Security Automation and Continuous Monitoring M. Cokus
Internet-Draft D. Haynes
Intended status: Informational D. Rothenberg
Expires: March 11, 2017 The MITRE Corporation
J. Gonzalez
Department of Homeland Security
September 7, 2016

OVAL(R) System Characteristics Model
draft-rothenberg-sacm-oval-sys-char-model-01

Abstract

This document specifies Version 5.11.1 of the OVAL System Characteristics Model which provides a framework for representing low-level system configuration information that can be extended to support platform-specific constructs.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 11, 2017.

Table of Contents

1. Introduction

The Open Vulnerability and Assessment Language (OVAL) [OVAL-WEBSITE] is an international, information security community effort to standardize how to assess and report upon the machine state of systems. For over ten years, OVAL has been developed in collaboration with any and all interested parties to promote open and publicly available security content and to standardize the representation of this information across the entire spectrum of security tools and services.

OVAL provides an established framework for making assertions about a system's state by standardizing the three main steps of the assessment process: representing the current machine state; analyzing the system for the presence of the specified machine state; and representing the results of the assessment which facilitates collaboration and information sharing among the information security community and interoperability among tools.

This draft is part of the OVAL contribution to the IETF SACM WG that standardizes the representation of the current machine state of a system. It is intended to serve as a starting point for the endpoint posture assessment data modeling needs of SACM specifically Posture Attributes.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. OVAL System Characteristics Model

The OVAL System Characteristics Model is used to represent low-level, system settings that describe the current state of a system. The OVAL System Characteristics Model serves as a basis for extension to create platform-specific, low-level configuration information models.

Property	Type	Count	Description
generator	oval:GeneratorType	1	Information regarding the generation of the OVAL System Characteristics. The timestamp property of the

				generator MUST represent the time at which the system state information was collected.
system_info	SystemInfoType	0..*		Information used to identify the system under test.
collected_objects	CollectedObjectsType	0..1		Contains the mapping between OVAL Objects defined in the OVAL Definitions and the OVAL Items that were collected from the system under test.
system_data	SystemDataType	0..1		Contains the OVAL Items that were collected from the system under test.
signature	ext:Signature	0..1		Mechanism to ensure the integrity and authenticity of the OVAL System Characteristics content.

Table 1: oval_system_characteristics Construct

2.1. SystemInfoType

The SystemInfoType defines the basic identifying information associated with the system under test.

Property	Type	Count	Description
os_name	string	0..1	The operating system running on the system under test.
os_version	string	1	The version of the operating system running on the system under test.
architecture	string	1	The hardware architecture type of the system under test.
primary_host_name	string	1	The primary host name of the system under test.
interfaces	any	0..*	The network interface(s) present on the system under test.
extension_point	any	0..*	An extension point that allows for the inclusion of any additional identifying information associated with the system under test.

Table 2: SystemInfoType Construct

2.2. InterfacesType

The InterfacesType provides a container for zero or more interfaces.

Property	Type	Count	Description
interface	InterfaceType	0..*	One or more interfaces.

Table 3: InterfacesType Construct

2.3. InterfaceType

The InterfaceType defines the information associated with a network interface on the system under test. This information may help to identify a specific system on a network.

Property	Type	Count	Description
interface_name	string	1	The name of the interface.
ip_address	string	1	The Internet Protocol (IP) address of the interface.
mac_address	string	1	The Media Access Control (MAC) address of the interface. MAC addresses MUST be formatted according to IEEE 802-2001 Section 9.2.1 [IEEE-STD-802-2001].

Table 4: InterfaceType Construct

2.4. CollectedObjectsType

The CollectedObjectType is a container for one or more objects of type ObjectType that were used for data collection on the system under test.

2.5. ObjectType

The ObjectType provides a mapping between an OVAL Object, defined in content based on the OVAL Definitions Model, and the OVAL Items collected on the system under test.

Property	Type	Count	Description
id	oval:ObjectIDPattern	1	The globally unique identifier of an OVAL Object.
version	unsigned integer	1	The version of the globally unique OVAL Object.
variable_instance	unsigned integer	0..1	The unique identifier that differentiates between each unique instance of an OVAL Object.

			If an OVAL Object utilizes an OVAL Variable, a unique instance of each OVAL Object must be created for each OVAL Variable value. Default Value: '1'
comment	string	0..1	The documentation associated with the OVAL Object referenced by the id property.
flag	oval:FlagEnumeration	1	The outcome associated with OVAL Item collection.
message	oval:MessageType	0..*	Any messages that are relayed from a tool at run-time.
variable_value	VariableValueType	0..*	The value(s) associated with the variable(s) used by the OVAL Object referenced by the id property.

reference	ReferenceType	0..*	The identifiers of OVAL Items collected by the OVAL Object referenced by the id property.

Table 5: ObjectType Construct

2.6. VariableValueType

The VariableValueType identifies an OVAL Variable and value that is used by an OVAL Object during OVAL Item collection.

Property	Type	Count	Description
variable_id	oval:VariableIDPattern	1	The unique identifier of an OVAL Variable.
value	string	1	A value associated with the OVAL Variable identified by the variable_id property.

Table 6: VariableValueType Construct

2.7. ReferenceType

The ReferenceType identifies an OVAL Item that was collected during OVAL Item collection.

Property	Type	Count	Description
item_ref	oval:ItemIDPattern	1	The unique identifier of an OVAL Item.

Table 7: ReferenceType Construct

2.8. SystemDataType

The SystemDataType provides a container for all of the OVAL Items that were collected on the system under test.

2.9. ItemType

The ItemType is the abstract OVAL Item that defines the common properties associated with all OVAL Items defined in the OVAL System Characteristics OVAL Component Models.

Property	Type	Count	Description
id	oval:ItemIDPattern	1	The unique identifier of an OVAL Item. The id property is unique within a given instantiation of the OVAL System Characteristics Model.
status	StatusEnumeration	0..1	The status property of an OVAL Item conveys the outcome of the system data collection effort. Default Value: 'exists'
message	MessageType	0..50	Any messages that are relayed from a tool at run-time during the collection of an OVAL Item.

Table 8: GeneratorType ItemType

2.10. EntityAttributeGroup

The EntityAttributeGroup defines the properties that are common to all OVAL Item Entities in the OVAL Language.

Property	Type	Count	Description
datatype	oval:DatatypeEnumeration	0..1	The datatype for the entity. Default Value: 'string'
mask	boolean	0..1	Tells the data

				collection that this entity contains sensitive data. Data marked with mask='true' should be used only in the evaluation, and not be included in the results. Note that when the mask property is set to 'true', all child field elements must be masked regardless of the child field's mask attribute value. Default Value: 'false'
status	StatusEnumeration	0..1		The status of the collection for an OVAL Item Entity. Default Value: 'exists'

Table 9: EntityAttributeGroup Construct

2.11. FlagEnumeration

The FlagEnumeration defines the acceptable outcomes associated with the collection of OVAL Items for a specified OVAL Object.

Value	Description
error	This value indicates that an error prevented the determination of the existence of OVAL Items on the system.
complete	This value indicates that every matching OVAL Item on the system has been identified and represented in the OVAL System Characteristics. It can be assumed that no additional matching OVAL Items exist on the system.
incomplete	This value indicates that matching OVAL Items exist on the system, however, only a subset of those matching OVAL Items have been identified and represented in the OVAL System Characteristics. It cannot be assumed that no additional matching OVAL Items exist on the system.
does not exist	This value indicates that no matching OVAL Items were found on the system.
not collected	This value indicates that no attempt was made to collect OVAL Items on the system.
not applicable	This value indicates that the specified OVAL Object is not applicable to the system under test.

