National Institute of Standards and Technology
U.S. Department of Commerce

Special Publication 800-126 (Draft)

The Technical Specification for the Security Content Automation Protocol (SCAP)

Recommendations of the National Institute of Standards and Technology

Christopher Johnson
Stephen Quinn
Karen Scarfone
David Waltermire

NIST Special Publication 800-126 (Draft)

The Technical Specification for the Security Content Automation Protocol (SCAP) (Draft)

Recommendations of the National Institute of Standards and Technology

Christopher Johnson
Stephen Quinn
Karen Scarfone
David Waltermire

C O M P U T E R   S E C U R I T Y

Computer Security Division
Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899-8930

July 2009

U.S. Department of Commerce

Gary Locke, Secretary

National Institute of Standards and Technology

Dr. Patrick D. Gallagher, Deputy Director

Reports on Computer Systems Technology

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) promotes the U.S. economy and public welfare by providing technical leadership for the nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analysis to advance the development and productive use of information technology. ITL's responsibilities include the development of technical, physical, administrative, and management standards and guidelines for the cost-effective security and privacy of sensitive unclassified information in Federal computer systems. This Special Publication 800-series reports on ITL's research, guidance, and outreach efforts in computer security and its collaborative activities with industry, government, and academic organizations.

National Institute of Standards and Technology Special Publication 800-126 (Draft)
Natl. Inst. Stand. Technol. Spec. Publ. 800-126, 42 pages (Jul. 2009)

Acknowledgments

The authors, Christopher Johnson, Stephen Quinn, Karen Scarfone, and David Waltermire of the National Institute of Standards and Technology (NIST), wish to thank their colleagues who reviewed drafts of this document and contributed to its technical content. The authors would like to acknowledge David Niemoller and Greg Witte of G2, Andy Bove of SecureAcuity, Jim Ronayne of Cobham, and Paul Cichonski, Angela Orebaugh, and Victoria Thompson of Booz Allen Hamilton, for their keen and insightful assistance throughout the development of the document. Additional acknowledgments will be added to the final version of the publication.

Table of Contents

List of Appendices

List of Tables

Executive Summary

The National Institute of Standards and Technology (NIST) produced this publication to facilitate the development of Security Content Automation Protocol (SCAP) tools and content. SCAP seeks to standardize system security management, promote interoperability of security products, and foster the use of standard expressions of security content. This technical specification describes the requirements and conventions that are to be employed to ensure the consistent and accurate exchange of SCAP content and the ability of the content to reliably operate on SCAP validated tools.

This specification is technically oriented, and assumes that readers possess a basic understanding of system security. The use cases described in this document do not represent an exhaustive list of all possible applications of SCAP. This publication describes the details of how the elements of SCAP interoperate and defines SCAP Version 1.0 in terms of both its component specifications and the requirements for SCAP content.

SCAP 1.0 is comprised of the six specifications referenced in Section 2: XCCDF, OVAL, CPE, CCE, CVE, and CVSS. These specifications are grouped into the following three categories:

- Languages. The SCAP languages provide standard vocabularies and conventions for expressing security policy, technical check mechanisms and assessment results.
- Enumerations. Each SCAP enumeration defines a standard nomenclature (naming format) and an official dictionary or list of items expressed using that nomenclature. For example, CVE provides common identifiers and a reference list for publicly known software flaw vulnerabilities.
- Vulnerability measurement and scoring systems. In SCAP, this refers to evaluating specific characteristics of a vulnerability and, based on those characteristics, generating a score that reflects the vulnerability's severity.

SCAP is a suite of specifications that use the eXtensible Markup Language (XML) to standardize the format and nomenclature by which security software products communicate information about software flaws and security configurations. SCAP includes software flaw and security configuration standard reference data, also known as *SCAP content*. This reference data is provided by the National Vulnerability Database (NVD),[1] which is managed by NIST and sponsored by the Department of Homeland Security (DHS). SCAP is a multi-purpose protocol that supports automated vulnerability checking, technical control compliance activities, and security measurement. The U.S. Federal Government, in cooperation with academia and private industry, is adopting SCAP and encourages its use in support of security automation activities and initiatives[2].

SCAP is achieving widespread adoption by major software and hardware manufacturers and has become a significant component of large information security management and governance programs. The protocol is expected to evolve and expand in support of the growing need to define and measure effective security controls, assess and monitor ongoing aspects of that information security, and to successfully manage systems in accordance with the risk management frameworks such as the one described in NIST Special Publication 800-53[3].

[1] The National Vulnerability Database can be found at http://nvd.nist.gov/ .
[2] Refer to http://www.whitehouse.gov/omb/memoranda/fy2008/m08-22.pdf
[3] The Risk Management Framework is described in Section 3.0 of NIST Special Publication 800-53, available at http://csrc.nist.gov/publications/drafts/800-53/800-53-rev3-FPD-clean.pdf

The authors hope to encourage, by detailing the specific and appropriate usage of the SCAP 1.0 components and their interoperability, the creation of reliable and pervasive SCAP content and to foster a wide array of tools that leverage SCAP capabilities.

1. Introduction

This document defines the technical specification for Version 1.0 of the Security Content Automation Protocol (SCAP). SCAP (pronounced S-CAP) consists of a suite of specifications for standardizing the format and nomenclature by which security software communicates information about software flaws and security configurations. This document describes the basics of the SCAP component specifications and their interrelationships, the characteristics of SCAP content, as well as SCAP requirements not defined in the individual SCAP component specifications. This guide provides recommendations on how to use SCAP to achieve security automation for organizations seeking to implement SCAP.

The scope of this document is limited to SCAP 1.0 and its component specifications. Other versions of SCAP and the component specifications, including emerging specifications and future versions of SCAP, are not addressed here.

1.1 Authority

The National Institute of Standards and Technology (NIST) developed this document in furtherance of its statutory responsibilities under the Federal Information Security Management Act (FISMA) of 2002, Public Law 107-347.

NIST is responsible for developing standards and guidelines, including minimum requirements, for providing adequate information security for all agency operations and assets; but such standards and guidelines shall not apply to national security systems. This guideline is consistent with the requirements of the Office of Management and Budget (OMB) Circular A-130, Section 8b(3), "Securing Agency Information Systems," as analyzed in A-130, Appendix IV: Analysis of Key Sections. Supplemental information is provided in A-130, Appendix III.

This guideline has been prepared for use by Federal agencies. It may be used by nongovernmental organizations on a voluntary basis and is not subject to copyright, though attribution is desired.

Nothing in this document should be taken to contradict standards and guidelines made mandatory and binding on Federal agencies by the Secretary of Commerce under statutory authority, nor should these guidelines be interpreted as altering or superseding the existing authorities of the Secretary of Commerce, Director of the OMB, or any other Federal official.

1.2 Audience

This document is intended for three primary audiences.

- Content authors and editors seeking guidance to ensure that the SCAP content they produce operates correctly, consistently and reliably in SCAP tools.
- Software developers and system integrators seeking to create, use, or exchange SCAP content in their products or service offerings.
- Content and/or tool developers preparing for SCAP validation at an accredited independent testing laboratory.

1.3 Document Structure

The remainder of this document is organized into the following four major sections:

- Section 2 defines SCAP 1.0 and explains the purpose of SCAP.
- Section 3 presents basic information on the specifications comprising SCAP 1.0.
- Section 4 defines conventions and requirements for using SCAP to achieve interoperability of content and tools.
- Section 5 presents use cases that demonstrate effective and compliant implementations of SCAP.

The document also contains appendices with supporting material:

- Appendix A contains an acronym and abbreviation list.
- Appendix B lists references and other resources related to SCAP 1.0 and its component specifications.
- Appendix C documents several SCAP extensions to the XCCDF component.

1.4 Document Conventions

Some of the requirements and conventions used in this document reference XML content. These references come in two forms, inline and indented. An example of inline references is

"A `<cpe_dict:cpe-item>` may contain `<cpe_dict:check>` elements that reference OVAL definitions".

In this example the notation `<cpe_dict:cpe-item>` can be replaced by the more verbose equivalent "the XML element whose qualified name is cpe_dict:cpe-item". An even more verbose equivalent is "the XML element in the namespace 'http://cpe.mitre.org/dictionary/2.0' whose local name is cpe-item".

An example of an indented reference is:

> "References to OVAL definitions are expressed using the following format:
>
> `<cpe_dict:check system=`
> `"http://oval.mitre.org/XMLSchema/oval-definitions-5"`
> `href="`*Oval_URL*`">[`*Oval_inventory_definition_id*`]</cpe_dict:check>`".

Indented references are intended to represent the form of actual XML content. Indented references represent literal content by the use of a `fixed-length font`, and parametric (freely replaceable) content by the use of an *italic font*. Square brackets '`[]`' are used to designate optional content. Thus "`[Oval_inventory_definition_id]`" designates optional parametric content.

Both inline and indented forms use qualified names to refer to specific XML elements. A qualified name associates a named element with a namespace. The namespace identifies the specific XML schema that defines (and consequently may be used to validate) the syntax of the element instance. A qualified name declares this schema to element association using the format '*prefix*:*element-name*'. The association of prefix to namespace is defined in the metadata of an XML document and generally will vary from document to document. In this specification, the conventional mappings listed in Table 1-1 are used.

