NIST Special Publication 800
NIST SP 800-126r4 ipd

Technical Specification for the Security Content Automation Protocol (SCAP)

SCAP Version 1.4

Initial Public Draft

Dragos Prisaca
Stephen D. Quinn
Jack Vander Pol
Daniel K. Harris

This publication is available free of charge from:
https://doi.org/10.6028/NIST.SP.800-126r4.ipd

NATIONAL INSTITUTE OF
STANDARDS AND TECHNOLOGY
U.S. DEPARTMENT OF COMMERCE

NIST Special Publication 800
NIST SP 800-126r4 ipd

Technical Specification for the Security Content Automation Protocol (SCAP)

SCAP Version 1.4

Initial Public Draft

Dragos Prisaca
Stephen D. Quinn
Computer Security Division
Information Technology Laboratory

Jack Vander Pol
Daniel K. Harris
Naval Information Warfare Center (NIWC) Atlantic

This publication is available free of charge from:
https://doi.org/10.6028/NIST.SP.800-126r4.ipd

December 2025

U.S. Department of Commerce
Howard Lutnick, Secretary of Commerce

National Institute of Standards and Technology
Craig Burkhardt, Acting Under Secretary of Commerce for Standards and Technology and Acting NIST Director

Publication History
Approved by the NIST Editorial Review Board on YYYY-MM-DD [Will be added to final publication.]
Supersedes NIST Series XXX (Month Year) DOI [Will be added to final publication, if applicable.]

How to Cite this NIST Technical Series Publication
Prisaca D, Quinn SD, Vander Pol J, Harris DK (2025) Technical Specification for the Security Content Automation Protocol (SCAP): SCAP Version 1.4. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) NIST SP 800-126r4 ipd. https://doi.org/10.6028/NIST.SP.800-126r4.ipd

Author ORCID iDs
Dragos Prisaca: 0009-0007-7361-8433
Stephen D. Quinn: 0000-0003-1436-684X

Public Comment Period
December 11, 2025 – February 11, 2026

Submit Comments
scap@nist.gov

National Institute of Standards and Technology
Attn: Computer Security Division, Information Technology Laboratory
100 Bureau Drive (Mail Stop 8930) Gaithersburg, MD 20899-8930

Additional Information
Additional information about this publication is available at https://csrc.nist.gov/pubs/sp/800/126/r4/ipd, including related content, potential updates, and document history.

All comments are subject to release under the Freedom of Information Act (FOIA).

Abstract

The Security Content Automation Protocol (SCAP) is a suite of specifications that standardize the format and nomenclature by which software flaw and security configuration information is communicated, both to machines and humans. This publication, along with its annex (NIST Special Publication 800-126Ar1) and a set of schemas, collectively define the technical composition of SCAP version 1.4 in terms of its component specifications, their interrelationships and interoperation, and the requirements for SCAP content.

Keywords

checklists; patch verification; security automation; security checklists; security configuration; Security Content Automation Protocol (SCAP); software flaws; vulnerabilities.

Reports on Computer Systems Technology

Call for Patent Claims

This public review includes a call for information on essential patent claims (claims whose use would be required for compliance with the guidance or requirements in this Information Technology Laboratory (ITL) draft publication). Such guidance and/or requirements may be directly stated in this ITL Publication or by reference to another publication. This call also includes disclosure, where known, of the existence of pending U.S. or foreign patent applications relating to this ITL draft publication and of any relevant unexpired U.S. or foreign patents.

ITL may require from the patent holder, or a party authorized to make assurances on its behalf, in written or electronic form, either:

a) assurance in the form of a general disclaimer to the effect that such party does not hold and does not currently intend holding any essential patent claim(s); or

b) assurance that a license to such essential patent claim(s) will be made available to applicants desiring to utilize the license for the purpose of complying with the guidance or requirements in this ITL draft publication either:

   i. under reasonable terms and conditions that are demonstrably free of any unfair discrimination; or

   ii. without compensation and under reasonable terms and conditions that are demonstrably free of any unfair discrimination.

Such assurance shall indicate that the patent holder (or third party authorized to make assurances on its behalf) will include in any documents transferring ownership of patents subject to the assurance, provisions sufficient to ensure that the commitments in the assurance are binding on the transferee, and that the transferee will similarly include appropriate provisions in the event of future transfers with the goal of binding each successor-in-interest.

The assurance shall also indicate that it is intended to be binding on successors-in-interest regardless of whether such provisions are included in the relevant transfer documents.

Such statements should be addressed to: scap@nist.gov

Table of Contents

List of Tables

List of Figures

Executive Summary

The Security Content Automation Protocol (SCAP) is a suite of specifications that standardize the format and nomenclature by which security configuration information is communicated to both machines and humans.[1] SCAP is a multi-purpose framework of specifications that support automated configuration, vulnerability and patch checking, technical and managerial control compliance activities, and security measurement. Goals for the development of SCAP include standardizing system security management, promoting the interoperability of security products, and fostering the use of standard expressions of security content.

Security configuration in SCAP format is curated and managed by the NIST National Checklist Program (NCP), which is described in NIST Special Publication (SP) 800-70r5 (Revision 5). This document, its annex [SP800-126Ar1], and a set of schemas collectively define the technical composition of SCAP version 1.4 in terms of its component specifications and requirements. The technical specification for SCAP describes the conventions for ensuring the consistent and accurate exchange of SCAP-conformant content and the ability to reliably use the content with SCAP-conformant products.

Organizations that develop SCAP 1.4-based content or products should adhere to the following recommendations:

- *Follow the requirements listed in this document, its annex, and the associated component specifications and set of schemas***.**

  Organizations should ensure that their implementation and use of SCAP 1.4 complies with the requirements detailed in each component specification, this document, its annex, and the set of schemas.

  If requirements conflict between component specifications, this document will provide clarification. If a component specification conflicts with this document, the requirements in this document take precedence. If a component specification or this document conflicts with the annex, the requirements in the annex take precedence. If a specification and a schema conflict, the requirements in the specification take precedence.