Table 10: FlagEnumeration Construct

2.12. StatusEnumeration

The StatusEnumeration defines the acceptable status values associated with the collection of an OVAL Item or the properties of an OVAL Item.

Value	Description
error	This value indicates that there was an error collecting an OVAL Item or a property of an OVAL Item.
exists	This value indicates that an OVAL Item, or a property of an OVAL Item, exists on the system and was collected.
does not exist	This value indicates that an OVAL Item, or a property of an OVAL Item, does not exist on the system.
not collected	This value indicates that no attempt was made to collect an OVAL Item or a property of an OVAL Item.

Table 11: StatusEnumeration Construct

2.13. EntityItemSimpleBaseType

The EntityItemSimpleBaseType is an abstract type that defines a base type for all simple OVAL Item Entities.

Property	Type	Count	Description
attributes	EntityAttributeGroup	1	The standard attributes available to all entities.
value	string	0..1	The value of the entity. An empty string value SHOULD be used when a status other than 'exists' is specified.

Table 12: EntityItemSimpleBaseType Construct

2.14. EntityItemComplexBaseType

The EntityItemComplexBaseType is an abstract type that defines a base type for all complex OVAL Item Entities.

Property	Type	Count	Description
attributes	EntityAttributeGroup	1	The standard attributes available to all entities.

Table 13: EntityItemComplexBaseType Construct

2.15. EntityItemIPAddressType

The EntityItemIPAddressType extends the EntityItemSimpleBaseType and describes an IPv4 or IPv6 IP address or prefix.

Property	Type	Count	Description
datatype	oval:SimpleDatatypeEnumeration	1	Possible values: o 'ipv4_address' o 'ipv6_address' Also allows an empty string value.

Figure 1: EntityItemIPAddressType Construct

2.16. EntityItemIPAddressStringType

The EntityItemIPAddressStringType extends the EntityItemSimpleBaseType and describes an IPv4 or IPv6 IP address or prefix or a string representation of the address.

Property	Type	Count	Description
datatype	oval:SimpleDatatypeEnumeration	1	Possible values: o 'ipv4_address' o 'ipv6_address' o 'string' Also allows an empty string value.

Figure 2: EntityItemIPAddressStringType Construct

2.17. EntityItemAnySimpleType

The EntityItemAnySimpleType extends the EntityItemSimpleBaseType and describes any simple data.

Property	Type	Count	Description
datatype	oval:SimpleDatatypeEnumeration	1	Any simple datatype. Also allows an empty string value.

Table 14: EntityItemAnySimpleType Construct

2.18. EntityItemBinaryType

The EntityItemBinaryType extends the EntityItemSimpleBaseType and describes any simple binary data.

Property	Type	Count	Description
datatype	oval:SimpleDatatypeEnumeration	1	This value is fixed as 'binary'. Also allows an empty string value.

Table 15: EntityItemBinaryType Construct

2.19. EntityItemBoolType

The EntityItemBoolType extends the EntityItemSimpleBaseType and describes any simple boolean data.

Property	Type	Count	Description
datatype	oval:SimpleDatatypeEnumeration	1	This value is fixed as 'boolean'. Also allows an empty string value.

Table 16: EntityItemBoolType Construct

2.20. EntityItemFloatType

The EntityItemFloatType extends the EntityItemSimpleBaseType and describes any simple float data.

Property	Type	Count	Description
datatype	oval:SimpleDatatypeEnumeration	1	This value is fixed as 'float'. Also allows an empty string value.

Table 17: EntityItemFloatType Construct

2.21. EntityItemIntType

The EntityItemIntType extends the EntityItemSimpleBaseType and describes any simple integer data.

Property	Type	Count	Description
datatype	oval:SimpleDatatypeEnumeration	1	This value is fixed as 'int'. Also allows an empty string value.

Table 18: EntityItemIntType Construct

2.22. EntityItemStringType

The EntityItemStringType extends the EntityItemSimpleBaseType and describes any simple string data.

Property	Type	Count	Description
datatype	oval:SimpleDatatypeEnumeration	0..1	This value is fixed as 'string'.

Table 19: EntityItemStringType Construct

2.23. EntityItemRecordType

The EntityItemRecordType extends the EntityItemComplexBaseType and allows assertions to be made on entities with uniquely named fields. It is intended to be used to assess the results of things such as SQL statements and similar data.

Property	Type	Count	Description
datatype	oval:ComplexDatatypeEnumeration	0..1	This value is fixed as 'record'.
field	EntityItemFieldType	0..*	Defines the name of the field whose value will be assessed.

Table 20: EntityItemRecordType Construct

2.24. EntityItemFieldType

The EntityItemFieldType defines an entity type that captures the details of a single field for a record.

Property	Type	Count	Description
attributes	EntityAttributeGroup	1	The standard attributes available to all entities.
name	string	1	The name of the field. Names MUST be all lower case characters in the range of a-z.
value	string	0..1	The value of the field. An empty string value SHOULD be used when a status other than 'exists' is specified.

Table 21: EntityItemFieldType Construct

2.25. EntityItemVersionType

The EntityItemVersionType extends the EntityItemSimpleBaseType and describes a version string data.

Property	Type	Count	Description
datatype	oval:SimpleDatatypeEnumeration	1	This value is fixed as 'version'. Also allows an empty string value.

Table 22: EntityItemVersionType Construct

2.26. EntityItemFileSetRevisionType

The EntityItemFileSetRevisionType extends the EntityItemSimpleBaseType and describes a file set revision string data.

Property	Type	Count	Description
datatype	oval:SimpleDatatypeEnumeration	1	This value is fixed as 'fileset_revision'. Also allows an empty string value.

Table 23: EntityItemFileSetRevisionType Construct

2.27. EntityItemIOSVersionType

The EntityItemIOSVersionType extends the EntityItemSimpleBaseType and describes a Cisco IOS version string data.

Property	Type	Count	Description
datatype	oval:SimpleDatatypeEnumeration	1	Possible values: o 'ios_version' o 'string' The string type is an option in order to allow use of regular expressions.

Figure 3: EntityItemIOSVersionType Construct

2.28. EntityItemEVRStringType

The EntityItemEVRStringType extends the EntityItemSimpleBaseType and describes an EPOCH:VERSION-RELEASE string data.

Property	Type	Count	Description
datatype	oval:SimpleDatatypeEnumeration	1	This value is fixed as 'evr_string'. Also allows an empty string value.

Table 24: EntityItemEVRStringType Construct

2.29. EntityItemDebianEVRStringType

The EntityItemDebianEVRStringType extends the EntityItemSimpleBaseType and describes an EPOCH:UPSTREAM_VERSION-DEBIAN_REVISION string data for a Debian package.

Property	Type	Count	Description
datatype	oval:SimpleDatatypeEnumeration	1	This value is fixed as 'debian_evr_string'. Also allows an empty string value.