Table 1-1. Conventional XML Mappings

Prefix	Namespace URI	Schema
cpe_dict	http://cpe.mitre.org/dictionary/2.0	CPE Dictionaries
cpe	http://cpe.mitre.org/language/2.0	Embedded CPE references
nvd	http://scap.nist.gov/schema/feed/vulnerability/2.0	Base schema for NVD data feeds
cve	http://scap.nist.gov/schema/vulnerability/0.4	NVD/CVE data feed elements and attributes
cvss	http://scap.nist.gov/schema/cvss-v2/0.2	NVD/CVSS data feed elements and attributes
dc	http://purl.org/dc/elements/1.1/	Simple Dublin Core elements.
xccdf	http://checklists.nist.gov/xccdf/1.1	XCCDF policy documents
xml	http://www.w3.org/XML/1998/namespace	Common XML attributes.
oval	http://oval.mitre.org/XMLSchema/oval-common-5	Common OVAL elements and attributes
oval-def	http://oval.mitre.org/XMLSchema/oval-definitions-5	OVAL definitions
xxxx-def	http://oval.mitre.org/XMLSchema/oval-definitions-5#*xxxx*	OVAL elements and attributes specific to an OS, Hardware or Application type *xxxx*[4]
oval-res	http://oval.mitre.org/XMLSchema/oval-results-5	OVAL results
oval-sc	http://oval.mitre.org/XMLSchema/oval-system-characteristics-5	OVAL system characteristics
oval-*xxxx-sc*	http://oval.mitre.org/XMLSchema/oval-system-characteristics-5#*xxxx*	OVAL system characteristic elements and attributes specific to an OS, Hardware or Application type *xxxx*
oval-var	http://oval.mitre.org/XMLSchema/oval-variables-5	The elements, types, and attributes that compose the core schema for encoding Open Vulnerability and Assessment Language (OVAL) Variables. This schema provided to give structure to any external variables and their values that an OVAL definition is expecting
sch	http://purl.oclc.org/dsdl/schematron	Schematron validation scripts
dsig	http://www.w3.org/2000/09/xmldsig#	Interoperable XML digital signatures

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 Request for Comment (RFC) 2119.[5]

[4] The types supported by OVAL 5.3 include the AIX, CATOS, ESX, FREE BSD, HP-UX, IOS, LINUX, PIXOS, SOLARIS, UNIX, WINDOWS, INDEPENDENT (common) operating systems, and APACHE application.
[5] RFC 2119, “Key words for use in RFCs to Indicate Requirement Levels”, is available at http://www.ietf.org/rfc/rfc2119.txt.

2. Overview of SCAP 1.0

NIST Special Publication 800-117, *Guide to Adopting and Using the Security Content Automation Protocol,*[6] defines the SCAP as being comprised of two major elements. [BAR09] First, SCAP is a protocol—a suite of six specifications that standardize the format and nomenclature by which security software communicates information about software flaws and security configurations embedded in XML. Second, SCAP also includes software flaw and security configuration standard reference data, also known as *SCAP content*. This reference data is provided by the National Vulnerability Database (NVD),[7] which is managed by NIST and sponsored by the Department of Homeland Security (DHS). SCAP can be used for several purposes, including automating vulnerability checking, technical control compliance activities, and security measurement. The U.S. Federal Government, in cooperation with academia and private industry, is adopting SCAP and is encouraging widespread support of it.

This document defines Version 1.0 of SCAP in terms of both its component specifications and the requirements for SCAP content. Organizations that use SCAP 1.0 should ensure that their use of it is compliant with the information presented in this document.

SCAP 1.0 uses the following specifications:

- Extensible Configuration Checklist Description Format (XCCDF) 1.1.4, a language for authoring security checklists/benchmarks and for reporting results of checklist evaluation [QUI08]
- Open Vulnerability and Assessment Language (OVAL) 5.3, a language for representing system configuration information, assessing machine state, and reporting assessment results
- Common Platform Enumeration (CPE) 2.2, a nomenclature and dictionary of hardware, operating systems and applications [BUT09]
- Common Configuration Enumeration (CCE) 5, a nomenclature and dictionary of security software configurations
- Common Vulnerabilities and Exposures (CVE), a nomenclature and dictionary of security-related software flaws[8]
- Common Vulnerability Scoring System (CVSS) 2.0, an open specification for measuring the relative severity of software flaw vulnerabilities [MEL07].

Section 3 presents detailed information on each of these specifications and provides examples of how these components are used in context.

Security products and checklist authors assemble content from SCAP data repositories to create viable SCAP-expressed security guidance. A security configuration checklist that documents desired security configuration settings, installed patches, and other system security elements using SCAP in a standardized format is known as an SCAP-expressed checklist. Such a checklist would use XCCDF to describe the checklist, CCE to identify security configuration settings to be addressed or assessed, and CPE to identify platforms for which the checklist is valid. The use of CCE and CPE entries within XCCDF checklists is an example of an SCAP convention — a requirement for valid SCAP usage. Another example of an SCAP convention is the mapping of individual checks within a checklist to external requirements such as

[6] NIST SP 800-117, *DRAFT Guide to Adopting and Using the Security Content and Automation Protocol (SCAP,)* is available at http://csrc.nist.gov/publications/drafts/800-117/draft-sp800-117.pdf.
[7] The National Vulnerability Database can be found at http://nvd.nist.gov/ .
[8] CVE does not have a version number.

security controls from NIST SP 800-53, *Recommended Security Controls for Federal Information Systems*.[9] These conventions are considered part of the definition of SCAP 1.0 and are described in Sections 3, 4 and 5 of this document. Organizations producing SCAP content should adhere to these conventions to ensure the highest degree of interoperability.

SCAP revisions are managed through a coordinated process defined within the SCAP Release Cycle[10]. The release cycle workflow manages changes related to SCAP specifications and validation processes including the addition of new specifications or updates to existing specifications. This process encourages community involvement, promotes transparency and awareness regarding proposed changes, and affords ample lead-time to prepare for pending changes.

[9] NIST SP 800-53, *Recommended Security Controls for Federal Information Systems,* is available at http://csrc.nist.gov/publications/PubsSPs.html.

[10] SCAP Release Cycle, http://scap.nist.gov/timeline.html

3. Basics of SCAP Components

SCAP 1.0 is comprised of the six specifications referenced in Section 2: XCCDF, OVAL, CPE, CCE, CVE, and CVSS. These specifications are grouped into the following three categories:

- **Languages.** SCAP languages provide a standardized means for identifying what is to be evaluated and for expressing how to check system state.
- **Enumerations.** SCAP enumerations provide a standardized nomenclature (naming format) and an associated dictionary of items expressed using that nomenclature. For example, CVE provides standardized names for publicly known software flaw vulnerabilities.
- **Vulnerability measurement and scoring systems.** Ability within SCAP to measure and evaluate specific vulnerability characteristics to derive a vulnerability severity score.

This section provides an introduction to the SCAP component specifications in each of these categories.

3.1 Languages

This section describes the two language specifications in SCAP 1.0: XCCDF 1.1.4 and OVAL 5.3. For each specification, the section describes its purpose and primary logical concepts, and provides examples.

3.1.1 Extensible Configuration Checklist Description Format (XCCDF) 1.1.4

XCCDF 1.1.4 is a specification language for writing security configuration checklists, vulnerability alerts, and other related documents. The specification is designed to support information interchange, document generation, organizational and situational tailoring, automated compliance testing, and compliance scoring. An XCCDF document represents a structured collection of system review capabilities for some set of target systems. The specification also defines a data model and format for storing results of benchmark compliance testing. The intent of XCCDF is to provide a uniform means of expressing security checklists and the results of checklist evaluation.

An XCCDF document is composed of one or more XCCDF rules. An XCCDF rule is a high-level definition of a technical check on a system. A rule does not directly specify how a check should be performed, but instead points to other XML documents (such as OVAL Definition files) that contain the actual instructions for performing the check. Table 3-1 shows sample values from an XCCDF rule. This particular rule is for ensuring that the minimum password length is set to at least eight characters. The System Check section of the rule specifies the OVAL Definition example presented in Table 3-3.

Table 3-1. XCCDF Rule Sample Data

Rule Field	Explanation	Sample Data
Rule ID	The identifier for this rule	MinimumPasswordLength-8
Title	The title for the rule	Minimum Password Length = 8
Description	The description of the rule	This setting specifies the minimum length of a password in characters. The rationale behind this setting is that longer passwords are more difficult to guess and crack than shorter passwords. The downside is that longer passwords are often more difficult for users to remember.
References	References to checklists and other documents that contain requirements to which this rule maps—in this case, the IA-5 (Authenticator Management) control from NIST SP 800-53	IA-5 (http://csrc.nist.gov/publications/nistpubs/800-53/SP800-53.pdf)
Requires	The group to which this rule belongs, if any; in this case, the IA-5 group	IA-5
Schema	The XML check system schema to use during rule evaluation (usually the OVAL schema)	http://oval.mitre.org/OVAL/XMLSchema/oval
OVAL Definition File Reference	Name of the OVAL Definition file	WindowsXP-SP800-68.xml
OVAL Definition ID	The identifier of the OVAL Definition to be used	oval:gov.nist.1:def:20

The number of rules appearing in a typical XCCDF document will vary depending upon the intended use case. The rules appearing in an XCCDF document may also be organized into multiple *XCCDF profiles* that specify collections of rules to be evaluated on particular types of systems. Profiles can be used to express multiple policies within a single benchmark document; allowing the benchmark author to publish technical security control settings tailored to the type of system or the environment in which the system is deployed. By creating a policy that corresponds to a particular set of requirements, such as those of the FISMA, the Defense Information Systems Agency's (DISA) Security Technical Implementation Guides (STIG), or the Health Insurance Portability and Accountability Act (HIPAA), the policy can be used to map those high-level requirements to the corresponding OVAL Definitions.