- *When creating SCAP content, adhere to the conventions specified in this document and its annex.*

  Security products and checklist authors assemble content from SCAP data repositories to create SCAP-conformant security guidance. Organizations that produce SCAP content to be shared between tools should adhere to the conventions described in this specification to ensure the highest degree of interoperability.

[1] Products that implement SCAP can also be used to support non-security use cases, such as configuration management, software inventory, and malware hunting.

1. Introduction

1.1. Purpose and Scope

This document, its annex [SP800-126A], and a set of schemas collectively provide the technical specification for version 1.4 of the Security Content Automation Protocol (SCAP). SCAP (pronounced /EHS-kap/) consists of a suite of specifications for standardizing the format and nomenclature by which software flaw and security configuration information is communicated to both machines and humans. This document defines requirements for creating and processing SCAP source content that build on the requirements defined within the individual SCAP component specifications. Each new requirement pertains to either using multiple component specifications together or further constraining one of the individual component specifications.[2]

To extend the contents of this document, [SP800-126A] has been created as an annex to specify additional entities that may be used in SCAP 1.4-conformant content creation and processing. It provides:

- Particular minor version updates to SCAP 1.4 component specifications
- Particular Open Vulnerability and Assessment Language (OVAL) platform schema versions

The scope of this document and its annex is limited to SCAP version 1.4. Other versions of SCAP and its component specifications are not addressed in these documents.

Future versions of SCAP will be defined in distinct revisions of this document and its annex, each clearly labeled with a document revision number and the appropriate SCAP version number.

1.2. Audience

This document is intended for:

- Content authors and editors who seek to ensure that the SCAP source content they produce operates correctly, consistently, and reliably in SCAP products
- Software developers and system integrators who seek to create, use, or exchange SCAP content in their products or service offerings

1.3. Document Structure

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

- Section 2 provides the high-level requirements for claiming conformance with the SCAP 1.4 specification.

[2] Refer to the individual component specifications to see their requirements.

- Section 3 details the requirements and recommendations for SCAP content syntax, structure, and development.
- Section 4 defines SCAP content processing requirements and recommendations.
- Section 5 provides additional content requirements and recommendations for particular use cases.
- Appendix A gives an overview of major security considerations for SCAP implementation.
- Appendix B provides an acronym and abbreviation list.
- Appendix C provides a glossary of selected terms used in this document.
- Appendix D lists references and other resources related to SCAP 1.4.
- Appendix E provides a log of significant changes to previous versions of this specification.

1.4. Document Conventions

In this document, the capitalized terms “MUST,” “MUST NOT,” “REQUIRED,” “SHALL,” “SHALL NOT,” “SHOULD,” “SHOULD NOT,” “RECOMMENDED,” “MAY,” and “OPTIONAL” are to be interpreted as described in Request for Comment (RFC) 2119 [RFC2119]. When these words appear in lowercase (e.g., “should,” “may”), they are not intended to be interpreted as RFC 2119 key words.

Italicized single terms are defined and listed in Appendix C.

Some of the requirements and conventions used in this document reference Extensible Markup Language (XML) content [XMLS]. These references come in two forms: inline and indented.

An example of an inline reference is:

> A *<cpe2_dict:cpe-item>* may contain *<cpe2_dict:check>* elements that reference OVAL definitions.

In this example, the notation *<cpe2_dict:cpe-item>* can be replaced by a more verbose equivalent: “The XML element whose qualified name is *cpe2_dict:cpe-item*.”

An example of an indented reference is:

> References to OVAL definitions are expressed using the following format:

```
<cpe2_dict:check system=
   "http://oval.mitre.org/XMLSchema/oval-definitions-5"
   href="Oval_URL">[Oval_inventory_definition_id]
</cpe2_dict:check>
```

When describing XML attributes within this document, the general convention is to reference the attribute and its associated element, including the namespace alias:

@attributeName for the *<prefix:localName>*

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 (i.e., freely replaceable) content by the use of *italics*. 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 XML model, and the XML schema is a definition and implementation of that model. 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 varies from document to document. In this specification, the conventional mappings listed in Table 1 are used.

Table 1. Conventional XML mappings

Prefix	Namespace	Schema
ai	http://scap.nist.gov/schema/asset-identification/1.1	Asset Identification
arf	http://scap.nist.gov/schema/asset-reporting-format/1.1	ARF
arf-rel	http://scap.nist.gov/specifications/arf/vocabulary/relationships/1.0#	ARF relationships
cat	urn:oasis:names:tc:entity:xmlns:xml:catalog	XML Catalog
con	http://scap.nist.gov/schema/scap/constructs/1.4	SCAP Constructs
cpe-dict-ext	http://scap.nist.gov/schema/cpe-extension/2.3	CPE Dictionary 2.3 schema extension
cpe2	http://cpe.mitre.org/language/2.0	Embedded CPE references
cpe2-dict	http://cpe.mitre.org/dictionary/2.0	CPE dictionaries
cve	http://scap.nist.gov/schema/vulnerability/0.4	NVD/CVE data feed elements and attributes
dc	http://purl.org/dc/elements/1.1/	Simple Dublin Core elements
ds	http://scap.nist.gov/schema/scap/source/1.2	SCAP source data stream collection
dt	http://scap.nist.gov/schema/xml-dsig/1.0	Security automation digital signature extensions
nvd	http://scap.nist.gov/schema/feed/vulnerability/2.0	Base schema for NVD data feeds
ocil	http://scap.nist.gov/schema/ocil/2.0	OCIL elements and attributes
oval	http://oval.mitre.org/XMLSchema/oval-common-5	OVAL elements and attributes
oval-def	http://oval.mitre.org/XMLSchema/oval-definitions-5	OVAL definitions
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-var	http://oval.mitre.org/XMLSchema/oval-variables-5	OVAL variable requirements
sch	http://purl.oclc.org/dsdl/schematron	Schematron used for validation
xccdf	http://checklists.nist.gov/xccdf/1.2	XCCDF policy documents
xlink	http://www.w3.org/1999/xlink	XML Linking Language
xml	http://www.w3.org/XML/1998/namespace	Common XML attributes
xs	http://www.w3.org/2001/XMLSchema	XML schema
xxxx-def, *xxxx*-sc	See [SP800-126A] for the mappings for OVAL definition and system characteristic schemas.	OVAL elements and attributes specific to an OS, Hardware, or Application type