Table 25: EntityItemDebianEVRStringType Construct

3. OVAL System Characteristics Model Schema

The XML Schema that implements this OVAL System Characteristics Model can be found below.

```
        <?xml version="1.0" encoding="utf-8"?>
<xsd:schema
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:oval="http://oval.mitre.org/XMLSchema/oval-common-5"
    xmlns:oval-sc=
    "http://oval.mitre.org/XMLSchema/
    oval-system-characteristics-5"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    xmlns:sch="http://purl.oclc.org/dsdl/schematron"
    xmlns:tns=
    "http://scap.nist.gov/schema/asset-identification/1.1"
    targetNamespace=
    "http://oval.mitre.org/XMLSchema/
    oval-system-characteristics-5"
    elementFormDefault="qualified" version="5.11">
    <xsd:import
        namespace="http://oval.mitre.org/XMLSchema/
        oval-common-5"
        schemaLocation="oval-common-schema.xsd"/>
    <xsd:import
        namespace="http://www.w3.org/2000/09/xmldsig#"
        schemaLocation="xmldsig-core-schema.xsd"/>
    <xsd:annotation>
        <xsd:documentation>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.</xsd:documentation>
        <xsd:appinfo>
            <schema>Core System
                Characteristics</schema>
            <version>5.11.1</version>
            <date>4/22/2015 09:00:00 AM</date>
            <terms_of_use>Copyright (C) 2010 United States Government.
              All Rights Reserved.</terms_of_use>
            <sch:ns prefix="oval-sc" uri=
  "http://oval.mitre.org/XMLSchema/
  oval-system-characteristics-5"/>
            <sch:ns prefix="xsi"
                uri="http://www.w3.org/2001/XMLSchema-instance"
            />
        </xsd:appinfo>
    </xsd:annotation>
    <!-- ================================================= -->
    <!-- ================================================= -->
    <!-- ================================================= -->
    <xsd:element
        name="oval_system_characteristics">
        <xsd:annotation>
            <xsd:documentation>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.</xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="generator"
                    type="oval:GeneratorType">
                    <xsd:annotation>
                        <xsd:documentation>The
                         generator section must be
                         present and provides
                         information about when
                         the system
                         characteristics file was
                         compiled and under what
                         version.</xsd:documentation>
                    </xsd:annotation>
                </xsd:element>
                <xsd:element name="system_info"
                    type="oval-sc:SystemInfoType">
                    <xsd:annotation>
                        <xsd:documentation>The
                         required system_info
```

```
                        element is used to record
                        information about the
                        system being
                        described.</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element
                name="collected_objects"
                type="oval-sc:CollectedObjectsType"
                minOccurs="0" maxOccurs="1">
                <xsd:annotation>
                    <xsd:documentation>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 collected_objects
                     section provides a
                     listing of all the
                     objects used to generate
                     this system
```



```
                     characteristics
                     file.</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="system_data"
                type="oval-sc:SystemDataType"
                minOccurs="0" maxOccurs="1">
                <xsd:annotation>
                    <xsd:documentation>The
                     optional system_data
                     section defines the
                     specific characteristics
                     that have been collected
                     from the
                     system.</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element ref="ds:Signature"
                minOccurs="0" maxOccurs="1">
                <xsd:annotation>
                    <xsd:documentation>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/.
                    </xsd:documentation>
                </xsd:annotation>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:key name="objectKey">
        <xsd:annotation>
            <xsd:documentation>Enforce
                uniqueness amongst the
                individual object ids used in
                the collected object
                section.</xsd:documentation>
```

```
            </xsd:annotation>
            <xsd:selector
                xpath="oval-sc:collected_objects/
                oval-sc:object"/>
            <xsd:field xpath="@id"/>
            <xsd:field xpath="@version"/>
            <xsd:field xpath="@variable_instance"
            />
        </xsd:key>
        <xsd:key name="itemKey">
            <xsd:annotation>
                <xsd:documentation>Enforce
                    uniqueness amongst the
                    individual item
                    ids.</xsd:documentation>
            </xsd:annotation>
            <xsd:selector
                xpath="oval-sc:system_data/*"/>
            <xsd:field xpath="@id"/>
        </xsd:key>
        <xsd:keyref name="itemKeyRef"
            refer="oval-sc:itemKey">
            <xsd:annotation>
                <xsd:documentation>Require that
                    each item reference refers to
                    a valid item
                    id.</xsd:documentation>
            </xsd:annotation>
            <xsd:selector
                xpath=
    "oval-sc:collected_objects/oval-sc:object/oval-sc:reference"/>
            <xsd:field xpath="@item_ref"/>
        </xsd:keyref>
     </xsd:element>
     <!-- ================================================= -->
     <!-- =================  GENERATOR  =================== -->
     <!-- ================================================= -->
     <!--
                The GeneratorType is defined by the oval-common-schema.
                Please refer to that documentation for a description of
                the complex type.
          -->
     <!-- ================================================= -->
     <!-- ================  SYSTEM INFO  ================== -->
     <!-- ================================================= -->
     <xsd:complexType name="SystemInfoType">
         <xsd:annotation>
             <xsd:documentation>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. 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.</xsd:documentation>
            <xsd:documentation>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.</xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element name="os_name"
```

```
                type="xsd:string">
                <xsd:annotation>
                    <xsd:documentation>The
                        required os_name elements
                        describes the operating
                        system of the machine the
                        data was collected
                        on.</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="os_version"
                type="xsd:string">
                <xsd:annotation>
                    <xsd:documentation>The
                        required os_version
                        elements describe the
                        operating system version
                        of the machine the data
                        was collected
                        on.</xsd:documentation>
```



```
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="architecture"
                type="xsd:string">
                <xsd:annotation>
                    <xsd:documentation>The
                        required architecture
                        element describes the
                        hardware architecture type
                        of the system data was
                        collected
                        on.</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="primary_host_name"
                type="xsd:string">
                <xsd:annotation>
                    <xsd:documentation>The
                        required primary_host_name
                        element is the primary
                        host name of the machine
                        the data was collected
                        on.</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="interfaces"
                type="oval-sc:InterfacesType">
                <xsd:annotation>
                    <xsd:documentation>The
                        required interfaces
                        element outlines the
                        network interfaces that
                        exist on the
                        system.</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:any minOccurs="0"
                maxOccurs="unbounded"
                processContents="lax">
                <xsd:annotation>
                    <xsd:documentation>The Asset
                        Identification
                        specification
                        (http://scap.nist.gov/specifications/ai/)
                        provides a standardized
                        way of reporting asset
                        information across
                        different
```



```
                        organizations.</xsd:documentation>
                    <xsd:documentation>The
```

```
                        information contained
                        within an AI
                        computing-device element
                        is similar to the
                        information collected by
                        OVAL's
                        SystemInfoType.</xsd:documentation>
                    <xsd:documentation>To support
                        greater interoperability,
                        an ai:computing-device
                        element describing the
                        system that data was
                        collected from may appear
                        at this point in an OVAL
                        System Characteristics
                        document.</xsd:documentation>
                </xsd:annotation>
            </xsd:any>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="InterfacesType">
        <xsd:annotation>
            <xsd:documentation>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.</xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element name="interface"
                type="oval-sc:InterfaceType"
                minOccurs="0"
                maxOccurs="unbounded">
                <xsd:annotation>
                    <xsd:documentation>Please
                        refer to the description
                        of the InterfaceType for
                        more
                        information.</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="InterfaceType">
```



```
        <xsd:annotation>
            <xsd:documentation>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.</xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element name="interface_name"
                type="xsd:string">
                <xsd:annotation>
                    <xsd:documentation>The
                        required interface_name
                        element is the name of the
                        interface</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="ip_address"
                type="xsd:string">
                <xsd:annotation>
                    <xsd:documentation>The
                        required ip_address
                        element holds the IP
                        address for the interface.
                        Note that the IP address
                        can be IPv4 or
                        IPv6.</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="mac_address"
```

```
                type="xsd:string">
                <xsd:annotation>
                    <xsd:documentation>The
                        required mac_address
                        element holds the MAC
                        address for the interface.
                        MAC addresses should be
                        formatted according to the
                        IEEE 802-2001 standard
                        which states that a MAC
                        address is a sequence of