An XCCDF document can be further organized into one or more *XCCDF groups*. A group can contain one or more related rules or groups. Groups allow multiple rules to be enabled or disabled collectively instead of individually.

Another option involving XCCDF rules is to have user-definable values for certain rules, known as *XCCDF values*. Table 3-2 shows sample data from an XCCDF value statement. This particular value statement defines the duration of the account lockout (in minutes) that occurs after consecutive failed login attempts have exceeded a specific threshold. In this case, the value has been set to 15 minutes and the operator field specifies that the system setting for lockout duration be greater than or equal to this value. A checklist user may choose to alter or override this value in the profile(s) that reference this value to account for specific organizational policies. The Lower-Bound and Upper-Bound fields establish the range of acceptable values that checklist users can enter when specifying a new value for AccountLockoutDurationTime.

Table 3-2. XCCDF Value Statement Sample Data

Rule Field	Explanation	Sample Data
Value ID	The identifier for this value	AccountLockoutDurationTime
Type	The type of the value (e.g., string, number)	Number
Operator	The comparison operator (in this case, the system's value for account lockout duration time must be greater than or equal to the specified value)	greater than or equal
Title	The title for the value	Account Lockout Duration Time
Description	The description of the value	This value specifies how long the user account should be locked out. This is often set to a low but substantial value (e.g., 15 minutes.
Question	Explanatory text that can be presented to the user when is customizing the checklist	Account lockout duration time (in minutes)
Value	The value assigned to the AccountLockoutDurationTime value	15
Default	A suggested default value number for checklist users' reference; not actually used when performing checks or applying configuration settings	15
Lower-Bound	If a checklist user alters the value number, it cannot be set lower than 10	10
Upper-Bound	If a checklist user alters the value number, it cannot be set higher than 30	30

3.1.2 Open Vulnerability and Assessment Language (OVAL) 5.3

OVAL is used to express standardized, machine-readable rules that can be used to assess the state of a system. Under SCAP, OVAL is commonly used to determine the presence of vulnerabilities and insecure configurations. A set of instructions used to check for a security problem, such as an incorrect minimum password length setting, is known as an *OVAL Definition*. A file containing one or more OVAL Definitions (often hundreds or even thousands) is known as an *OVAL Definition file*.

There are four types of OVAL Definitions:[11]

- Vulnerability definitions, which define "the conditions that must exist on a computer for a specific vulnerability to be present"
- Patch definitions, which define "the conditions on a computer that determine whether a particular patch is appropriate for a system"
- Inventory definitions, which define "the conditions on a computer that determine whether a specific piece of software is installed on the system"
- Compliance definitions, which define "the conditions on a computer that determine compliance with a specific policy or configuration statement".

[11] These definitions are taken from the OVAL Web site's "Structure of the Language" page, located at http://oval.mitre.org/language/about/structure.html.

Table 3-3 shows sample values that have been extracted from an actual OVAL compliance definition. Explanations of each value have also been provided. The definition ID, version, and class are standard fields that are part of every OVAL Definition. The exact types of information contained in the metadata vary among definitions, but at a high level they explain the intent of the definition. The criteria provide the technical details of how the system will be checked for the items of interest, such as the presence of a vulnerability or the value of a configuration setting. Each OVAL Definition has a single top-level criterion that can contain one or more sub-criteria. The operator associated with each criterion specifies how the results produced by the sub-criteria are combined (e.g., AND, OR).

The example in Table 3-3 has two criteria. One of the criteria is an *OVAL Test*, which is a specific system check—in this case, that the system is configured to require a minimum password length of at least eight characters. The other criterion is actually another definition—in this case, an inventory definition that confirms that the target system is running Windows XP SP2 on a 32-bit architecture. The ordering of the criteria ensures that the inventory definition is evaluated before the test is performed. This ordering is essential since the test results may be invalid if run on a different operating system version.

Table 3-3. OVAL Definition Sample Data

Definition Field	Explanation	Sample Data
ID	Identifier for this definition; must be unique within the OVAL Definition file	oval:gov.nist.1:def:20
Version	Version of the definition	1
Class	Defines the type of definition (e.g., compliance, inventory, patch, vulnerability)	Compliance
Metadata		
Title	Short description for the definition	Minimum Password Length of 8 Characters
Affected product	The operating system or application version(s) to which this definition is applicable	Microsoft Windows XP, SP2, 32 bit
References	References to checklists and other documents that contain requirements to which this definition maps	NIST SP800-68 Appendix A, 1.4b, http://csrc.nist.gov/itsec/download_WinXP.html DISA FSO Checklist, 5.4.1.3 DISA VMS 6XID V0001106 DISA PDI ID 1740
Description	Description for the definition	The minimum allowable password length is 8 characters
NIST	Identifies NIST SP 800-53 security controls to which this definition maps—in this case, IA-5, which is Authenticator Management[12]	IA-5

[12] The XCCDF profile that references this OVAL Definition also specifies IA-5 as the NIST SP 800-53 mapping. This has been done in both the XCCDF document and OVAL Definition file in case either file is used without the other.

Definition Field	Explanation	Sample Data
Additional references	References to additional security configuration approaches that contain requirements to which this definition maps	NSA NT Guide: Chap 5, p. 30; NSA WIN2K Guide, Group Policy: Security Configuration Toolset: Chap. 3, p. 22; NSA XP Guide: Chap. 4, p. 21; DODD 8500.1 Para 4.18; DODI 8500.2 DCCS-2, DCSC-1; CJCSM 6510.01 App. A, Enclosure A, Para. 5.b (8)
Criteria		
Definition reference	The identifier of another OVAL Definition, OVAL definition references another OVAL definition (extended definition)	oval:gov.nist.1:def:9
Definition comment	A brief explanation of what the definition addresses; in this case, it is used to determine if the target system is running Windows XP SP2 on a 32-bit architecture	Precondition 9: Windows family, Windows XP, SP2, 32 bit
Test reference	An identifier for an OVAL Test that is run when evaluating the OVAL definition.	oval:gov.nist.1:tst:16
Test comment	A brief explanation of what the test addresses; in this case, it is used to determine if the target system requires a minimum password length of 8 characters	Minimum password length is 8 characters

As the example in Table 3-3 shows, definitions often reference one or more tests. The instructions that comprise each test are also included in the OVAL Definition file. A test does not directly contain the technical details of checking the system but instead references other OVAL constructs. Typically, a test references an *OVAL Object*, which is a logical construct for a portion of the target system (e.g., password policy, file, Windows registry key), and an *OVAL State*, which is a particular check of the specified OVAL object (e.g., verifying that the password policy requires a minimum password length of at least eight characters, verifying the existence of a file). An OVAL State can also reference one or more OVAL Variables, which are user-definable values (e.g., minimum password length value of eight). This modular approach introduces additional complexity but fosters reuse and allows OVAL Definitions to be used without requiring the details of test construction to be exposed. Individuals seeking detailed information can refer to the OVAL Definition file for the definition, Test, Object, and State ID numbers, and instructions associated with each entity. More technical details on OVAL Definition files, including examples of the XML code for OVAL Definitions, are presented in Section 4. An OVAL Definition tutorial is also available from the OVAL Web site at http://oval.mitre.org/language/about/definition.html.

3.2 Enumerations

This section describes the three enumeration specifications in SCAP 1.0: CPE 2.2, CCE 5, and CVE. SCAP enumerations typically consist of an identifier, an associated description or definition, and a list of supporting references. For each specification, the section describes its purpose and provides examples of entries. The section also explains the interdependencies between these specifications and other SCAP component specifications.

3.2.1 Common Platform Enumeration (CPE) 2.2

CPE 2.2 is a standard naming convention for operating systems, hardware, and applications. The purpose of CPE is to provide consistent, easily parsed names that can be shared by multiple parties and solutions to refer to the same specific platform type[13].

The syntax of an individual CPE Name, as defined in Section 5 of the CPE 2.2 Specification, is as follows:

```
cpe:/{part}:{vendor}:{product}:{version}:{update}:{edition}:{language}
```

For example, "`cpe:/o:redhat:enterprise_linux:2.1::es`" refers to Red Hat Enterprise Server 2.1. The "o" indicates that this CPE describes an operating system. In this example, the edition field is blank, indicating that this CPE refers to all editions of Red Hat Enterprise Server 2.1.

CPE Names are used in conjunction with many of the SCAP specifications to provide an association to asset-related information. CPE is used by SCAP in the following ways:

- **XCCDF** – In an XCCDF checklist, CPE Names can be used to identify the hardware or software platform to which an XCCDF object (e.g., benchmark, profile, group, rule) applies.
- **CCE** – CPE Names can be associated with configuration vulnerabilities to identify platforms covered by CCE technical mechanisms.
- **CVE** – CVEs are related to one or more product platforms expressed as CPEs. The mapping of CPEs to CVEs is performed by NVD analysts and is published in the NVD vulnerability data feed.

3.2.2 Common Configuration Enumeration (CCE) 5

The CCE 5 naming scheme is a dictionary of names for security configuration settings for deployed software. Each type of security-related configuration issue is assigned a unique identifier to facilitate fast and accurate correlation of configuration data across multiple information sources and tools. MITRE publishes an XML schema for the CCE[14].