2. SCAP 1.4 Definition

In general, the SCAP version 1.4 is comprised of 12 component specifications in five categories:

1. **Languages.** The SCAP languages provide standard vocabularies and conventions for expressing security policies, technical check mechanisms, and assessment results. The SCAP language specifications are the Extensible Configuration Checklist Description Format (XCCDF), Open Vulnerability and Assessment Language (OVAL), and Open Checklist Interactive Language (OCIL).
2. **Reporting formats.** SCAP reporting provides the necessary constructs to express collected information in standardized formats. The SCAP reporting format specifications are Asset Reporting Format (ARF) and Asset Identification. Although Asset Identification is not explicitly a reporting format, SCAP uses it as a key component in identifying the assets that reports relate to.
3. **Identification schemes.** The SCAP identification schemes provide a means to identify key concepts, such as software products, vulnerabilities, and configuration items that use standardized identifier formats. They also provide a means to associate individual identifiers with additional data pertaining to the subject of the identifier. The SCAP identification scheme specifications are the Common Platform Enumeration (CPE), Common Configuration Enumeration (CCE), and Common Vulnerabilities and Exposures (CVE).
4. **Measurement and scoring systems.** In SCAP, this refers to evaluating the specific characteristics of a security weakness (e.g., software vulnerabilities and security configuration issues) and generating a score that reflects their relative severity. The SCAP measurement and scoring system specifications are the Common Vulnerability Scoring System (CVSS) and Common Configuration Scoring System (CCSS).
5. **Integrity.** An SCAP integrity specification helps preserve the integrity of SCAP content and results. The Trust Model for Security Automation Data (TMSAD) is the SCAP integrity specification.

More specifically, the five categories for the *component specifications* included in SCAP 1.4 are:

1. **Languages**
    - Extensible Configuration Checklist Description Format (XCCDF) 1.2, a language for authoring security checklists/benchmarks and for reporting results of evaluating them [XCCDF]
    - Open Vulnerability and Assessment Language (OVAL) 5.12, a language for representing system configuration information, assessing machine states, and reporting assessment results [OVAL][3]

[3] See the SCAP 1.4 annex document [SP800-126A] for the OVAL component specification (core schema) versions and platform schema versions that are supported by SCAP 1.4.

  - Open Checklist Interactive Language (OCIL) 2.0, a language for representing checks that collect information from people or existing data stores made by other data collection efforts [OCIL]

2. **Reporting formats**
  - Asset Reporting Format (ARF) 1.1, a format for expressing the transport format of information about assets and the relationships between assets and reports [ARF]
  - Asset Identification 1.1, a format for uniquely identifying assets based on known identifiers and/or known information about the assets [AI]

3. **Identification schemes**
  - Common Platform Enumeration (CPE) 2.3, a nomenclature and dictionary of hardware, operating systems, and applications [CPE]
  - Common Configuration Enumeration (CCE) 5, a nomenclature and dictionary of software security configurations [CCE]
  - Common Vulnerabilities and Exposures (CVE), a nomenclature and dictionary of security-related software flaws[4] [CVE]

4. **Measurement and scoring systems**
  - Common Vulnerability Scoring System (CVSS) 3, a system for measuring the relative severity of software flaw vulnerabilities [CVSS]
  - Common Configuration Scoring System (CCSS) 1.0, a system for measuring the relative severity of system security configuration issues [CCSS]

5. **Integrity**
  - Trust Model for Security Automation Data (TMSAD) 1.0, a specification for using digital signatures in a common trust model applied to other security automation specifications [TMSAD]

All references to these specifications within this document are to the minor version numbers listed in the annex.[5] These versions represent the baseline of interoperability for all SCAP products that support SCAP 1.4. Support for older versions of these specifications may be included in an SCAP 1.4 product, but support for legacy SCAP versions should not interfere with the product's ability to address the requirements in this specification and the annex. In some cases, support for specific legacy versions of these specifications is required by this specification, such as the requirements discussed in Sec. 4.1.

Combinations of these specifications can be used together for particular functions, such as security configuration checking. These functions, known as *SCAP use cases*, are ways in which a product can use SCAP. The collective XML content used for a use case is called an *SCAP data stream*, which is a specific instantiation of SCAP content. There are two types of SCAP data

[4] CVE does not have a version number.
[5] Section 1 of the SCAP 1.4 annex document [SP800-126A] provides definitions for the terms "major version" and "minor version."

streams: an *SCAP source data stream* holds the input content, and an *SCAP result data stream* holds the output content. The major elements of a data stream (e.g., an XCCDF benchmark, a set of OVAL definitions) are referred to as *stream components*.

Products and source content may want to claim conformance to one or more of the SCAP use cases (see Sec. 5) for a variety of reasons. For example, a product may want to assert that it uses SCAP content properly and can interoperate with other products using valid SCAP content. Another example is a policy mandating that an organization use SCAP source content to perform vulnerability assessments and other security operations.

This section provides the high-level requirements that a product or source content must meet for conformance with the SCAP 1.4 specification. Such products and source content are referred to as *SCAP-conformant.* Most of the requirements listed in this section reference other sections that fully define the requirements.

If requirements are in conflict between component specifications, this document will provide clarification. If a component specification conflicts with this document, the requirements in this document SHALL take precedence. This document will be republished with errata as needed, and the errata SHALL take precedence over the original document content.

