what was collected. The optional variable_instance identifier is a unique id that differentiates every unique instance of an object. Languages that include OVAL might reference the same definition multiple times. Each time a different set of values is supplied for the variables, resulting in multiple instances of an object to be defined by the OVAL System Characteristics file. (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 item associated with an object for each combination of supplied values. The optional comment attribute provides a short description of the object. The required flag attribute holds information regarding the success of the data collection. For example, if there was an error looking for items that match the object specification, then the flag would be 'error'. Please refer to the description of FlagEnumeration for details about the different flag values.

Attributes:

- id oval:ObjectIDPattern (required)

- version xsd:integer (required)
- variable_instance xsd:integer (optional -- default='1')
- comment xsd:string (optional)
- flag oval-sc:FlagEnumeration (required)

Child Elements	Type	MinOccurs	MaxOccurs
message	oval:MessageType	0	unbounded
variable_value	oval-sc:VariableValueType	0	unbounded
reference	oval-sc:ReferenceType	0	unbounded

== VariableValueType ==

The VariableValueType complex type holds the value to a variable used during the collection of an object. The required variable_id attribute is the unique id of the variable being identified.

Attributes:

- variable_id oval:VariableIDPattern (required)

Simple Content	xsd:anySimpleType

== ReferenceType ==

The ReferenceType complex type specifies an item in the system characteristics file. This reference is used to link global OVAL Objects to specific items.

Attributes:

- item_ref oval:ItemIDPattern (required)

== SystemDataType ==

The SystemDataType complex type is a container for one or more item elements. Each item defines a specific piece of data on the system.

Child Elements	Type	MinOccurs	MaxOccurs
oval-sc:item	n/a	1	unbounded

< item >

The abstract item element holds information about a specific item on a system. An item might be a file, a rpm, a process, etc. This element is extended by the different component schemas through substitution groups. Each item represents a unique instance of an object as specified by an OVAL Object. For example, a single file or a single user. Each item my be referenced by more than one object in the collected object section. Please refer to the description of ItemType for more details about the information stored in items.

== ItemType ==

The ItemType complex type specifies an optional message element that is used to pass things like error messages during data collection to a tool that will utilize the information. The required id attribute is an unique (to the file) identifier that allows the specific item to be referenced. The optional object_ref attribute allows the item to be linked to an object declaration used to identify the item being collected. The required status attribute holds information regarding the success of the data collection. For example, if an item exists on the system then the status would reflect this with a value of 'exists'. If there was an error collecting a any information about an item that is known to exist, then the status would be 'error'. An error specific to a particular entity should be addressed at the entity level and not the item level. Note that an item should not have a status of 'does not exist' as there simply would be no item element in the system characteristics file for this case. If part of the object declaration does not exist on the system (say a path exists but the filename does not), then no item should be written for this object and the object element in the collected_object section should have a flag value of 'does not exist'.

Attributes:

- id oval:ItemIDPattern (required)
- status oval-sc:StatusEnumeration (optional -- default='exists')

Child Elements	Type	MinOccurs	MaxOccurs
message	oval:MessageType	0	1

-- FlagEnumeration --

The FlagEnumeration simple type defines the valid flags associated with a collected object. These flags are meant to provide information about how the specified object was handled by the data collector. In order to evaluate an OVAL Definition, information about the defined objects need to be available. The flags help detail the success of trying to collect information related to these objects.

Value	Description
error	A flag of 'error' says that there was an error trying to identify objects on the system that match the specified object declaration. This flag is not meant to be used when there was an error retrieving a specific attribute, but rather when it could not be determined if an instance of the object exists or not. Any error in retrieving specific attributes should be represented by setting the status of that specific attribute to 'error'.

complete	Every matching item on the system has been identified and is represented in the system characteristics file. It can be assumed that no additional matching items exist on the system.
incomplete	An instance of the specified object exists on the system, but only some of the matching items have been identified and are represented in the system characteristics file. It is unknown if additional matching items also exist. Note that with a flag of incomplete, each item that has been identified matches the object declaration, but additional items might also exist on the system.
does not exist	A flag of 'does not exist' means that no matching item was found on the system.
not collected	An attempt to collect information on items matching the object was not made. An object with this flag will produce an 'unknown' result during analysis since it is unknown if a matching items exists on the system or not. This is different from an 'error' flag since with an 'error' flag an attempt to collect information was made. With the 'not collected' flag, no attempt was made.
not applicable	The specified object is not applicable to the system being characterized. An example would be trying to collect objects related to a Red Hat system off a Windows system.

-- StatusEnumeration --

The StatusEnumeration simple type defines the valid status messages associated with collection of specific information associated with an item.

Value	Description
error	A status of 'error' says that there was an error collecting information associated with an item as a whole or a specific entity.
exists	A status of 'exists' says that the item or specific piece of information exists on the system and has been collected.
does not exist	A status of 'does not exist' says that the item or specific piece of information does not exist and therefore has not been collected. This status assumes that an attempt was made to collect the information, but the information just doesn't exist. This can happen when a certain entity is only pertinent to particular instances, or when xsi:nil is used to refer to a higher level object.
not collected	A status of 'not collected' says that no attempt was made to collect the item or specific piece of information so it is

	unknown what the value is and if it even exists.

== EntityItemBaseType ==

The EntityItemBaseType complex type is an abstract type that defines the default attributes associated with every entity. The optional datatype determines the type of data expected. (the default datatype is 'string') The optional status attribute holds information regarding the success of the data collection. For example, if there was an error collecting a particular piece of data, then the status would be 'error'.

Attributes:

- datatype oval:DatatypeEnumeration (optional -- default='string')
- status oval-sc:StatusEnumeration (optional -- default='exists')

Simple Content	xsd:anySimpleType

== EntityItemAnyType ==

The EntityItemAnyType type is extended by the entities of an individual item. This type provides uniformity to each entity by including the attributes found in the EntityItemBaseType. This specific type describes any simple data.

Attributes:

Simple Content	oval-sc:EntityItemBaseType

== EntityItemBinaryType ==

The EntityItemBinaryType type is extended by the entities of an individual item. This type provides uniformity to each entity by including the attributes found in the EntityItemBaseType. This specific type describes simple binary data. The empty string is also allowed for cases where there was an error in the data collection of an entity and a status needs to be reported.

== EntityItemBoolType ==

The EntityItemBoolType type is extended by the entities of an individual item. This type provides uniformity to each entity by including the attributes found in the EntityItemBaseType. This specific type describes simple boolean data. The empty string is also allowed for cases where there was an error in the data collection of an entity and a status needs to be reported.

== EntityItemFloatType ==

The EntityItemFloatType type is extended by the entities of an individual item. This type provides uniformity to

each entity by including the attributes found in the EntityItemBaseType. This specific type describes simple float data. The empty string is also allowed for cases where there was an error in the data collection of an entity and a status needs to be reported.

== EntityItemIntType ==

The EntityItemIntType type is extended by the entities of an individual item. This type provides uniformity to each entity by including the attributes found in the EntityItemBaseType. This specific type describes simple integer data. The empty string is also allowed for cases where there was an error in the data collection of an entity and a status needs to be reported.

== EntityItemStringType ==

The EntityItemStringType type is extended by the entities of an individual item. This type provides uniformity to each entity by including the attributes found in the EntityItemBaseType. This specific type describes simple string data.