There are five attributes in a CCE entry: a unique identifier number, a description of the configuration issue, logical parameters of the CCE, the associated technical mechanisms related to the CCE, and references to additional sources of information. Figure 1 provides an example of these attributes for a CCE 5 entry for Windows XP:

[13] The MITRE Corporation maintains the CPE specification and NIST maintains the Official CPE Dictionary. More information on CPE is available at http://cpe.mitre.org/. The Official CPE Dictionary is available at http://nvd.nist.gov/cpe.cfm

[14] See http://cce.mitre.org for additional information.

Figure 1. Example CCE Entry

CCE ID:	CCE-3108-8
Definition:	The correct service permissions for the Telnet service should be assigned.
Parameters:	(1) set of accounts (2) list of permissions
Technical Mechanisms:	(1) set via Security Templates (2) defined by Group Policy
References:	Listed at http://cce.mitre.org/lists/cce_list.html

References to CCEs are used by some of the other SCAP specifications to provide an association to particular security configuration settings. In an XCCDF checklist, CCEs can be used to specify which security configuration settings are of interest (i.e., which settings should be checked). Similarly, OVAL uses CCE entries to relate specific definitions to an actual configuration setting.

3.2.3 Common Vulnerabilities and Exposures (CVE)

CVE is a dictionary of unique, common names for publicly known software flaws[15]. This common naming convention allows sharing of data within and among organizations and enables effective integration of services and tools. For example, a remediation tool may use CVE information from several scanning tools and monitoring sensors, enabling an integrated risk mitigation solution. CVE provides the following:

- A comprehensive list of publicly known software flaws
- A globally unique name to identify each vulnerability
- A basis for discussing priorities and risks of vulnerabilities
- A way for a user of disparate tools and services to integrate vulnerability information

A CVE vulnerability entry consists of a unique name (e.g., CVE-2000-0001), a short description (e.g., "RealMedia server allows remote attackers to cause a denial of service via a long ramgen request."), and references to public advisories on the vulnerability.

CVE is used in conjunction with other SCAP specifications to satisfy the following use cases:

- **XCCDF.** In an XCCDF checklist, CVEs are used to uniquely identify which software flaw vulnerabilities are of interest (i.e., flaws that are to be checked during the evaluation of the checklist).
- **CVSS.** CVSS scores are associated with CVE entries to uniformly express the fundamental characteristics of the software flaw and to provide a severity score based on these characteristics.
- **OVAL.** Including the specific CVE entry in the OVAL metadata enables a reviewer to accurately understand the basis for a given OVAL definition such as a Vulnerability or Patch test

[15] CVE issuance is managed by The MITRE Corporation and is sponsored by the DHS National Cyber Security Division (NCSD). General CVE information is available at http://cve.mitre.org

Working with researchers, MITRE assigns CVE IDs to publicly known vulnerabilities in commercial and open source software. The CVE repository maintained by NIST contains all CVEs issued by MITRE as well as supplemental data such as CVSS base scores, Common Weakness Enumeration (CWE) identifiers, vendor statements, and Spanish language translations. NVD provides fine-grained searching and statistical analysis capabilities as well[16].

3.3 Common Vulnerability Scoring System (CVSS) 2.0

CVSS 2.0 provides a repeatable method for consistently evaluating and expressing the risk associated with a given software flaw (e.g., CVE). The use of this shared scoring model allows meaningful comparisons of vulnerability severity scores. CVSS provides three metric groups that can be used to derive a vulnerability score:

- Base, which uses the intrinsic characteristics of the vulnerability to provide a generic score
- Temporal, which captures external factors that may change over time (e.g., availability of exploit code). The base score is adjusted to render a temporal score that accounts for the temporal factors
- Environmental, which characterizes the severity of a vulnerability in the context of an organization's operating environment

The purpose of performing CVSS scoring is to help organizations understand the relative importance of various vulnerabilities so that they can effectively assess, prioritize and mitigate vulnerabilities. Because hundreds of vulnerabilities are publicly announced every week, it is important for organizations to have an easy way to identify those vulnerabilities that have the greatest operational impact. NVD analysts compute and publish CVSS base scores for all CVEs, but organizations are encouraged to further tailor these scores by employing the temporal and environmental metrics to more precisely measure the risk a vulnerability represents within their specific organization.

Complete examples of CVSS measures and scores are available in the official CVSS 2.0 specification [MEL07]. A brief example of base measures, extracted from [MEL07], is [AV:N/AC:L/Au:N/C:C/I:C/A:C], with a base score of 10.0. The bracketed notation for the base measures is known as a *vector*. The first half of the notation indicates that the Access Vector is Network, the Access Complexity is Low, and the Authentication requirement is None. The second half of the notation indicates that the potential impact to Confidentiality, Integrity, and Availability is Complete. The scoring scale is 0 to 10, with 10 being the most severe, so a score of 10.0 indicates the highest severity possible.

The CVSS Special Interest Group (CVSS-SIG) from the Forum of Incident Response and Security Teams (FIRST) developed CVSS 2.0. More information on CVSS can be found at http://www.first.org/cvss.

[16] CVEs and associated NIST-provided metadata can be viewed at http://nvd.nist.gov/nvd.cfm

4. SCAP General Requirements and Conventions

As described in NIST Special Publication 800-117, *Guide to Adopting and Using the Security Content Automation Protocol,*[17] the motivation for creating SCAP was to provide a standardized approach to maintaining the security of enterprise systems, enhance interoperability of security products, and enable consistent security assessments. The following conventions and requirements were established to help satisfy these goals by ensuring that validated tools and content interoperate as designed and provide the expected results.

4.1 XCCDF Conventions and Requirements

An SCAP XCCDF stream is the overarching machine-readable XML document that defines the policies and test conditions to be evaluated or applied. Compliant XCCDF streams include Definition documents that state the policy and Result documents that contain both policy statements and actual test results.

An SCAP Benchmark document validates against the XCCDF schema (http://nvd.nist.gov/scap/xccdf/docs/xccdf-1.1.4.xsd) and conforms to all relevant content requirements as outlined in the XCCDF Specification [QUI08].

In cases where localized text is used, US English is the default language and systems SHOULD test for lang="en-US".

4.1.1 Metadata

XCCDF metadata provides descriptive information about the data stream. The metadata is used by SCAP tools to assist in the selection of the appropriate benchmark, ensure that the most recent or correct version of a benchmark is used, and to provide additional information about the benchmark. The following metadata elements SHALL be included in an SCAP XCCDF document:

- `<xccdf:title>` - a title that indicates the purpose of the benchmark.
- `<xccdf:description>` - a comment regarding the purpose and intended audience of the benchmark.
- `<xccdf:notice>` - clarifications, suggestions, or warnings regarding the use of the benchmark, including but not limited to terms of use, legal notices or copyright statements.
- `<xccdf:version>` - an indicator of a particular revision of the benchmark.
- `<xccdf:reference>` - a reference to additional information, preferably including a URL, to obtain additional information regarding the benchmark.

4.1.2 XCCDF and CPE Dependencies

For all SCAP content, the applicability of XCCDF `<xccdf:Benchmark>` elements to specific IT platforms SHALL be specified using Common Platform Enumeration (CPE) Names.

[17] NIST SP 800-117, *DRAFT Guide to Adopting and Using the Security Content and Automation Protocol (SCAP,)* is available at http://csrc.nist.gov/publications/drafts/800-117/draft-sp800-117.pdf.

CPE Names used within an XCCDF benchmark, SHALL match the names of existing Official CPE Dictionary[18] entries where possible. If multiple matches are found within the dictionary (e.g., deprecated and current CPE Names), the most current CPE Name SHOULD be used.

The XCCDF benchmark SHOULD bind every rule to one or more CPEs. Each rule bound to a CPE Name SHOULD be declared in the required CPE dictionary stream and each OVAL inventory class definition referenced from the dictionary stream SHOULD be specified in the required CPE inventory stream.

4.1.3 XCCDF and CCE Dependencies

XCCDF `<xccdf:Rule>` elements MAY be used to define a policy requiring compliance with a specific configuration setting. When a configuration setting having one or more associated CCE Identifiers from the CCE List is expressed as an XCCDF rule, an `<xccdf:ident>` element[19] reference SHALL be provided within the `<Rule>` element. The `<xccdf:ident>` element provides a globally unique identifier for a specific configuration setting.

The `<xccdf:ident>` element syntax SHALL be used as follows:

1. The system attribute for the `<xccdf:ident>` element SHALL be defined using the CCE Version 5 system identifier "*http://cce.mitre.org*".
2. The *CCE Identifier* SHALL be used for the `<xccdf:ident>` element content.

For example:

```
<Rule id="AuditAccountLogonEvents">
      <title>Audit Account Logon Events</title>
      …
      <ident system="http://cce.mitre.org">CCE-3867-0</ident>
      <ident system="http://cce.mitre.org">CCE-3008-0</ident>
      …
</Rule>
```

A rule result of "pass" indicates that the target platform complies with the configuration setting guidance expressed in the XCCDF rule.

4.1.4 XCCDF and CVE Dependencies

XCCDF `<xccdf:Rule>` elements MAY be used to assess security related software flaws. When this assessment is associated with one or more associated CVE Identifiers from the CVE vulnerability feeds, an `<xccdf:ident>` element[20] reference within the `<xccdf:Rule>` element SHALL be provided.

The `<xccdf:ident>` element syntax SHALL be used as follows:

1. The system attribute for the `<xccdf:ident>` element SHALL be defined using the CVE system identifier "*http://cve.mitre.org*".
2. The *CVE Identifier* SHALL be used for the `<xccdf:ident>` element content.