                        six octet values,
                        separated by hyphens,
                        where each octet is
                        represented by two
                        hexadecimal digits.
```



```
                        Uppercase letters should
                        also be used to represent
                        the hexadecimal digits A
                        through
                        F.</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
    <!-- ================================================ -->
    <!-- ============  COLLECTED OBJECTS  ================ -->
    <!-- ================================================ -->
    <xsd:complexType name="CollectedObjectsType">
        <xsd:annotation>
            <xsd:documentation>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.</xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element name="object"
                type="oval-sc:ObjectType"
                minOccurs="1"
                maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="ObjectType">
        <xsd:annotation>
            <xsd:documentation>The ObjectType
                complex type provides a reference
                between items collected and a
                related global OVAL
                Object.</xsd:documentation>
            <xsd:documentation>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'.
                For details on how to handle
                items, when an OVAL Object does
                not exist on the system, please
                see the ItemType documentation.
                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.</xsd:documentation>
            <xsd:documentation>The required id
                attribute is the id of the global
                OVAL Object.</xsd:documentation>
            <xsd:documentation>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.</xsd:documentation>
            <xsd:documentation>The optional
                variable_instance identifier is a
                unique id that differentiates each
                unique instance of an object.
                Capabilities that use OVAL may
                reference the same definition
                multiple times and provide
                different variable values each
                time the definition is referenced.
                This will result in multiple
                instances of an object being
                included in the OVAL System
                Characteristics file (definitions
                that do not use variables can only
                have one unique instance). The
                inclusion of this unique instance
                identifier allows the OVAL Results
                document to associate the correct
                objects and items for each
                combination of supplied
                values.</xsd:documentation>
            <xsd:documentation>The optional
                comment attribute provides a short
                description of the
                object.</xsd:documentation>
            <xsd:documentation>The required flag
                attribute holds information
                regarding the outcome 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.</xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element name="message"
                type="oval:MessageType"
                minOccurs="0"
                maxOccurs="unbounded">
                <xsd:annotation>
                    <xsd:documentation>The
                        optional message element
                        holds an error message or
                        some other string that the
                        data collection engine
                        wishes to pass
                        along.</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="variable_value"
                type="oval-sc:VariableValueType"
                minOccurs="0"
                maxOccurs="unbounded">
                <xsd:annotation>
                    <xsd:documentation>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
                        unique set of matching
                        items depending on the
                        value 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 a
                        collection of values, then
                        multiple variable_value
                        elements would exist with
                        the same variable_id
                        attribute.
                    </xsd:documentation>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="reference"
                type="oval-sc:ReferenceType"
                minOccurs="0"
                maxOccurs="unbounded">
                <xsd:annotation>
                    <xsd:documentation>The
                        optional reference element
                        links 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.</xsd:documentation>
                </xsd:annotation>
            </xsd:element>
        </xsd:sequence>
        <xsd:attribute name="id"
            type="oval:ObjectIDPattern"
            use="required"/>
        <xsd:attribute name="version"
            type="xsd:nonNegativeInteger"
            use="required"/>
        <xsd:attribute name="variable_instance"
            type="xsd:nonNegativeInteger"
            use="optional" default="1"/>
        <xsd:attribute name="comment"
```



```
            type="xsd:string" use="optional"/>
        <xsd:attribute name="flag"
            type="oval-sc:FlagEnumeration"
            use="required"/>
    </xsd:complexType>
    <xsd:complexType name="VariableValueType">
        <xsd:annotation>
            <xsd:documentation>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.</xsd:documentation>
            </xsd:annotation>
            <xsd:simpleContent>
                <xsd:extension
                    base="xsd:anySimpleType">
                    <xsd:attribute name="variable_id"
                        type="oval:VariableIDPattern"
                        use="required"/>
                </xsd:extension>
            </xsd:simpleContent>
        </xsd:complexType>
        <xsd:complexType name="ReferenceType">
            <xsd:annotation>
                <xsd:documentation>The ReferenceType
                    complex type specifies an item in
                    the system characteristics file.
                    This reference is used to link
                    global OVAL Objects to specific
                    items.</xsd:documentation>
            </xsd:annotation>
            <xsd:attribute name="item_ref"
                type="oval:ItemIDPattern"
                use="required"/>
        </xsd:complexType>
        <!-- ================================================= -->
        <!-- ===============  SYSTEM DATA  ==================== -->
        <!-- ================================================= -->
        <xsd:complexType name="SystemDataType">
            <xsd:annotation>
                <xsd:documentation>The SystemDataType
                    complex type is a container for
                    one or more item elements. Each
                    item defines a specific piece of
                    data on the
```



```
                    system.</xsd:documentation>
            </xsd:annotation>
            <xsd:sequence>
                <xsd:element ref="oval-sc:item"
                    minOccurs="1"
                    maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
        <xsd:element name="item"
            type="oval-sc:ItemType" abstract="true">
            <xsd:annotation>
                <xsd:documentation>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
                    may 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.</xsd:documentation>
            </xsd:annotation>
        </xsd:element>
        <xsd:complexType name="ItemType">
            <xsd:annotation>
                <xsd:documentation>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.</xsd:documentation>
```

```
            <xsd:documentation>The required id
                attribute is a unique (to the
                file) identifier that allows the
                specific item to be
                referenced.</xsd:documentation>
            <xsd:documentation>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 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. When creating
                items, any entities that can
                successfully be collected should
                be reported.</xsd:documentation>
            <xsd:documentation>In some cases, when
                an item for a specified object
                does not exist, it may be
                beneficial to report a partial
                match of an item showing what
                entities did exist and what
                entities did not exist for
                debugging purposes. This is
                especially true when considering
                items that are collected by
                objects with hierarchical object
                entities. An example of such a
                case is when a file_object has a
                path entity equal to 'C:\' and a
                filename entity equal to
                'test.txt' where 'test.txt' does
                not exist in the 'C:\' directory.
                This would result in the creation
                of a partially matching file_item
                with a status of 'does not exist'
                where the path entity equals 'C:\'
                and the filename entity equals
                'test.txt' with a status of 'does
                not exist'. By showing the partial
                match, someone reading a
                system-characteristics document
                can quickly see that a matching
                file_item did not exist because
                the specified filename did not
                exist and not that the specified
                path did not exist. Again, please
                note that the implementation of
                partial matches, when an item for
                a specified object does not exist,
```



```
                is completely optional.
            </xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:element name="message"
                type="oval:MessageType"
                minOccurs="0" maxOccurs="50"/>
        </xsd:sequence>
        <xsd:attribute name="id"
            type="oval:ItemIDPattern"
            use="required"/>
        <xsd:attribute name="status"
            type="oval-sc:StatusEnumeration"
            use="optional" default="exists"/>
```

```
    </xsd:complexType>
    <!-- ================================================= -->
    <!-- ====================  SIGNATURE  ================== -->
    <!-- ================================================= -->
    <!--
The signature element is defined by the xmldsig schema.
Please refer to that documentation for a description of
the valid elements and types.  More information about the
official W3C Recommendation regarding XML digital
signatures can be found at
http://www.w3.org/TR/xmldsig-core/.
        -->
    <!-- ================================================= -->
    <!-- ================  ENUMERATIONS  ================= -->
    <!-- ================================================= -->
    <xsd:simpleType name="FlagEnumeration">
        <xsd:annotation>
            <xsd:documentation>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 needs to be available. The
                flags help detail the outcome of
                attempting to collect information
                related to these
                objects..</xsd:documentation>
            <xsd:appinfo>
                <evaluation_documentation>Below is
                    a table that outlines how each
```



```
                    FlagEnumeration value effects
                    evaluation of a given test.
                    Note that this is related to
                    the existence of a unique set
                    of items identified by an
                    object and not each item's
                    compliance with a state. The
                    left column identifies the
                    FlagEnumeration value in
                    question. The right column
                    specifies the
                    ResultEnumeration value that
                    should be used when evaluating
                    the collected
                    object.</evaluation_documentation>