The requirements in [SP800-126A] SHALL take precedence over conflicting requirements in this document or the component specifications. If an SCAP specification or component specification and a schema are in conflict, the requirements in the specification SHALL take precedence over all conflicting requirements in the schema.

2.1. Product Conformance

There are two types of SCAP-conformant products: content producers and content consumers. *Content producers* generate SCAP source data streams. *Content consumers* accept existing SCAP source data streams, process them, and (when required by the supported use cases) produce SCAP result data streams. Products that claim conformance with the SCAP 1.4 specification SHALL comply with the following requirements:

1. **Adhere to component specifications.**

   Adhere to the requirements in each applicable SCAP component specification, both for each component that the product implements and for each component required to implement the selected SCAP use cases. The authoritative references for component specifications are listed in [SP800-126A].

2. **Implement errata.**

   Adhere to the requirements in the errata for this document and SP 800-126A that are in effect as of the product's release date.

3. **Producer requirements.**

   For content producers, generate SCAP source data streams that are (a) XML well-formed, (b) schema-valid per the applicable component specifications, and (c) conformant to the source content conformance requirements in Sec. 2.2 and the use case requirements in Sec. 5.

4. **Consumer requirements.**

   For content consumers, accept and process SCAP source data streams and, when required by the supported use cases, generate SCAP result data streams that are XML well-formed and schema-valid. Content consumers SHALL follow all processing requirements in Sec. 4 for each selected SCAP component specification and each component specification required to implement the selected SCAP use cases and SHALL detect and report non-conformant input.

5. **Conformance claim.**

   Make an explicit claim of conformance to this specification in documentation provided to end users. The claim SHALL identify at least (a) this specification and version (SCAP 1.4), (b) the implemented roles (e.g., producer, consumer), (c) the supported SCAP use cases (Sec. 5), and (d) the implemented SCAP component specifications and versions.

2.2. Source Content Conformance

Source content (i.e., source data streams) that claim conformance with the SCAP 1.4 specification SHALL comply with the following requirements:

1. **Component specifications.**

   Adhere to the requirements specified in each applicable SCAP component specification. This includes both the component specifications explicitly selected for the content and any additional component specifications required to support the selected SCAP use cases. Authoritative references for each specification are provided in [SP800-126A].

2. **Errata.**

   Adhere to the requirements published in errata for this document and SP 800-126A. In the event of a conflict, the errata SHALL take precedence.

3. **Syntax, structure, and use cases.**

   Follow all syntax, structural, and design requirements defined in Sec. 3 for each applicable component specification. In addition, follow all use case-specific requirements defined in Sec. 5.

3. SCAP Content Requirements and Recommendations

This section defines the SCAP 1.4 content syntax, structure, and development requirements and recommendations for SCAP-conformant content and products. Organizations are encouraged to adopt the optional recommendations to promote stronger interoperability and greater content consistency. The first part of the section discusses SCAP source data streams. The middle of the section groups requirements and recommendations by specification: XCCDF, OVAL, OCIL, CPE, CCE, CVE, CVSS, and CCSS, in that order. Finally, the last part of the section discusses applying XML digital signatures to source data streams.

3.1. SCAP Source Data Stream

This subsection discusses SCAP source data streams only; SCAP result data streams are discussed in Sec. 4.4 as part of the requirements for SCAP processing.

An *SCAP source data stream collection* is composed of SCAP data streams and SCAP source components.[6] The components section contains an unbounded number of *SCAP source components*, each consisting of data expressed using one or more of the SCAP specifications. The data streams section contains one or more source data streams, each of which references the source components that compose the data stream. This model allows source components to be reused across multiple data streams. Many data streams are allowed in a data stream collection to allow for the grouping of related or similar source data streams.

Figure 1 shows a possible relationship between data stream collections, data streams, and components.

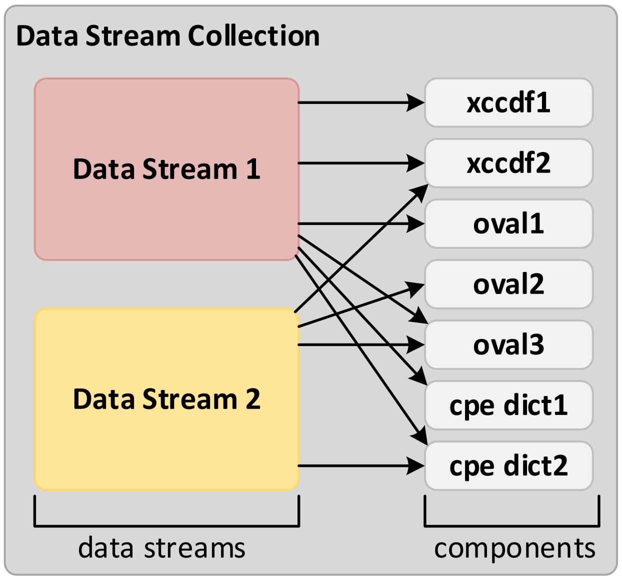

Fig. 1. Notional SCAP data stream collection

[6] See https://scap.nist.gov/revision/1.4/#example for a sample of an SCAP source data stream collection and its sections.

In Fig. 1, data stream 1 points to xccdf1, xccdf2, oval1, oval3, cpe dict1, and cpe dict2. Data stream 2 points to xccdf2, oval2, oval3, and cpe dict2. Each data stream is a collection of links to the components it references, represented as `<ds:component-ref>`. Each component-ref encapsulates the information required to allow the content consumer to connect the components that are embedded in content with the data stream component that should be used. Content authors MAY place components in any order. For example, some authors might choose to place dictionary components first to help optimize data stream parsing.

Links in a `<ds:component-ref>` element serve two purposes: to indicate which component is being referred to and to provide a map to associate references within a component to other links within the data stream. The latter allows a data stream to define context for each component's references within the bounds of the data stream's own set of links. Figure 2 provides a conceptual example that illustrates how a data stream is constructed.