[18] The Official CPE Dictionary is located at http://nvd.nist.gov/cpe.cfm

[19] See NIST IR 7275r3, The XCCDF Specification version 1.1.4, p.21 table, p.22 paragraph 5, and p.59 section "<ident>" for additional details.

[20] See NIST IR 7275r3, The XCCDF Specification version 1.1.4, p.21 table, p.22 paragraph 5, and p.59 section "<ident>" for additional details.

For example:

```
<Rule id="SQLInjectionVulnerability"
      <title>SQL Injection Vulnerability</title>
      …
      <ident system="http://cve.mitre.org">CVE-2008-6865</ident>
      <ident system="http://cve.mitre.org">CVE-2008-6866</ident>
      …
</Rule>
```

A rule result of "pass" indicates that the target platform satisfies all the conditions of the XCCDF rule and is unaffected by the vulnerability or exposure referenced by the CVE.

4.1.5 XCCDF/OVAL Definition Dependencies

A rule MAY refer to one or more OVAL Definitions to implement the technical tests necessary to determine the pass/fail status of the rule. Embedded OVAL Definitions are not supported by SCAP XCCDF. References from SCAP compliant XCCDF to OVAL Definitions SHALL use the form:

```
<check-content-ref href="OVAL_Source_URI" [name="OVAL_Definition_Id"]/>
```

The `href` attribute identifies the OVAL Definition XML stream. When present, the optional `name` attribute refers to a specific OVAL definition in the designated content stream. When an XCCDF rule references a specific OVAL Definition, an OVAL Definitions source SHALL be available to resolve the reference.

SCAP stylistic conventions specify that the optional `name` attribute be omitted if the rule is designed to evaluate the current patch level of the target platform. The following rule specification is an example of this convention:

```
<Rule id="SecurityPatchesUpToDate" selected="false" weight="10.0">
      <title>Security Patches Up-To-Date</title>
      <description>Keep systems up to current patch levels</description>
      <check system="http://oval.mitre.org/XMLSchema/oval-definitions-5">
            <check-content-ref href="scap-win2000-patches.xml"/>
      </check>
</Rule>
```

In the previous example, the `<xccdf:check-content-ref>` element's `href` attribute refers to an OVAL Definitions stream containing one or more OVAL patch definitions. This `check-content-ref` is equivalent to *referencing* a virtual OVAL Definition of the form:

```
<oval_definitions xmlns:oval-def="http://oval.mitre.org/XMLSchema/oval-definitions-5">
 <definitions>
      <definition id="identifier of patch definition" version="0" class="patch">
        …
        <criteria >
         <extend_definition definition_ref="identifier of patch definition 1"/>
         …
         <extend_definition definition_ref="identifier of patch definition N"/>
        </criteria>
```

```
    </definition>
</oval_definitions>
```

where the extended definitions are the individual patch definitions defined in the OVAL content stream.

If any `<xccdf:Rule>` references an OVAL patch definition, a patch scan source SHALL be used to resolve the reference.

4.1.6 XCCDF/OVAL Variable Dependencies

Content authors SHOULD refrain from hard coding assessment values into the OVAL Definitions to maximize the flexibility and reuse of OVAL modules. The recommended approach is to define these values as XCCDF value parameters. An acceptable alternative is to represent these values as discrete OVAL Variables or within an OVAL Variables file.

When the OVAL Definition(s) referenced from a rule require one or more external variable bindings, the check-export element(s) that define the binding of XCCDF values to OVAL variables SHALL precede the check-content-ref element. The format of these elements is:

```
<check-export xmlns=http://checklists.nist.gov/xccdf/1.1
      value-id="XCCDF_Value_id" export-name="OVAL_External_Variable_id"/>
```

The following check element example demonstrates the use of this convention:

```
<check system="http://oval.mitre.org/XMLSchema/oval-definitions-5">
      <check-export export-name="oval:gov.nist.fdcc.xp:var:66711"
      value id="NoSlowLink_var"/>
      <check-export export-name="oval:gov.nist.fdcc.xp:var:66712"
      value-id="NoBackgroundPolicy_var"/>
      <check-export export-name="oval:gov.nist.fdcc.xp:var:66713"
      value-id="NoGPOListChanges_var"/>
      <check-content-ref href="fdcc-winxp-oval.xml" name="oval:gov.nist.fdcc.xp:def:6671"/>
</check>
```

The type and value binding of the specified XCCDF Value is constrained to match that lexical representation of the indicated OVAL Variable Data Type. SCAP compliant tools that process variable content are required to support the range and precision of the int and float data types. Table 4-1 summarizes the constraints regarding data type usage.

Table 4-1. XCCDF-OVAL Data Export Matching Constraints

OVAL Data Type	Matching XCCDF Data Type
Int	number
Float	number
Boolean	boolean
string, EVR string, version, IOS_version, fileset_revision	string

4.1.7 XCCDF Test Results

XCCDF test results are documented as the contents of a `<xccdf:TestResult>` element that either stands alone as the root of an XML document or is embedded as a child-element of a `<xccdf:Benchmark>` root element. In the latter case, the associated benchmark is the embedding benchmark; in the former, the `<xccdf:TestResults>` document requires an embedded `<xccdf:Benchmark>` element that identifies the associated benchmark. `<xccdf:Benchmark>` elements are ignored in `<xccdf:TestResult>` elements that are embedded in their associated benchmark.

To be considered valid SCAP content, the following conditions SHALL be met:

- When using a profile during the processing of XCCDF content, the test results SHALL embed a `<xccdf:Profile>` element that identifies the non-abstract profile in the associated benchmark whose evaluation results are reported by the test results.
- Reported rule results SHALL include all selected rules within the specified Profile.
- Reported value-settings SHALL include all those values that are exported by the reported rules. The specific settings are those determined by the reported Profile.
- The `<identity>` tag identifies the security principal used to access rule evaluation on the target(s).
- The `<rule-result>` elements SHALL report the result of the application of each selected rule against all specified targets. The `rule_idref` attribute of the `<xccdf:rule-result>` SHALL identify the selected rule and each `<xccdf:instance>` elements SHALL identify the corresponding `<xccdf:target>` element.

4.1.8 Assigning CVE Identifiers to Rule Results

The XCCDF `<xccdf:rule-result>` element provides data indicating the result of assessing a system using the identified XCCDF `<xccdf:Rule>` element. If the target XCCDF `<xccdf:Rule>` identified by the `<rule-result>` `rule-idref` attribute has one or more `<ident>` elements[21] with the "*http://cve.mitre.org*" system identifier, then each `<xccdf:ident>` element SHOULD also appear within the `<xccdf:rule-result>` element.

For example:

```
<rule-result idref="minimum_password_length"
      xmlns="http://checklists.nist.gov/xccdf/1.1">
      …
      <cdf:Rule id="java-upgrade-278" selected="1" weight="0.5">
      <cdf:title>Java Bug Fix Upgrade Installed</cdf:title>
      <cdf:ident system="http://cve.mitre.org/> CVE-2006-0614 </cdf:ident>
      …
</rule-result>
```

[21] See NIST IR 7275r3, The XCCDF Specification version 1.1.4, p.30 table and p.59 section "<ident>".

4.1.9 Assigning CCE Identifiers to Rule Results

The XCCDF `<xccdf:rule-result>` element provides data indicating the result of assessing a system using the identified XCCDF `<xccdf:Rule>` element. If the target XCCDF `<xccdf:Rule>` identified by the `<xccdf:rule-result>` `rule-idref` attribute has one or more `<xccdf:ident>` elements with the "*http://cce.mitre.org*" system identifier, then each `<xccdf:ident>` element SHOULD also appear within the `<rule-result>` element. For example:

```
<rule-result idref="minimum_password_length"
      xmlns="http://checklists.nist.gov/xccdf/1.1">
      …
      <ident system="http://cce.mitre.org">CCE-2981-9</ident>
      …
</rule-result>
```

4.1.10 Mapping OVAL Results to XCCDF Results

When a `<xccdf:Rule>` element references an OVAL Definition, the `<xccdf:rule-result>` that results from the application of that rule specifies an XCCDF rule result that is mapped from the OVAL Definition Result. This result is calculated by applying the referenced OVAL Definition to a target platform.

In some cases the derived results may seem counterintuitive, but when viewed in the appropriate context the underlying logic is evident. For example, if an OVAL Definition of class "compliance" is processed and the XCCDF returns a result of "True", the tool is conveying the fact that the system was found to be compliant with that check and therefore returns a "Pass" result. A similar definition for a vulnerable condition will return results of "False" if that vulnerability was not found on the examined devices, resulting in a "Pass" from the XCCDF rule. SCAP compliant processors that generate XCCDF rule results SHALL apply the mapping illustrated in Table 4-2 when deriving XCCDF rule results from OVAL definition results.

SCAP users may reference several classes of OVAL Definitions from a single XCCDF document (e.g., a single SCAP-expressed checklist that performs configuration verification AND patch compliance checks.) Users of multiple OVAL Definition classes must consider the effect of the definition class when interpreting definition results and ensure that rule evaluation produces the correct results.