                <evaluation_chart xml:space="preserve">
                  ||
  flag value      ||      test result is
                  ||
------------------||----------------------------
  error           ||  error
  complete        ||    (test result depends on
  incomplete      ||     check_existence and
  does not exist  ||     check attributes)
  not collected   ||  unknown
  not applicable  ||  not applicable
------------------||----------------------------
                 </evaluation_chart>
            </xsd:appinfo>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="error">
                <xsd:annotation>
                    <xsd:documentation>A flag of
                        'error' indicates that
                        there was an error trying
                        to identify items 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
                        entity, but rather when it
```

```
                        could not be determined if
                        an item exists or not. Any
                        error in retrieving a
                        specific entity should be
```



```
                        represented by setting the
                        status of that specific
                        entity to
                        'error'.</xsd:documentation>
                </xsd:annotation>
            </xsd:enumeration>
            <xsd:enumeration value="complete">
                <xsd:annotation>
                    <xsd:documentation>A flag of
                        'complete' indicates that
                        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.</xsd:documentation>
                </xsd:annotation>
            </xsd:enumeration>
            <xsd:enumeration value="incomplete">
                <xsd:annotation>
                    <xsd:documentation>A flag of
                        'incomplete' indicates
                        that a matching item
                        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.</xsd:documentation>
                </xsd:annotation>
            </xsd:enumeration>
            <xsd:enumeration
                value="does not exist">
                <xsd:annotation>
                    <xsd:documentation>A flag of
                        'does not exist' indicates
```



```
                        that the underlying
                        structure is installed on
                        the system but no matching
                        item was found. For
                        example, the Windows
                        metabase is installed but
                        there were no items that
                        matched the
                        metabase_object. In this
                        example, if the metabase
                        itself was not installed,
                        then the flag would have
                        been 'not
                        applicable'.</xsd:documentation>
                </xsd:annotation>
            </xsd:enumeration>
            <xsd:enumeration value="not collected">
                <xsd:annotation>
```

```
                    <xsd:documentation>A flag of
                        'not collected' indicates
                        that no attempt was made
                        to collect items on the
                        system. An object with
                        this flag will produce an
                        'unknown' result during
                        analysis since it is
                        unknown if matching items
                        exists on the system or
                        not. This is different
                        from an 'error' flag
                        because an 'error' flag
                        indicates that an attempt
                        was made to collect items
                        on system whereas a 'not
                        collected' flag indicates
                        that an attempt was not
                        made to collect items on
                        the
                        system.</xsd:documentation>
                </xsd:annotation>
            </xsd:enumeration>
            <xsd:enumeration
                value="not applicable">
                <xsd:annotation>
                    <xsd:documentation>A flag of
                        'not applicable' indicates
                        that the specified object
                        is not applicable to the
```



```
                        system being
                        characterized. This could
                        be because the data
                        repository is not
                        installed or that the
                        object structure is for a
                        different flavor of
                        systems. An example would
                        be trying to collect
                        objects related to a Red
                        Hat system off of a
                        Windows system. Another
                        example would be trying to
                        collect an rpminfo_object
                        on a Linux system if the
                        rpm packaging system is
                        not installed. If the rpm
                        packaging system is
                        installed and the
                        specified rpminfo_object
                        could not be found, then
                        the flag would be 'does
                        not
                        exist'.</xsd:documentation>
                </xsd:annotation>
            </xsd:enumeration>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:simpleType name="StatusEnumeration">
        <xsd:annotation>
            <xsd:documentation>The
                StatusEnumeration simple type
                defines the valid status messages
                associated with collection of
                specific information associated
                with an item.</xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="error">
                <xsd:annotation>
                    <xsd:documentation>A status of
                        'error' says that there
                        was an error collecting
                        information associated
                        with an item as a whole or
                        a specific entity. An item
                        would have a status of
                        'error' if a problem
```

```
                        occurred that prevented
                        the item from being
                        collected. For example, a
                        file_item would have a
                        status of 'error' if a
                        handle to the file could
                        not be opened because the
                        handle was already in use
                        by another program. Also,
                        if an item has entities
                        with a status of 'error'
                        and entities with a status
                        of 'exists', the status of
                        'error' must not be
                        propagated up to the item
                        level as the item may
                        still be
                        usable.</xsd:documentation>
                </xsd:annotation>
            </xsd:enumeration>
            <xsd:enumeration value="exists">
                <xsd:annotation>
                    <xsd:documentation>A status of
                        'exists' says that the
                        item or specific piece of
                        information exists on the
                        system and has been
                        collected.</xsd:documentation>
                </xsd:annotation>
            </xsd:enumeration>
            <xsd:enumeration
                value="does not exist">
                <xsd:annotation>
                    <xsd:documentation>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
                        does not exist. This can
                        happen when a certain
                        entity is only pertinent
                        to particular instances or
                        if the information for
```



```
                        that entity is not
                        set.</xsd:documentation>
                </xsd:annotation>
            </xsd:enumeration>
            <xsd:enumeration value="not collected">
                <xsd:annotation>
                    <xsd:documentation>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.</xsd:documentation>
                </xsd:annotation>
            </xsd:enumeration>
        </xsd:restriction>
    </xsd:simpleType>
    <!-- ================================================= -->
    <!-- ===============  ENTITY TYPES  ================== -->
    <!-- ================================================= -->
```

```
    <xsd:attributeGroup
        name="EntityAttributeGroup">
        <xsd:annotation>
            <xsd:documentation>The
                EntityAttributeGroup is a
                collection of attributes that are
                common to all entities. This group
                defines these attributes and their
                default values. Individual
                entities may limit allowed values
                for these attributes, but all
                entities will support these
                attributes.</xsd:documentation>
            <xsd:appinfo>
                <sch:pattern
                    id="oval-sc_entity_rules">
                    <sch:rule
                        context=
             "oval-sc:system_data/*/*|
              oval-sc:system_data/*/*/*">
                            <sch:assert flag="WARNING"
                  test="not(@status) or @status='exists'
                  or .=''">Warning: item
                  <sch:value-of
                  select="../@id"/> - a
```



```
                  value for the
                  <sch:value-of
                  select="name()"/> entity
                  should only be supplied
                  if the status attribute
                  is 'exists'</sch:assert>
 <!--<sch:assert test="if (@datatype='binary') then
 (matches(., '^ [0-9a-fA-F]*$')) else (1=1)">
 <sch:value-of select="../@id"/> - A
 value of '<sch:value-of select="."/>' for the
 <sch:value-of select="name()"/>
 entity is not valid given a datatype of binary.
 </sch:assert>-->

 <!--<sch:assert test="if (@datatype='boolean')
 then (matches(.,'^true$|^false$|^1$|^0$')) else
 (1=1)"><sch:value-of select="../@id"/>
 - A value of '<sch:value-of select="."/>' for the
 <sch:value-of select="name()"/> entity is not valid given a
 datatype of boolean.</sch:assert>-->

 <!--<sch:assert test="if (@datatype='evr_string') then
 (matches(.,'^[^:\-]*:[^:\-]*-[^:\-]*$')) else (1=1)">
 <sch:value-of select="../@id"/> - A value of
 '<sch:value-of select="."/>' for the
 <sch:value-of select="name()"/> entity is not valid given a
 datatype of evr_string.</sch:assert>-->

 <!--<sch:assert test="if (@datatype='float') then
 (matches(.,'^[+\-]?[0-9]+([\.][0-9]+)?
 ([eE][+\-]?[0-9]+)?$|^NaN$|^INF$|^\-INF$')) else
 (1=1)"><sch:value-of select="../@id"/>
 - A value of '<sch:value-of select="."/>' for the
 <sch:value-of select="name()"/>
 entity is not valid given a datatype of float.
 </sch:assert>-->

 <!--<sch:assert test="if (@datatype='int') then
 (matches(.,'^[+\-]?[0-9]+$')) else (1=1)">
 <sch:value-of select="../@id"/> - A value of
 '<sch:value-of select="."/>' for the <sch:value-of
 select="name()"/> entity is not valid given a datatype of int.
 </sch:assert>-->
                  </sch:rule>
                  <sch:rule
                      context="oval-sc:system_data/*/*[not(
 (@xsi:nil='1' or @xsi:nil='true')) and @datatype='int']|
 oval-sc:system_data/*/*/*[not((@xsi:nil='1' or @xsi:nil='true'))
```

```
 and @datatype='int']">
                        <sch:assert
                         test="(not(contains(.,'.'))) and
                         (number(.) = floor(.))"
                         ><sch:value-of
                         select="../@id"/> - The
                         datatype for the
                         <sch:value-of
                         select="name()"/> entity
                         is 'int' but the value is
                         not an
                         integer.</sch:assert>
 <!--  Must test for decimal point because
       number(x.0) = floor(x.0) is true -->
                    </sch:rule>