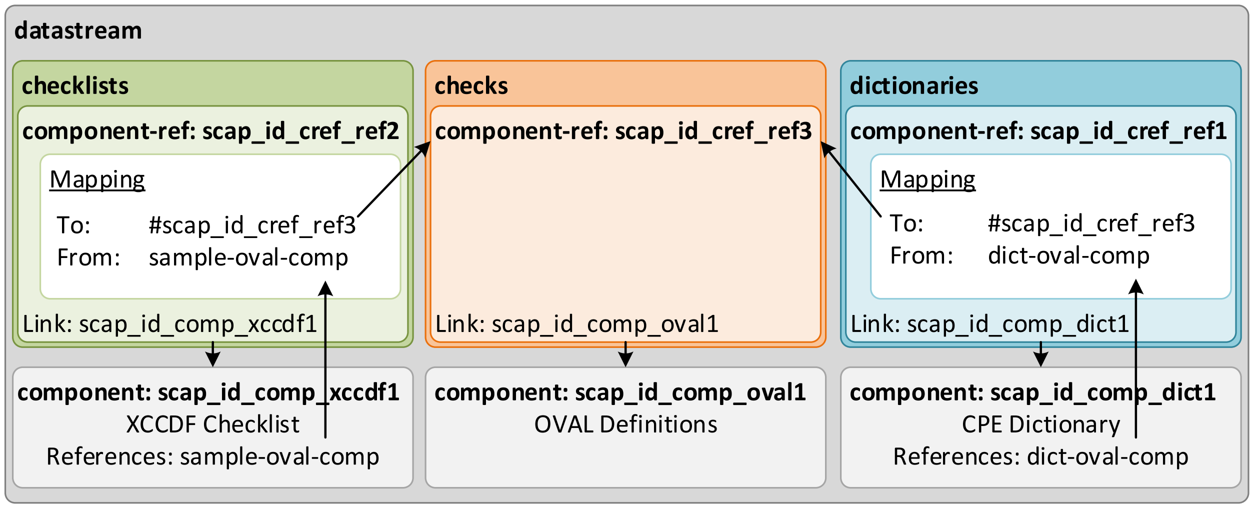
datastream
checklists
component-ref: scap_id_cref_ref2
Mapping
To: #scap_id_cref_ref3
From: sample-oval-comp
Link: scap_id_comp_xccdf1
checks
component-ref: scap_id_cref_ref3
Link: scap_id_comp_oval1
dictionaries
component-ref: scap_id_cref_ref1
Mapping
To: #scap_id_cref_ref3
From: dict-oval-comp
Link: scap_id_comp_dict1
component: scap_id_comp_xccdf1
XCCDF Checklist
References: sample-oval-comp
component: scap_id_comp_oval1
OVAL Definitions
component: scap_id_comp_dict1
CPE Dictionary
References: dict-oval-comp

Fig. 2. SCAP data stream

The following XML is a stripped-down example of the source data stream depicted in Fig. 2.

```
<ds:data-stream-collection id="scap_datastream_collection_1" schematron-
version="1.4">
 <ds:data-stream id="scap_id_datastream_ds1" scap-version="1.4" use-
case="CONFIGURATION">
   <ds:dictionaries>
     <ds:component-ref id="scap_id_cref_ref1" xlink:href="#scap_id_comp_dict1">
       <cat:catalog>
         <cat:uri name="dict-oval-comp" uri="#scap_id_cref_ref3"/>
       </cat:catalog>
     </ds:component-ref>
   </ds:dictionaries>
   <ds:checklists>
     <ds:component-ref id="scap_id_cref_ref2" xlink:href="#scap_id_comp_xccdf1">
       <cat:catalog>
         <cat:uri name="sample-oval-comp" uri="#scap_id_cref_ref3"/>
       </cat:catalog>
     </ds:component-ref>
   </ds:checklists>
   <ds:checks>
     <ds:component-ref id="scap_id_cref_ref3" xlink:href="#scap_id_comp_oval1"/>
   </ds:checks>
```

```
20   </ds:data-stream>
21   <ds:component id="scap_id_comp_xccdf1" timestamp="2016-01-22T14:00:00">
22     <xccdf:Benchmark id="xccdf_gov.nist_benchmark_SCAP13" style=" SCAP_1.4">
23       …
24       <xccdf:Rule id="xccdf_gov.nist_rule_id-001">
25         <xccdf:check system="http://oval.mitre.org/XMLSchema/oval-definitions-5">
26           <xccdf:check-content-ref href="sample-oval-comp"
name="oval:gov.nist:def:1"/>
27         </xccdf:check>
28       </xccdf:Rule>
29     </xccdf:Benchmark>
30   </ds:component>
31   <ds:component id="scap_id_comp_oval1" timestamp="2016-01-22T14:00:00">
32     <oval-def:oval_definitions>...</oval-def:oval_definitions>
33   </ds:component>
34   <ds:component id="scap_id_comp_dict1" timestamp="2016-01-22T14:00:00">
35     <cpe2-dict:cpe-list>
36       <cpe2-dict:cpe-item
name="cpe:/a:oracle:database_server:11.1.0.6.0::enterprise">
37         <cpe2-dict:check href="dict-oval-comp"
38          system="http://oval.mitre.org/XMLSchema/oval-definitions-5”>
39          oval:gov.nist:def:2</cpe2-dict:check>
40         <cpe-dict-ext:cpe23-item
41          name="cpe:2.3:a:oracle:database_server:11.1.0.6.0:-:-:-:enterprise:-:-:-
"/>
42       </cpe2-dict:cpe-item>
43     </cpe2-dict:cpe-list>
44   </ds:component>
45  </ds:data-stream-collection>
```

In Fig. 2, the data stream links to three components. The OVAL component scap_id_comp_oval1 (see XML lines 31-33 above) does not reference external content, so there are no mappings captured for it. The XCCDF component (scap_id_comp_xccdf1) (see XML lines 21-30) and the CPE Dictionary component (scap_id_comp_dict1) (see XML lines 34-44) reference other components (e.g., scap_id_cref_ref3).