Table 4-2. Deriving XCCDF Rule Results from OVAL Definition Results

OVAL Definition Result		XCCDF Rule Result
Error		Error
Unknown		Unknown
Not applicable		Notapplicable
Not evaluated		Notchecked
Definition Class	**Definition Result**	Pass
Compliance	True	
Vulnerability	False	
Inventory	True	
Patch	False	
Definition Class	**Definition Result**	Fail
Compliance	False	
Vulnerability	True	
Inventory	False	
Patch	True	

4.2 OVAL Conventions and Requirements

When used for SCAP purposes, OVAL content SHALL comply with one of the following document schema:

- `<oval-def:oval_definitions>` document – A specification of OVAL Definitions, Tests, Objects, States and Variables. This document may optionally be used as a component of an SCAP data source.
- `<oval-var:oval_variables>` document – A specification of external OVAL Variable bindings. This document may optionally be used as a component of an SCAP data source.
- `<oval-sys:oval_system_characteristics>` document – A specification of target system characteristics, that is, the specification of OVAL Object values queried from a target system
- `<oval-res:oval_results>` document – The evaluation results of specified definitions and tests, as well as a copy of the OVAL System Characteristics from which the results can be derived.

4.2.1 OVAL Schema Specification

For the purposes of SCAP, OVAL content SHALL validate against OVAL schema bundle Version 5.3 or 5.4. In support of OVAL upward compatibility, content that validates against a Version 5.3 schema bundle SHALL validate against a Version 5.4 schema bundle.

All of the OVAL content SHALL contain an `<oval:generator>` element. The bundle version of any particular document instance SHALL be specified using the `<oval:schema_version>` content element of the `<oval:generator>` as in this example:

```
<oval:generator>
  <oval:product_name>The OVAL Repository</oval:product_name>
  <oval:schema_version>5.3</oval:schema_version>
</oval:generator>
```

The bundle version of an `<oval-def:oval_definitions>` document SHOULD be chosen as 5.3 if the content validates against the Version 5.3 schema bundle.

The bundle version of an `<oval-var:oval_variables>` document SHALL be the same as that of the `<ovaldef:oval_definitions>` document whose external variables are bound by the variables document.

If an `<oval-sc:oval_system_characteristics>` or `<oval-res:oval_results>` document is generated as a consequence of the application of a `<oval-def:oval_definitions>` *document*, then the bundle version of the generated document SHALL be the same as that of the `<oval-def:oval_definitions>` *document.*

4.2.2 OVAL Definitions and Affected Platforms

The `<oval-def:metadata>` element of an `<oval-def:definition>` optionally identifies platforms affected by including `<oval-def:affected>` elements. One or more of these elements SHALL be present whenever the class of the `<oval-def:definition>` is "vulnerability", "compliance", "patch", or "inventory".[22] `<oval-def:affected>` elements MAY be used when the definition class is "miscellaneous". If more than one `<oval-def:affected>` elements is included in definition metadata, then the family attribute of each of the `<oval-def:affected>` elements SHALL be bound to the same value[23]. Thus each `<oval-def:definition>` is either associated with a single family, or the family association of the definition is undefined (only allowed for OVAL definitions whose class is "miscellaneous").

If the family association of an OVAL Definition is undefined, any definitions extended by that definition SHALL also have undefined family associations. If the family association of an OVAL Definition is specified, then any definitions extended by that definition SHALL be the same as that of the extending definition or SHALL have an undefined family association.

An OVAL Definition's family association also determines the kinds of tests that it can reference as `<oval-def:criterion>`. Table 4-3 maps the family associations to the test component schema[24] allowed for the family. Each component namespace is designated by its fractional part; for example, #windows refers to the component namespace URI http://oval.mitre.org/XMLSchema/oval-definitions-5#windows.

Table 4-3. Association of Family to Component Schemas

Family	Allowed Subschemas
Undefined	#independent
ios	#independent #ios
Macos	#independent #macos
unix	#independent #hpux #linux #solaris #unix
windows	#independent #windows

[22] The OVAL Definition Schema is available at http://oval.mitre.org/language/download/schema/version5.3/ovaldefinition/documentation/oval-definitions-schema.pdf

[23] The supported family values are "ios", "macos", "unix" and "windows". (http://oval.mitre.org/language/download/schema/version5.3/ovaldefinition/documentation/oval-common-schema.pdf).

[24] The OVAL 5.3 test subschema namespaces are:
- http://oval.mitre.org/XMLSchema/oval-definitions-5#independent -- Supports any OS.
- http://oval.mitre.org/XMLSchema/oval-definitions-5#ios – Supports Cisco IOS
- http://oval.mitre.org/XMLSchema/oval-definitions-5#hpux – Supports HP-UX
- http://oval.mitre.org/XMLSchema/oval-definitions-5#unix – Supports Unix dialects
- http://oval.mitre.org/XMLSchema/oval-definitions-5#linux – Supports Linux
- http://oval.mitre.org/XMLSchema/oval-definitions-5#macos – Supports Apple Macintosh
- http://oval.mitre.org/XMLSchema/oval-definitions-5#solaris – Supports Sun Solaris
- http://oval.mitre.org/XMLSchema/oval-definitions-5#windows – Supports Microsoft Windows
- http://oval.mitre.org/XMLSchema/oval-definitions-5#freebsd – Supports FreeBSD
- http://oval.mitre.org/XMLSchema/oval-definitions-5#apache – Supports Apache applications

Refer to http://oval.mitre.org/language/download/schema/version5.3/index.html.

4.2.3 OVAL Definitions and Compliance Validation

An OVAL compliance Definition is an `<oval-def:oval_definitions>` document that specifies definitions for validating the compliance status of target platforms. An OVAL compliance definition SHALL specify at least one definition of class "compliance." An OVAL compliance definition may also reference definitions of class "inventory" that are extended (transitive) by the "compliance" class definitions.

If an OVAL "compliance" class definition maps to one or more CCE identifiers, the definition SHOULD include `<oval-def:reference>` elements that reference those identifiers using the following format:

<oval-def:reference source="CCE" ref_id="*CCE_identifier*"/>

4.2.4 OVAL Definitions and Vulnerability Assessment

An OVAL vulnerability definition is an `<oval-def:oval_definitions>` document that specifies definitions for assessing the vulnerability status of target platforms. An OVAL vulnerability definition SHALL specify at least one definition of class "vulnerability" or "patch". An OVAL vulnerability definition may also reference definitions of class "inventory" or "compliance" that are extended (transitive) by the "vulnerability" class definitions.

If an OVAL "patch" or "vulnerability" class definition maps to one or more CVE identifiers, the definition SHOULD include `<oval-def:reference>` elements that reference those identifiers using the following format:

<oval-def:reference source="CVE" ref_id="*CVE_identifier*"/>

OVAL "patch" class definitions SHOULD also reference source patch identifiers, if they exist.

4.2.5 OVAL Definitions and Patch Assessment

An OVAL patch definition is an `<oval-def:oval_definitions>` document that specifies definitions for assessing the patch status of target platforms. An OVAL patch definition SHALL specify at least one definition of class "patch". An OVAL patch definition may also include definitions of class "inventory" that are extended (transitive) by the "patch" class definitions.

If an OVAL "patch" class definition is associated with a source specific identifier (for example KB numbers for Microsoft patches) these identifiers SHOULD be included in `<oval-def:reference>` elements contained by the definition. For example:

<oval-def:reference source="*www.microsoft.com/Patch*" ref_id="*KB912919*"/>

If an OVAL "patch" class definition maps to one or more CVE identifiers, the definition SHOULD include `<oval-def:reference>` elements that reference those identifiers using the following format:

<oval-def:reference source="CVE" ref_id="*CVE_identifier*"/>

4.2.6 OVAL Inventories

An OVAL Inventory component is an `<oval-def:oval_definitions>` document that specifies only "inventory" class definitions for verifying CPE match conditions.

4.2.7 OVAL Results

While the OVAL specification permits limited result status reporting, SCAP-compliant content includes full status reporting including Error, Unknown, Not Applicable, Not Evaluated, True, and False. Section 4.1.10 provides additional detail about OVAL results.

Results returned SHALL be compliant with the OVAL results schema[25]. In order to support SCAP instances where OVAL thin content (only the ID of the definition and the results) is preferred, SCAP content SHALL support all valid values for the ContentEnumeration directives controlling the expected content of the results file. Specific product requirements may be implemented through the Derived Test Requirements.

4.3 CPE Conventions

CPE Names supported by the Official CPE Dictionary data feed or the custodial list supported by The MITRE Corporation[26] may be used by SCAP components to reference CPE Names. The process for assigning new CPE Names is supported by The MITRE Corporation.[27] Local enumerations are permitted, but if a CPE Name for a product or platform exists in the Official CPE Dictionary, the tool SHALL use that official identifier.

Section 8 of CPE Specification 2.2 provides the defining structure of the Official CPE Dictionary. For certain names, a `<cpe_dict:cpe-item>` may contain one or more checks of the form a `<check>` element that references OVAL system inventory definitions using the following format:

```
<cpe_dict:check system="http://oval.mitre.org/XMLSchema/oval-definitions-5"
      href="Oval_URL">Oval_inventory_definition_id</cpe_dict:check>
```

For example:

```
<cpe-list xmlns="http://cpe.mitre.org/dictionary/2.0"
          xmlns:cpe_dict="http://cpe.mitre.org/dictionary/2.0">

     <cpe-item name="cpe:/o:microsoft:windows_2003">
          <title>Microsoft Windows Server 2003</title>
          <check system="http://oval.mitre.org/XMLSchema/oval-definitions-5">
                            oval:org.mitre.oval:def:128
          </check>
      </cpe-item>
</cpe-list>
```

The referenced OVAL inventory definition specifies the technical procedure for determining whether or not a specific target asset is an instance of the CPE Name specified by the `<cpe_dict:cpe-item>` element. This usage is encouraged for CPE dictionary components of SCAP expressed data streams.