                </sch:pattern>
            </xsd:appinfo>
          </xsd:annotation>
          <xsd:attribute name="datatype"
              type="oval:DatatypeEnumeration"
              use="optional" default="string">
              <xsd:annotation>
                  <xsd:documentation>The optional
                      datatype attribute determines
                      the type of data expected (the
                      default datatype is 'string').
                      Note that the datatype
                      attribute simply defines the
                      type of data as found on the
                      system, it is not used during
                      evaluation. An OVAL Definition
                      defines how the data should be
                      interpreted during analysis.
                      If the definition states a
                      datatype that is different
                      than what the system
                      characteristics presents, then
                      a type cast must be
                      made.</xsd:documentation>
              </xsd:annotation>
          </xsd:attribute>
          <xsd:attribute name="mask"
              type="xsd:boolean" use="optional"
              default="false">
              <xsd:annotation>
                  <xsd:documentation>The optional
                      mask attribute is used to
                      identify values that have been
```



```
                      hidden for sensitivity
                      concerns. This is used by the
                      Result document which uses the
                      System Characteristics schema
                      to format the information
                      found on a specific system.
                      When the mask attribute is set
                      to 'true' on an OVAL Entity or
                      an OVAL Field, the
                      corresponding collected value
                      of that OVAL Entity or OVAL
                      Field MUST NOT be present in
                      the "results" section of the
                      OVAL Results document; the
                      "oval_definitions" section
                      must not be altered and must
                      be an exact copy of the
                      definitions evaluated. Values
                      MUST NOT be masked in OVAL
                      System Characteristics
                      documents that are not
                      contained within an OVAL
                      Results document. It is
                      possible for masking conflicts
                      to occur where one entity has
                      mask set to true and another
```

```
                    entity has mask set to false.
                    A conflict will occur when the
                    mask attribute is set
                    differently on an OVAL Object
                    and matching OVAL State or
                    when more than one OVAL
                    Objects identify the same OVAL
                    Item(s). When such a conflict
                    occurs the result is always to
                    mask the
                    entity.</xsd:documentation>
                </xsd:annotation>
            </xsd:attribute>
            <xsd:attribute name="status"
                type="oval-sc:StatusEnumeration"
                use="optional" default="exists">
                <xsd:annotation>
                    <xsd:documentation>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'.</xsd:documentation>
                </xsd:annotation>
            </xsd:attribute>
        </xsd:attributeGroup>

        <xsd:complexType
            name="EntityItemSimpleBaseType"
            abstract="true">
            <xsd:annotation>
                <xsd:documentation>The
                    EntityItemSimpleBaseType complex
                    type is an abstract type that
                    serves as the base type for all
                    simple item
                    entities.</xsd:documentation>
            </xsd:annotation>
            <xsd:simpleContent>
                <xsd:extension
                    base="xsd:anySimpleType">
                    <xsd:attributeGroup
                        ref="oval-sc:EntityAttributeGroup"
                    />
                </xsd:extension>
            </xsd:simpleContent>
        </xsd:complexType>

        <xsd:complexType
            name="EntityItemComplexBaseType"
            abstract="true">
            <xsd:annotation>
                <xsd:documentation>The
                    EntityItemComplexBaseType complex
                    type is an abstract type that
                    serves as the base type for all
                    complex item
                    entities.</xsd:documentation>
            </xsd:annotation>
            <xsd:attributeGroup
                ref="oval-sc:EntityAttributeGroup"/>
        </xsd:complexType>

        <xsd:complexType
            name="EntityItemIPAddressType">
            <xsd:annotation>
                <xsd:documentation>The
```

```
                EntityItemIPAddressType type is
                extended by the entities of an
                individual item. This type
                provides uniformity to each entity
                by including the attributes found
                in the EntityItemSimpleBaseType.
                This specific type describes any
                IPv4/IPv6 address or address
                prefix.</xsd:documentation>
            </xsd:annotation>
            <xsd:simpleContent>
                <xsd:restriction
                    base="oval-sc:EntityItemSimpleBaseType">
                    <xsd:simpleType>
                        <xsd:restriction
                            base="xsd:string"/>
                    </xsd:simpleType>
                    <xsd:attribute name="datatype"
                        use="required">
                        <xsd:simpleType>
                            <xsd:restriction
                             base="oval:SimpleDatatypeEnumeration">
                             <xsd:enumeration
                             value="ipv4_address"/>
                             <xsd:enumeration
                             value="ipv6_address"/>
                            </xsd:restriction>
                        </xsd:simpleType>
                    </xsd:attribute>
                </xsd:restriction>
            </xsd:simpleContent>
        </xsd:complexType>
        <xsd:complexType
            name="EntityItemIPAddressStringType">
            <xsd:annotation>
                <xsd:documentation>The
                EntityItemIPAddressStringType type
                is extended by the entities of an
                individual item. This type
                provides uniformity to each entity
                by including the attributes found
                in the EntityItemSimpleBaseType.
                This specific type describes any
                IPv4/IPv6 address, address prefix,
                or its string
                representation.</xsd:documentation>
            </xsd:annotation>
            <xsd:simpleContent>
```



```
                <xsd:restriction
                    base="oval-sc:EntityItemSimpleBaseType">
                    <xsd:simpleType>
                        <xsd:restriction
                            base="xsd:string"/>
                    </xsd:simpleType>
                    <xsd:attribute name="datatype"
                        use="optional"
                        default="string">
                        <xsd:simpleType>
                            <xsd:restriction
                             base="oval:SimpleDatatypeEnumeration">
                             <xsd:enumeration
                             value="ipv4_address"/>
                             <xsd:enumeration
                             value="ipv6_address"/>
                             <xsd:enumeration
                             value="string"/>
                            </xsd:restriction>
                        </xsd:simpleType>
                    </xsd:attribute>
                </xsd:restriction>
            </xsd:simpleContent>
        </xsd:complexType>
        <xsd:complexType
            name="EntityItemAnySimpleType">
            <xsd:annotation>
                <xsd:documentation>The
                EntityItemAnySimpleType type is
                extended by the entities of an
```

```
                individual item. This type
                provides uniformity to each entity
                by including the attributes found
                in the EntityItemSimpleBaseType.
                This specific type describes any
                simple data.</xsd:documentation>
            </xsd:annotation>
            <xsd:simpleContent>
                <xsd:restriction
                    base="oval-sc:EntityItemSimpleBaseType">
                    <xsd:simpleType>
                        <xsd:restriction
                            base="xsd:string"/>
                    </xsd:simpleType>
                    <xsd:attribute name="datatype"
                        type="oval:SimpleDatatypeEnumeration"
                        use="optional"
                        default="string"/>
```



```
                </xsd:restriction>
            </xsd:simpleContent>
        </xsd:complexType>
        <xsd:complexType name="EntityItemBinaryType">
            <xsd:annotation>
                <xsd:documentation>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 EntityItemSimpleBaseType.
                    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.</xsd:documentation>
            </xsd:annotation>
            <xsd:simpleContent>
                <xsd:restriction
                    base="oval-sc:EntityItemSimpleBaseType">
                    <xsd:simpleType>
                      <xsd:union
                      memberTypes="xsd:hexBinary
                      oval:EmptyStringType"
                      />
                    </xsd:simpleType>
                    <xsd:attribute name="datatype"
                        type="oval:SimpleDatatypeEnumeration"
                        use="required" fixed="binary"
                    />
                </xsd:restriction>
            </xsd:simpleContent>
        </xsd:complexType>
        <xsd:complexType name="EntityItemBoolType">
            <xsd:annotation>
                <xsd:documentation>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 EntityItemSimpleBaseType.
                    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.</xsd:documentation>
```

```
        </xsd:annotation>
        <xsd:simpleContent>
            <xsd:restriction
                base="oval-sc:EntityItemSimpleBaseType">
                <xsd:simpleType>
                    <xsd:union
                    memberTypes="xsd:boolean
                    oval:EmptyStringType"
                    />
                </xsd:simpleType>
                <xsd:attribute name="datatype"
                    type="oval:SimpleDatatypeEnumeration"
                    use="required" fixed="boolean"
                />
            </xsd:restriction>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType name="EntityItemFloatType">
        <xsd:annotation>
            <xsd:documentation>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 EntityItemSimpleBaseType.
                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.</xsd:documentation>
        </xsd:annotation>
        <xsd:simpleContent>
            <xsd:restriction
                base="oval-sc:EntityItemSimpleBaseType">
                <xsd:simpleType>
                    <xsd:union
                    memberTypes="xsd:float oval:EmptyStringType"
                    />
                </xsd:simpleType>
                <xsd:attribute name="datatype"
                    type="oval:SimpleDatatypeEnumeration"
```



```
                    use="required" fixed="float"/>
            </xsd:restriction>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType name="EntityItemIntType">
        <xsd:annotation>
            <xsd:documentation>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 EntityItemSimpleBaseType. 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.</xsd:documentation>
        </xsd:annotation>
        <xsd:simpleContent>
            <xsd:restriction
                base="oval-sc:EntityItemSimpleBaseType">
                <xsd:simpleType>
                    <xsd:union
                    memberTypes="xsd:integer