When referencing components within the example data stream, a mapping indicates that when scap_id_comp_xccdf1 references “sample-oval-comp,” the content is found through the link to the component identified as “scap_id_comp_oval1” (see XML lines 26, 13, and 18). Similarly, when the scap_id_comp_dict1 component references “dict-oval-comp,” the component reference is resolved through the link to the component identified as “scap_id_comp_oval1” (see XML lines 37, 6, and 18). This approach associates SCAP components within a data stream at the SCAP logical level, allowing components to be reused across data streams within the same data stream collection. This reuse can be accomplished irrespective of how references are made within a given component.

The design of the SCAP source data stream is important for the following reasons:

1. Individual components may be developed outside of an SCAP data stream, where linking to other components is not necessarily known when the component is created.
2. The SCAP source data stream creates links between different components that were not necessarily designed to reference each other. For example, XCCDF was not designed to reference a particular checking system; it can reference OVAL, OCIL, and other checking systems.

3. The logical link mapping in the data stream places a layer of capability within the data stream to control the dereferencing of URIs within components, creating a complete solution related to bundling components.
4. The SCAP source data stream format is intended to be easily adaptable for use in future communication models (e.g., web services, transport protocols, tasking mechanisms).
5. The SCAP source data stream format supports more comprehensive validation of component content, including interrelationships between components.

3.1.1. Source Data Stream Data Model

The tables in this section formalize the SCAP source data stream data model. The tables contain requirements and SHALL be interpreted as follows:

- The "Element Name" field indicates the name for the XML element being described. Each element name has a namespace prefix indicating the namespace to which the element belongs. See Table 1 for a mapping of namespace prefixes to namespaces.
- The "Element Definition" field indicates the prose description of the element. The definition field MAY contain key words, as indicated in [RFC2119].
- The "Properties" field is broken into four subfields:
    1. The "Name" column indicates the name of a property that MAY, SHOULD, or SHALL be included in the described element, in accordance with the cardinality indicated in the "Count" column and any [RFC2119] requirement words in the "Property Definition" column.
    2. The "Type" column indicates the REQUIRED data type for the value of the property. There are two categories of types: literal and element. A literal type indicates the type of literal as defined in [XMLS]. An element type references the name of another element that ultimately defines that property.
    3. The "Count" column indicates the cardinality of the property within the element. The property SHALL be included in the element in accordance with the cardinality. If a range is given, and "n" is the upper bound of the range, then the upper limit SHALL be unbounded.
    4. The "Property Definition" column defines the property in the context of the element. The definition MAY contain key words, as indicated in [RFC2119].

Table 2. ds:data-stream-collection

Element Name: ds:data-stream-collection			
Element Definition	The top-level element for an SCAP data stream collection. It contains the data streams, the components that comprise this data stream collection, and any data stream signatures.		
Properties:			
Name	**Type**	**Count**	**Property Definition**
id	literal – ID	1	The identifier for the data stream collection. This identifier SHALL be globally unique (see Sec. 3.1.3).
schematron-version	literal – token	1	The version of the SCAP Requirements Schematron schema to which the data stream collection conforms.
data-stream	element – ds:data-stream	1-n	An element that represents a single data stream collection (see Table 3).
component	element – ds:component	1-n	An element that represents content expressed using an SCAP component specification (see Table 12).
extended-component	element – ds:extended-component	0-n	An element that holds non-SCAP components to enable extension (see Table 13).
Signature	element – dsig:Signature	0-n	An XML digital signature element. Sections 3.10 and 4.8 define the requirements for this element.

Table 3. ds:data-stream

Element Name: ds:data-stream			
Element Definition	A data stream. This element contains the links to all of the components that comprise this data stream.		
Properties			
Name	**Type**	**Count**	**Property Definition**
id	literal – ID	1	The identifier for the data stream. This identifier SHALL be globally unique (see Sec. 3.1.3).
use-case	literal – token	1	The use case represented by the data stream. The value SHALL be one of the following: CONFIGURATION, VULNERABILITY, INVENTORY, or OTHER. The value selected SHALL indicate which type of content is being represented, as defined in Sec. 5. The value "OTHER" is for content that does not correspond to a specific use case. This content SHALL be valid according to the requirements defined in Sec. 3 and 4.
scap-version	literal – token	1	The targeted SCAP version. The value SHALL be 1.4, 1.3, 1.2, 1.1, or 1.0. The value SHALL indicate which version of SCAP the content is conformant with. 1.4 SHALL be specified to be conformant with this version of SCAP.
timestamp	literal – dateTime	0-1	The date and time when this data stream was created.
dictionaries	element – ds:dictionaries	0-1	Links to dictionary components (see Table 4).
checklists	element – ds:checklists	0-1	Links to checklist components (see Table 5).
checks	element – ds:checks	1	Links to check components (see Table 6).

extended-components	element – ds:extended-components	0-1	Links to non-standard components (see Table 7). See Section 4.2 for information on processing this element.

543 **Table 4. ds:dictionaries**

Element Name: ds:dictionaries			
Element Definition	A container element that holds references to one or more dictionary components.		
Properties			
Name	**Type**	**Count**	**Property Definition**
component-ref	element – component-ref	1-n	SHALL contain a reference to a dictionary component (a component containing CPE dictionary content).

544 **Table 5. ds:checklists**

Element Name: ds:checklists			
Element Definition	A container element that holds references to one or more checklists.		
Properties			
Name	**Type**	**Count**	**Property Definition**
component-ref	element – component-ref	1-n	SHALL contain a reference to a checklist component (i.e., a component containing an *<xccdf:Benchmark>* or an *<xccdf:Tailoring>* element).