[25] The OVAL schemas are described in detail at http://oval.mitre.org/language/about
[26] The Official CPE Dictionary is located at http://nvd.nist.gov/cpe.cfm
[27] Ibid.

If a `<cpe_dict:cpe-item>` contained in a CPE dictionary component of an SCAP data stream references an OVAL "inventory" definition, then that definition SHALL be resolved by a CPE Inventory component in the same data stream[28]. Furthermore the title of the `<cpe_dict:cpe-item>` SHALL match the title of an affected platform bound to the referenced definition[29].

4.4 CCE Conventions

CCE identifiers are used by SCAP components to reference Common Configuration Enumerations. CCE identifiers for new configuration settings are assigned by the CCE Content Team.[30] To maintain consistency and accuracy among the SCAP validated tools, if a CCE entry for a particular configuration setting exists in the Official CCE Dictionary, the tool SHALL use the official CCE identifier.

The NVD provides a data feed that correlates the CCE identifiers with the *Recommended Security Controls for Federal Information Systems and Organizations* as described in Special Publication 800-53.

4.5 CVE Conventions

CVE identifiers supported by the NVD CVE data feed may be used by SCAP components to reference CVEs.[31] CVE references in other SCAP content may include both "candidate" and "entry" status identifiers. The process for submitting unpublished vulnerabilities and obtain CVE identifiers is available from MITRE via http://cve.mitre.org/cve/obtain_id.html.

It should be noted that not all CVE entries identify an associated patch or remediation; in fact, the ability to determine the availability of a patch or remediation is a valuable feature of the CVE component. Vendors are encouraged to reference CVE entries in notifications (e.g., security patch bulletins) to support the use of automated tools and to ensure clarity when referencing a given vulnerability. Similarly, CVE authors are encouraged to reference applicable vendor patch identification whenever possible.

NIST provides a CVE data feed to support dynamic and current vulnerability information and associated metadata (e.g., CVSS values). The current schema is available at http://nvd.nist.gov/download.cfm.

4.6 CVSS Conventions

SCAP CVSS scoring vectors MAY be bound to CVE identifiers. If there is an accompanying CVSS score for a CVE, products SHOULD use it. CVSS score data is provided by the NVD CVE data feed, which maps CVE to CVSS scoring vectors for all CVE identifiers. If a CVSS Base Metric is provided, it SHALL reflect the current Base score as reflected in the official source.

The CVSS specification (described at http://www.first.org/cvss/cvss-guide.html) supports Base score metrics that characterize the severity of the vulnerability using the intrinsic characteristics of the vulnerability. CVSS also allows weighting of the base score using Temporal Metrics (e.g., Exploitability, Report Confidence) and Environmental Metrics (e.g., Collateral Damage Potential, Target Distribution). SCAP users are encouraged to leverage the flexibility provided within the CVSS component by using the Temporal and Environmental factors when applicable.

[28] More information is provided in section 4.7.
[29] Section 4.2.2 explains more detail about OVAL definitions.
[30] http://cce.mitre.org/lists/creation_process.html documents the CCE Creation Process.
[31] NIST provides the NVD CVE data feed at http://nvd.nist.gov/download.cfm - CVE_FEED

4.7 SCAP Data Sources

An SCAP data source is the expression of one or more SCAP components that can be processed by an SCAP-validated product. The specific logical composition of an SCAP data source depends on the use case. All SCAP data sources include one or more component XML documents that are authored to support a specific SCAP use case and adapted to address the specific policy or metric intentions of the use case.

Table 4-4 identifies these XML sources along with their naming conventions.

Table 4-4. SCAP XML Data Sources

Component	Document Root Element	Stream Locator
XCCDF Benchmark	<xccdf:Benchmark>	*xxxx*xccdf.xml
OVAL Compliance	<ovaldef:oval_definitions>	*xxxx*compliance.xml
OVAL Patch	<ovaldef:oval_definitions>	*xxxx*patches.xml
OVAL Vulnerability	<ovaldef:oval_definitions>	*xxxx*vulnerabilty.xml
OVAL Inventory	<ovaldef:oval_definitions>	*xxxxinventory*.xml
CPE Dictionary	<cpe_dict:cpe-list>	*xxxx*cpe-oval.xml
CPE Inventory	<ovaldef:oval_definitions>	*xxxx*cpe-dictionary.xml

The Stream Locator is a relative stream URL whose base is the URL of the deployed data source. The notation '*xxxx*' designates a locator prefix that is associated with the data source that contains the component stream.

Where possible, SCAP data feeds are provided to ensure updated and accurate definitions and mappings. When a valid data feed exists, such as the CCE data feed at the NVD website, this data source is the preferred method for SCAP component reference items.

5. SCAP Use-Case Requirements

To facilitate implementation of the SCAP requirements specified in Section 4, this section describes specific uses that demonstrate effective use of the protocol. The content of this section identifies the input data source conventions identified with the SCAP components and associates these with the following use case examples:

- Configuration Verification
- Vulnerability Assessment
- Patch Validation
- Inventory Collection

These examples are not intended to limit SCAP but provide a framework for future use cases and document the specifics of the data streams described.

SCAP enables many types of automated assessment, each with discrete benefits and each considered separate content. For example, vulnerability assessment (i.e. quantitative and repeatable measurement and scoring of software flaw vulnerabilities across systems) is related to but separate from configuration verification.

5.1 SCAP Configuration Verification with XCCDF and OVAL

SCAP enables automated processes to compare system characteristics and settings against an SCAP-expressed checklist. Using such a process, such as that referenced in Special Publication 800-68, *Guide to Securing Microsoft Windows XP Systems for IT Professionals*, a user may confirm compliance and identify deviations from checklists appropriate for relevant operating systems and/or applications.

The following data sources are necessary to support SCAP-compliant configuration verification use cases:

Table 5-1. SCAP Configuration Verification Data Sources

Component	Stream Locator	Required/Optional
XCCDF Benchmark	xxxxxccdf.xml	Required
OVAL Compliance	xxxxoval.xml	Required
OVAL Patch	xxxxpatches.xml	Optional
CPE Dictionary	xxxxcpe-dictionary.xml	Required
CPE Inventory	xxxxcpe-oval.xml	Required

For an SCAP configuration verification data source to be processed by the appropriate SCAP-validated product:

- Each Rule specified in the XCCDF benchmark SHALL include an `<ident>` element containing a CCE reference, where an appropriate reference exists.
- If an `<ident>` is specified in an XCCDF benchmark Rule, then that reference SHALL match the CCE reference found in the associated OVAL definition(s).
- The XCCDF `<xccdf:Benchmark>` element SHALL contain references to one or more CPEs.

- A rule bound to a specific CCE may be associated with one or more of the controls described in Special Publication 800-53. The mapping to these controls may be represented in the XCCDF benchmark by applying the following conventions:
  - Use of an official, dynamic data feed is preferred to static coding of values in SCAP data sources. The NVD provides a data feed[32] that correlates CCE identifiers with the control identifiers described in NIST Special Publication 800-53.
  - The 800-53 controls referenced within an XCCDF benchmark are represented by `<xccdf:Group>` elements. Note that the XCCDF control group identifiers correspond to the control identifiers found in Appendix F of Special Publication 800-53.
  - Each `<xccdf:Rule>` SHALL be associated with one or more 800-53 controls by capturing the ID of the each associated control in an `<xccdf:requires>` element within the rule. This association allows rules to be enabled/disabled at the profile level through the selection/deselection of the associated control groups.
  - The chosen convention for mapping XCCDF rules to 800-53 controls within a benchmark SHOULD be uniformly applied to all rules having 800-53 mappings.
- XCCDF configuration scanning processes SHALL produce XCCDF Results and OVAL Results that comply with the XCCDF and OVAL Results schema.
- XCCDF Results documents SHALL include a result for each rule that was evaluated during the scan. OVAL Results documents SHALL include the results of every OVAL definition used to generate the reported rule results.
- If an XCCDF rule references a specific OVAL definition, the definition MUST be a compliance class definition.
- An XCCDF benchmark MAY include a "patches up-to-date" rule that references an OVAL patch component stream. If such a rule is used, the OVAL patch component MUST be included in the OVAL compliance data source.
- An XCCDF benchmark MAY enumerate one patch per rule. If this approach is used, a specific OVAL definition of class "patch" MUST be referenced in the OVAL Patch component stream.

5.2 SCAP Vulnerability Assessment

In the context of SCAP, a vulnerability is defined as a software flaw, bug, or defect that introduces a security exposure. SCAP enables interoperability among vulnerability scanners and reporting tools to provide consistent detection and reporting of these flaws and supports comprehensive remediation tool capabilities. Section 4.1.10 documents the reasons that vulnerability assessment and configuration verification are significantly different use cases, and shows how SCAP results are interpreted under these two use cases.

5.2.1 SCAP Vulnerability Assessment Using XCCDF and OVAL

Effective vulnerability assessment using a combination of SCAP components requires the following data sources:

[32] The link to the CCE data feed will be entered here.