                    oval:EmptyStringType"
                    />
                </xsd:simpleType>
                <xsd:attribute name="datatype"
                    type="oval:SimpleDatatypeEnumeration"
                    use="required" fixed="int"/>
            </xsd:restriction>
```

```
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType name="EntityItemStringType">
        <xsd:annotation>
            <xsd:documentation>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 EntityItemSimpleBaseType.
                This specific type describes
                simple string
                data.</xsd:documentation>
```



```
        </xsd:annotation>
        <xsd:simpleContent>
            <xsd:restriction
                base="oval-sc:EntityItemSimpleBaseType">
                <xsd:simpleType>
                    <xsd:restriction
                        base="xsd:string"/>
                </xsd:simpleType>
                <xsd:attribute name="datatype"
                    type="oval:SimpleDatatypeEnumeration"
                    use="optional" fixed="string"
                />
            </xsd:restriction>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType name="EntityItemRecordType">
        <xsd:annotation>
            <xsd:documentation>The
                EntityItemRecordType defines an
                entity that consists of a number
                of named fields. This structure is
                used for representing a record
                from a database query and other
                similar structures where multiple
                related fields must be collected
                at once. Note that for all
                entities of this type, the only
                allowed datatype is
                'record'.</xsd:documentation>
            <xsd:documentation>Note the datatype
                attribute must be set to
                'record'.</xsd:documentation>
 <!--
   NOTE: The restriction that the only allowed
   datatype is 'record' is enforced by scheamtron rules
   placed on each entity that uses this type.
   This is due to the fact that this type is developed
   as an xsd:extension of the oval-sc:EntityItemFieldType.
   This base type declares a datatype attribute. To restrict
   the datatype attribute to only allow 'record' would
   need an xsd:restriction. We cannot do both and
   xsd:extension and an xsd:restriction at the same time.
 -->
            <xsd:documentation>Note that when the
                mask attribute is set to 'true',
                all child field elements must be
                masked regardless of the child
                field's mask attribute
```



```
                value.</xsd:documentation>
        </xsd:annotation>
        <xsd:complexContent>
            <xsd:extension
                base="oval-sc:EntityItemComplexBaseType">
                <xsd:sequence>
                    <xsd:element name="field"
                        type="oval-sc:EntityItemFieldType"
```

```
                        minOccurs="0"
                        maxOccurs="unbounded"/>
                </xsd:sequence>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>
    <xsd:complexType name="EntityItemFieldType">
        <xsd:annotation>
            <xsd:documentation>The
                EntityItemFieldType defines an
                element with simple content that
                represents a named field in a
                record that may contain any number
                of named fields. The
                EntityItemFieldType is much like
                all other entities with one
                significant difference, the
                EntityItemFieldType has a name
                attribute.</xsd:documentation>
            <xsd:documentation>The required name
                attribute specifies a name for the
                field. Field names are lowercase
                and may occur more than once to
                allow for a field to have multiple
                values.</xsd:documentation>
            <xsd:documentation>Note that when the
                mask attribute is set to 'true' on
                a field's parent element the field
                must be masked regardless of the
                field's mask attribute
                value.</xsd:documentation>
        </xsd:annotation>
        <xsd:simpleContent>
            <xsd:extension
                base="xsd:anySimpleType">
                <xsd:attribute name="name"
                    use="required">
                    <xsd:annotation>
                        <xsd:documentation>A
                         string restricted to
```



```
                         disallow upper case
                         characters.</xsd:documentation>
                    </xsd:annotation>
                    <xsd:simpleType>
                        <xsd:restriction
                         base="xsd:string">
                         <xsd:pattern
                         value="[^A-Z]+"/>
                        </xsd:restriction>
                    </xsd:simpleType>
                </xsd:attribute>
                <xsd:attributeGroup
                    ref="oval-sc:EntityAttributeGroup"
                />
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType name="EntityItemVersionType">
        <xsd:annotation>
            <xsd:documentation>The
                EntityItemVersionType type is
                extended by the entities of an
                individual item. This type
                provides uniformity to each entity
                by including the attributes found
                in the EntityItemSimpleBaseType.
                This specific type describes
                version data.</xsd:documentation>
        </xsd:annotation>
        <xsd:simpleContent>
            <xsd:restriction
                base="oval-sc:EntityItemSimpleBaseType">
                <xsd:simpleType>
                    <xsd:restriction
                        base="xsd:string"/>
                </xsd:simpleType>
                <xsd:attribute name="datatype"
                    type="oval:SimpleDatatypeEnumeration"
```

```
                    use="required" fixed="version"
                />
            </xsd:restriction>
        </xsd:simpleContent>
    </xsd:complexType>
    <xsd:complexType
        name="EntityItemFilesetRevisionType">
        <xsd:annotation>
            <xsd:documentation>The
                EntityItemFilesetRevisionType type
```



```
                is extended by the entities of an
                individual item. This type
                provides uniformity to each entity
                by including the attributes found
                in the EntityItemSimpleBaseType.
                This specific type represents the
                version string related to filesets
                in HP-UX.</xsd:documentation>
            </xsd:annotation>
            <xsd:simpleContent>
                <xsd:restriction
                    base="oval-sc:EntityItemSimpleBaseType">
                    <xsd:simpleType>
                        <xsd:restriction
                            base="xsd:string"/>
                    </xsd:simpleType>
                    <xsd:attribute name="datatype"
                        type="oval:SimpleDatatypeEnumeration"
                        use="required"
                        fixed="fileset_revision"/>
                </xsd:restriction>
            </xsd:simpleContent>
        </xsd:complexType>
        <xsd:complexType
            name="EntityItemIOSVersionType">
            <xsd:annotation>
                <xsd:documentation>The
                    EntityItemIOSVersionType type is
                    extended by the entities of an
                    individual item. This type
                    provides uniformity to each entity
                    by including the attributes found
                    in the EntityItemSimpleBaseType.
                    This specific type represents the
                    version string for
                    IOS.</xsd:documentation>
            </xsd:annotation>
            <xsd:simpleContent>
                <xsd:restriction
                    base="oval-sc:EntityItemSimpleBaseType">
                    <xsd:simpleType>
                        <xsd:restriction
                            base="xsd:string"/>
                    </xsd:simpleType>
                    <xsd:attribute name="datatype"
                        type="oval:SimpleDatatypeEnumeration"
                        use="required"
                        fixed="ios_version"/>
```



```
                </xsd:restriction>
            </xsd:simpleContent>
        </xsd:complexType>
        <xsd:complexType
            name="EntityItemEVRStringType">
            <xsd:annotation>
                <xsd:documentation>The
                    EntityItemEVRStringType type is
                    extended by the entities of an
                    individual item. This type
                    provides uniformity to each entity
                    by including the attributes found
```

```
                in the EntityItemSimpleBaseType.
                This type represents the epoch,
                version, and release fields, for
                an RPM package, as a single
                version string. It has the form
                "EPOCH:VERSION-RELEASE". Note that
                a null epoch (or '(none)' as
                returned by rpm) is equivalent to
                '0' and would hence have the form
                0:VERSION-RELEASE. Comparisons
                involving this datatype should
                follow the algorithm of librpm's
                rpmvercmp()
                function.</xsd:documentation>
            </xsd:annotation>
            <xsd:simpleContent>
                <xsd:restriction
                    base="oval-sc:EntityItemSimpleBaseType">
                    <xsd:simpleType>
                        <xsd:restriction
                            base="xsd:string"/>
 <!-- TODO: Should there be a pattern restriction
      here to enforce the pattern mentioned above? -->
                    </xsd:simpleType>
                    <xsd:attribute name="datatype"
                        type="oval:SimpleDatatypeEnumeration"
                        use="required"
                        fixed="evr_string"/>
                </xsd:restriction>
            </xsd:simpleContent>
        </xsd:complexType>
        <xsd:complexType
            name="EntityItemDebianEVRStringType">
            <xsd:annotation>
                <xsd:documentation>The
                  EntityItemDebianEVRStringType type
```



```
                is extended by the entities of an
                individual item. This type
                provides uniformity to each entity
                by including the attributes found
                in the EntityItemSimpleBaseType.
                This type represents the epoch,
                upstream_version, and
                debian_revision fields, for a
                Debian package, as a single
                version string. It has the form
                "EPOCH:UPSTREAM_VERSION-DEBIAN_REVISION".
                Note that a null epoch (or
                '(none)' as returned by dpkg) is
                equivalent to '0' and would hence
                have the form
                0:UPSTREAM_VERSION-DEBIAN_REVISION.
                Comparisons involving this
                datatype should follow the
                algorithm outlined in Chapter 5 of
                the "Debian Policy Manual"
                (https://www.debian.org/doc/debian-policy/
                ch-controlfields.html#s-f-Version).
                An implementation of this is the
                cmpversions() function in dpkg's
                enquiry.c.</xsd:documentation>
            </xsd:annotation>
            <xsd:simpleContent>
                <xsd:restriction
                    base="oval-sc:EntityItemSimpleBaseType">
                    <xsd:simpleType>
                        <xsd:restriction
                            base="xsd:string"/>
 <!-- TODO: Should there be a pattern restriction here
      to enforce the pattern mentioned above? -->
                    </xsd:simpleType>
                    <xsd:attribute name="datatype"
                        type="oval:SimpleDatatypeEnumeration"
                        use="required"
                        fixed="debian_evr_string"/>
                </xsd:restriction>
            </xsd:simpleContent>
        </xsd:complexType>
```

```
</xsd:schema>
```