545 **Table 6. ds:checks**

Element Name: ds:checks			
Element Definition	A container element that holds references to one or more check components.		
Properties			
Name	**Type**	**Count**	**Property Definition**
component-ref	element – component-ref	1-n	SHALL contain a reference to a check component (i.e., a component containing check content). See Sec. 3.2.4.2 for information on SCAP-checking system support and requirements.

546 **Table 7. ds:extended-components**

Element Name: ds:extended-components			
Element Definition	A container element that holds references to one or more extended components for the SCAP data stream, including non-standard components.		
Properties			
Name	**Type**	**Count**	**Property Definition**
component-ref	element – component-ref	1-n	SHALL contain a reference to a non-standard component (a *<ds:extended-component>* element). See Table 13.

Table 8. ds:component-ref

Element Name: ds:component-ref			
Element Definition	An element that encapsulates the information necessary to link to a component within the data stream collection or to external content, which gives context to the reference. This is a simple XLink [XLINK].		
Properties			
Name	**Type**	**Count**	**Property Definition**
id	literal - ID	1	The identifier for the reference. This identifier SHALL be globally unique (see Sec. 3.1.3).
type	literal – xlink:type	0-1	The type of XLink represented. The *<ds:component-ref>* is constrained to a simple XLink, so the value of this field SHALL be 'simple' if specified.
href	literal – xlink:href	1	A URI to the target component (either local to the data stream collection or remote). When referencing a local component, the URI SHALL be in the form '#' + componentId (e.g., "#component1"). When referencing external content, the URI SHALL be in the form of scheme:[//[user:password@]host[:port]][/]path[?query][#fragment], as specified in [RFC3986], and SHALL dereference to an XML stream that includes the SCAP source data stream collection and the target component (e.g., "file:Data_Stream_Collection.xml#scap_gov.nist_comp_1").
catalog	element – cat:catalog	0-1	An XML Catalog that defines the mapping between external URI links in the component being referenced by this *<ds:component-ref>* and where those URIs should map to within the context of this data stream. See Table 9.

Table 9. cat:catalog

Element Name: cat:catalog			
Element Definition	A catalog element defined by the OASIS XML Catalog specification [XMLCAT]. Within an SCAP source data stream, this element SHALL contain one or more *<cat:uri>* and/or *<cat:rewriteURI>* elements, and it SHALL NOT contain any other elements or attributes. Refer to Sec. 7 of [XMLCAT] for information on determining which catalog entry to apply.		
Properties			
Name	**Type**	**Count**	**Property Definition**
uri	element – cat:uri	0-n (at least 1 of this or rewriteURI SHALL be provided)	Maps a reference in the enclosing *<ds:component-ref>* element's component to some other *<ds:component-ref>* element that SHALL be used to resolve the reference. See Table 10.
rewriteURI	element – cat:rewriteURI	0-n (at least 1 of this or uri SHALL be provided)	A rewriteURI element defined by the OASIS XML Catalog specification [XMLCAT]. Within an SCAP source data stream, this element can be used to rewrite the beginning of a reference in the enclosing *<ds:component-ref>* element's component to some other *<ds:component-ref>* element that SHALL be used to resolve the reference. See Table 11.

Table 10. cat:uri

Element Name: cat:uri			
Element Definition	A URI element defined by the OASIS XML Catalog specification [XMLCAT]. Within an SCAP source data stream, this element maps a reference in the enclosing `<ds:component-ref>` element's component to some other `<ds:component-ref>` element that SHALL be used to resolve the reference. A `<cat:uri>` element SHALL have a `@name` attribute and a `@uri` attribute.		
Properties			
Name	**Type**	**Count**	**Property Definition**
name	literal – xs:anyURI	1	The `@name` attribute is the source of the mapping and SHALL contain a URI that matches a "referenced URI" in the data stream component referenced by the `<ds:component-ref>` that holds this element. The "referenced URI" is a URI entry defined within the model used within the data stream component.
uri	literal – xs:anyURI	1	The `@uri` attribute is the destination of the mapping and SHALL be populated with the value "#" + `@id` of a `<ds:component-ref>`. When resolving the URI in the `@name` attribute, the `<ds:component-ref>` pointed to by the `@uri` attribute SHALL be used.

Table 11. cat:rewriteURI

Element Name: cat: rewriteURI			
Element Definition	A rewriteURI element defined by the OASIS XML Catalog specification [XMLCAT]. Within an SCAP source data stream, this element can be used to rewrite the beginning of a reference in the enclosing `<ds:component-ref>` element's component to some other `<ds:component-ref>` element that SHALL be used to resolve the reference. A `<cat: rewriteURI>` element SHALL have a `@uriStartString` attribute and a `@rewritePrefix` attribute specified. See [XMLCAT] for more details.		
Properties			
Name	**Type**	**Count**	**Property Definition**
uriStartString	literal – xs:anyURI	1	The `@uriStartString` attribute SHALL be populated with the start of a URI of an external link specified within the component referenced by this element's enclosing `<ds:component-ref>` element that is to be replaced.
rewritePrefix	literal – xs:anyURI	1	The `@rewritePrefix` attribute SHALL be populated with a string that will replace the matched `@uriStartString` value. The resulting URI SHALL be used to resolve the link.

Table 12. ds:component

Element Name: ds:component			
Element Definition	A container for a single component. The types of components are defined in Sec. 3.1.2.		
Properties			
Name	**Type**	**Count**	**Property Definition**
id	literal – ID	1	The identifier for the component. This identifier SHALL be globally unique (see Sec. 3.1.3).
timestamp	literal – dateTime	1	Indicates when the `<ds:component>` was created or last updated.