Table 5-2. SCAP Vulnerability Assessment Data Sources

Component	Stream Locator	Required/Optional
XCCDF Benchmark	xxxxxccdf.xml	Required
OVAL Vulnerability	xxxxoval.xml	Required
OVAL Patch	xxxxpatches.xml	Optional
CPE Dictionary	xxxxcpe-dictionary.xml	Required
CPE Inventory	xxxxcpe-oval.xml	Required

For an SCAP Vulnerability Assessment to be performed by the appropriate SCAP-validated product, the following conditions SHALL be met:

- The XCCDF `<xccdf:Benchmark>` element SHALL contain references to one or more CPEs.
- XCCDF Vulnerability Scanning SHALL generate an XCCDF Results file. The XCCDF Results document SHALL include a result for each rule that was evaluated during the scan.
- Each Rule specified in an XCCDF benchmark SHALL include an `<ident>` element containing a CVE reference, where an appropriate reference exists.
- Each Rule specified in an XCCDF benchmark SHALL reference a specific OVAL vulnerability, patch, or inventory definition; except in cases where no automated mechanism exists to express a check in OVAL
- If OVAL Results are generated:
  - OVAL Results SHALL be expressed in compliance with the OVAL Results schema and,
  - OVAL Results documents SHALL include the results of every OVAL definition used to generate the reported rule results.
- If a CVE reference is specified in an XCCDF benchmark rule, then that reference SHALL match the CVE reference found in the associated OVAL definition(s).

5.2.2 SCAP Vulnerability Assessment Using Standalone OVAL

For an OVAL-only vulnerability assessment to be processed by the appropriate SCAP-validated product, the following SHALL be present:

- A Standalone OVAL Vulnerability Data Stream SHALL include an OVAL Vulnerability XML stream component that defines the applied OVAL vulnerability class definitions.
- OVAL Definitions SHALL include CVE references, if such exist.
- OVAL vulnerability data scanning SHALL generate an OVAL Results document that complies with the OVAL Results schema and includes the results of every OVAL definition used to generate the reported rule results.
- The OVAL Results document SHALL include a definition result with supporting system-characteristics data for every definition in the vulnerability data source.

5.3 Inventory Collection

Organizations require a consistent protocol for integrating inventory information from among a broad range of products, and SCAP provides excellent methods for collecting this data. For example, SCAP inventory data is an important input to the Risk Management Framework,[33] establishing an effective foundation for system categorization and baseline security controls. For SCAP tools to collect this inventory information, the following data sources are required:

Table 5-3. SCAP Inventory Collection

Component	Stream Locator	Required/Optional
XCCDF Benchmark	xxxxxccdf.xml	Optional
CPE Dictionary	xxxxcpe-dictionary.xml	Required
CPE Inventory	xxxxcpe-oval.xml	Required

In order for an Inventory scan to be processed by the appropriate SCAP-validated product:

- The inventory data source SHALL include an OVAL Inventory component that defines the applied OVAL inventory class definitions.
- OVAL vulnerability data scanning SHALL generate an OVAL Results document that complies with the OVAL Results schema and includes the results of every OVAL definition used to generate the reported rule results.
- The results document SHALL include a definition result with supporting system-characteristics data for every definition in the Inventory component.

[33] The Risk Management Framework is explained in NIST Special Publication 800-53 Revision 3 at http://csrc.nist.gov/publications/nistpubs/800-53-Rev1/800-53-rev1-final-clean-sz.pdf

Appendix A— Acronyms and Abbreviations

Appendix A defines selected acronyms and abbreviations used in the document.

CCE	Common Configuration Enumeration
CJCSM	Chairman of the Joint Chiefs of Staff Manual
CNA	Candidate Naming Authority
CPE	Common Platform Enumeration
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
DODI	Department of Defense Instruction
DHS	Department of Homeland Security
DISA	Defense Information Systems Agency
DODD	Department of Defense Directive
EVR	Epoch/Version/Release
FDCC	Federal Desktop Core Configuration
FIRST	Forum of Incident Response and Security Teams
FISMA	Federal Information Security Management Act
FSO	DISA Field Security Operations
GPO	Active Directory Group Policy Object
HIPAA	Health Insurance Portability and Accountability Act
IA	Information Assurance
IOS	CISCO Internetwork Operating System
IT	Information Technology
NCSD	National Cyber Security Division
NIST	National Institute of Standards and Technology
NISTIR	NIST Interagency Report
NSA	National Security Agency
NVD	National Vulnerability Database
OMB	Office of Management and Budget
OS	Operating System
OVAL	Open Vulnerability and Assessment Language
PDI	DISA Potential Discrepancy Item
RFC	Request for Comments
SCAP	Security Content Automation Protocol
SIG	Special Interest Group
SP	Service Pack
SP	Special Publication
STIG	Security Technical Implementation Guide

URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VMS	DISA Vulnerability Management System
XCCDF	eXtensible Configuration Checklist Description Format
XML	eXtensible Markup Language

Appendix B—References and other Resources

Appendix B lists references and other resources related to SCAP 1.0 and its component specifications.

[BUT09] Buttner, A. and Ziring, N., "Common Platform Enumeration (CPE)—Specification, Version 2.2", MITRE Corporation, March 11, 2009. http://cpe.mitre.org/files/cpe-specification_2.2.pdf

[QUI08] Quinn, S. and Ziring, N., "Specification for the Extensible Configuration Checklist Description Format (XCCDF) Version 1.1.4", January 2008, http://csrc.nist.gov/publications/nistir/ir7275r3/NISTIR-7275r3.pdf

[QUIN08] Quinn, S., Scarfone, K., Souppaya, M., "National Checklist Program for IT Products—Guidelines for Checklist Users and Developers", September 2008, http://csrc.nist.gov/publications/drafts/800-70-rev1/Draft-SP800-70-r1.pdf

[BAR09] Barrett, M., Johnson, C., Mell, P., Quinn, S., Scarfone, K., "Guide to Adopting and Using the Security Content Automation Protocol (SCAP)", May 2009, http://csrc.nist.gov/publications/drafts/800-117/draft-sp800-117.pdf

[MEL07] Mell, P., Scarfone, K., and Romanosky, S., "A Complete Guide to the Common Vulnerability Scoring System Version 2.0", FIRST, July 2007. http://www.first.org/cvss/cvss-guide.html

The resources below may be retrieved from the NIST SCAP web site:

[1] SCAP Schema Bundle – A collection of all schema required to process SCAP data feeds and Expressed Streams (http://scap.nist.gov/scap1.0/resources/scap-schema-bundle.zip)

[2] SCAP Sample Content – A collection of sample SCAP data feed and expression content that complies with the provisions of this specification. (http://scap.nist.gov/scap1.0/resources/scap-sample-bundle.zip)

[3] CVE Specification and description (http://scap.nist.gov/revision/1.0/index.html#cve)

[4] CCE Specification and description (http://scap.nist.gov/revision/1.0/index.html#cce)

[5] CPE Specification and description (http://scap.nist.gov/revision/1.0/index.html#cpe)

[5] CVSS Specification and description (http://scap.nist.gov/revision/1.0/index.html#cvss)

[6] XCCDF Specification and description (http://scap.nist.gov/revision/1.0/index.html#xccdf)

[7] OVAL Specification and description (http://scap.nist.gov/revision/1.0/index.html#oval)

[8] NIST SP 800-53 Control Group Definition (http://scap.nist.gov/scap1.0/resources/800-53-controls.xml)

Appendix C—SCAP Extensions to the XCCDF Specification

C.1 Rule and Group Selection

Rules and Groups may be selected for application in the context of either a Benchmark or a Profile contained by a Benchmark. This extension expands on the semantics of rule and group Selection.

C.1.1. A group or rule is selected in a Benchmark if and only if at least one of the following is true:

a. The group or rule is immediately contained by the Benchmark and the 'selected' attribute of the group or rule is bound to true.

b. The group or rule is immediately contained by a group that is selected in the Benchmark and the 'selected' attribute of the subject group or rule is bound to true.

c. The group or rule is selected by association relative[34] to the Benchmark.

C.1.2. A group or rule will be selected in a Profile only if it is either explicitly or implicitly selected in the Profile.

C.1.3. A group or rule is explicitly selected in a Profile only if there exists a `<select>` contained by the Profile whose 'idref' attribute is bound to the id of the group or rule and whose 'selected' attribute is set to true. A group or rule is explicitly deselected in a Profile only if there exists a `<select>` contained by the Profile whose 'idref' attribute is bound to the id of the group or rule and whose 'selected' attribute is set to false.

C.1.4. A group or rule is implicitly selected in a Profile if and only if it is not explicitly selected or deselected in the Profile and at least one of the following is true:

a. The group or rule is selected in the Benchmark.

b. The group or rule is selected in the Profile extended by the subject Profile.

c. The group or rule is selected by association relative to the Profile.

C.2 Selection by Association

The sequence of `<requires>` and `<conflicts>` optionally bound to a group or rule creates a set of directed associations rooted on the subject group or rule and terminating on other Groups or Rules. The resulting directed graphs are valid only if they are acyclic. The associations rooted on a group or rule may be used to determine an associative selection predicate on the subject group or rule. The selection predicate is evaluated as follows

C.2.1. The selection value of an `<requires>` relative to the Benchmark is true only if all of the Groups or Rules referenced by the element are selected by the Benchmark, likewise the selection value relative to a Profile is true only if all of the Groups or Rules referenced by the element are selected in the Profile.

[34] Selection by association is discussed in C.2.

C.2.2. The selection value of an `<conflicts>` relative to the Benchmark is true only if at least one of the Groups or Rules referenced by the element are deselected by the Benchmark, likewise the selection value relative to a Profile is true only if at least one of the Groups or Rules referenced by the element are deselected in the Profile.

C.2.3. The selection value of a sequence of <requires> and <conflicts> is true relative to the Benchmark only if at least one of the elements in the sequence evaluates to true relative to the Benchmark, likewise the value of the sequence relative to a Profile is true only if at least one of the elements in the sequence evaluates to true relative to the Profile.