4. Intellectual Property Considerations

DHS, on behalf of the United States, owns the registered OVAL trademarks, identifying the OVAL STANDARDS SUITE and any component part, as that suite has been provided to the IETF Trust. A "(R)" will be used in conjunction with the first use of any OVAL trademark in any document or publication in recognition of DHS's trademark ownership.

5. Acknowledgements

The authors wish to thank DHS for sponsoring the OVAL effort over the years which has made this work possible. The authors also wish to thank the original authors of this document Jonathan Baker, Matthew Hansbury, and Daniel Haynes of the MITRE Corporation as well as the OVAL Community for its assistance in contributing and reviewing the original document. The authors would also like to acknowledge Dave Waltermire of NIST for his contribution to the development of the original document.

6. IANA Considerations

This memo includes no request to IANA.

7. Security Considerations

While OVAL is just a set of data models and does not directly introduce security concerns, it does provide a mechanism by which to represent endpoint posture assessment information. This information could be extremely valuable to an attacker allowing them to learn about very sensitive information including, but, not limited to: security policies, systems on the network, criticality of systems, software and hardware inventory, patch levels, user accounts and much more. To address this concern, all endpoint posture assessment information should be protected while in transit and at rest. Furthermore, it should only be shared with parties that are authorized to receive it.

Another possible security concern is due to the fact that content expressed as OVAL has the ability to impact how a security tool operates. For example, content may instruct a tool to collect certain information off a system or may be used to drive follow-up actions like remediation. As a result, it is important for security tools to ensure that they are obtaining OVAL content from a trusted source, that it has not been modified in transit, and that proper

validation is performed in order to ensure it does not contain malicious data.

8. Change Log

8.1. -00 to -01

There are no textual changes associated with this revision. This revision simply reflects a resubmission of the document so that it remains in active status.

9. References

9.1. Normative References

[IEEE-STD-802-2001]

IEEE, "IEEE Std 802-2001 - IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture", 1999, <http://ieeexplore.ieee.org/servlet/opac?punumber=4039949>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

9.2. Informative References

[OVAL-WEBSITE] The MITRE Corporation, "The Open Vulnerability and Assessment Language", 2015, <http://ovalproject.github.io/>.

Authors' Addresses

Michael Cokus
The MITRE Corporation
903 Enterprise Parkway, Suite 200
Hampton, VA 23666
USA

Email: msc@mitre.org

Daniel Haynes
The MITRE Corporation
202 Burlington Road
Bedford, MA 01730
USA

Email: dhaynes@mitre.org

David Rothenberg
The MITRE Corporation
202 Burlington Road
Bedford, MA 01730
USA

Email: drothenberg@mitre.org

Juan Gonzalez
Department of Homeland Security
245 Murray Lane
Washington, DC 20548
USA

Email: juan.gonzalez@dhs.gov