Table 13. ds:extended-component

Element Name: ds:extended-component			
Element Definition	This element holds content that does not fit within the other defined component types described in Table 12. Authors SHOULD use this element as an extension point to capture content that is not captured in a regular component. The content of this element SHALL be an XML element in a namespace other than the SCAP source data stream namespace. Linking through a `<ds:extended-component>` element SHALL make the data stream non-conformant with SCAP.		
Properties			
Name	**Type**	**Count**	**Property Definition**
id	literal – ID	1	The identifier for the component. This identifier SHALL be globally unique (see Sec. 3.1.3).
timestamp	literal – dateTime	1	Indicates when the `<ds:extended-component>` was created or last updated.

3.1.2. Source Data Stream Collection Validation

The SCAP source data stream collection SHALL validate against the XML schema representation for the source data stream and all associated Schematron schemas. The SCAP components referenced by each `<ds:component>` and `<ds:extended-component>` element SHALL validate against the corresponding component schema and its embedded Schematron rules. All of the SCAP-related schemas are referenced at https://scap.nist.gov/revision/1.4/#schema. See Sec. 2 in [SP800-126A] for a list of SCAP component schema and Schematron schema locations. These XML and Schematron schemas will be updated if any errors are found. If the old schema links change, updated links will be provided in the annex as errors are corrected.

Each SCAP source data stream component SHALL use one of the elements specified in Table 14 as its document element.

Table 14. SCAP source data stream component document elements

Component	Document Element
XCCDF Benchmark	`<xccdf:Benchmark>`
XCCDF Tailoring	`<xccdf:Tailoring>`
OVAL	`<oval-def:oval_definitions>`
OCIL	`<ocil:ocil>`
CPE Dictionary	`<cpe2-dict:cpe-list>`

SCAP source data stream components SHOULD NOT use any constructs that are deprecated in its associated specification. While Sec. 4.1 requires that products support deprecated constructs, these constructs should be avoided to minimize the impact to content use when the constructs are removed from future revisions of the associated specifications. Single data streams in a data stream collection SHALL NOT reference any component in the collection more than once.

If applicable, each component SHALL validate against its associated Schematron schema. For the SCAP source data stream collection, it SHALL validate against the version of the SCAP

Schematron rules as specified on the *`<ds:data-stream-collection>`* element's *`@schematron-version`* attribute, and it SHOULD also validate against the latest Schematron rules. NIST provides and maintains a set of Schematron rules to check well-formed SCAP content. The Schematron schemas for the SCAP specification and its applicable component specifications are located at https://scap.nist.gov/revision/1.4/#schematron. Source content SHOULD pass all Schematron assertions in the Schematron rule files. When creating source content, failed assertions with a "WARNING" or "INFO" flag MAY be disregarded if the assertion discovers an issue in the content that is justifiable and expected based on the needs of the content author. When executing source content, all failed assertions with a "WARNING" or "INFO" flag SHALL be disregarded.

The Schematron schemas are interpretations of the specifications, and the implementations of their rules are subject to change. Whenever a change is made to a Schematron schema used for this SCAP version, the SCAP Schematron change log document will be updated, and the new Schematron schema will be posted. The latest Schematron schema SHOULD be used in place of any earlier versions. If the latest file is unavailable, the version specified **on the** *`<ds:data-stream-collection>`* **element's** *`@schematron-version`* **attribute** SHALL be used instead.

3.1.2.1. Informative Notes

- **Validation tooling:** NIST has provided an SCAP Content Validation Tool to assist with checking that SCAP source and result content is well-formed, cross-references resolve, and required values are present. It can report errors and warnings in XML and HTML formats. Use of this or any particular tool is not required for conformance. The tool's availability, feature set, and maintenance are not guaranteed and may change over time. Products and content authors MAY employ any equivalent validation tool that implements the requirements in this document and the associated component specifications.
- **Schematron rules:** NIST has published Schematron rules to assist content authors and implementers in checking for conditions that are not enforced by XML Schema, such as structural or semantic consistency across components. Following these rules is not required for conformance, and NIST does not guarantee their availability, accuracy, or maintenance over time. Content authors and product developers MAY employ these or equivalent rule sets as an additional quality-assurance mechanism.

3.1.3. Globally Unique Identifiers

The elements listed in Table 15 have special conventions for the format of their identifiers (`@id` attribute).

Table 15. Element identifier format convention

Element	Identifier Format Convention
`<ds:data-stream-collection>`	scap_*namespace*_collection_*name*
`<ds:data-stream>`	scap_*namespace*_datastream_*name*

Element	Identifier Format Convention
`<ds:component-ref>`	scap_*namespace*_cref_*name*
`<ds:component>`	scap_*namespace*_comp_*name*
`<ds:extended-component>`	scap_*namespace*_ecomp_*name*

Authors SHALL follow these conventions because they preserve the global uniqueness of the resulting identifiers. In Table 15, *namespace* contains a valid reverse-DNS-style string (limited to letters, numbers, periods, and the hyphen character) that is associated with the content author. Examples include "com.acme.finance" and "gov.tla." These namespace strings MAY have any number of parts, and SCAP content consumers processing them SHALL treat them as case-insensitive (e.g., com.ABC is considered identical to com.abc). The *name* in the format conventions SHALL be an NCName-compliant string [XMLS].

3.2. Extensible Configuration Checklist Description Format (XCCDF)

This section lists requirements and recommendations for using the Extensible Configuration Checklist Description Format (XCCDF) to express an XCCDF benchmark or tailoring component of an SCAP source data stream (see Table 14). They are organized by the following categories: general, *<xccdf:Benchmark>*, *<xccdf:Profile>*, *<xccdf:Rule>*, *<xccdf:Value>*, and *<xccdf:Group>*.

3.2.1. General

The *@xml:base* attribute SHALL NOT be allowed in XCCDF content. This attribute is not compatible with the SCAP data stream model.

Descriptive information within XCCDF MAY be used by SCAP products to assist in selecting the appropriate SCAP data stream, ensure that the most recent or correct version of an XCCDF document is used, and provide additional information about the document. The following requirements and conventions apply to the *<xccdf:Benchmark>*, *<xccdf:Profile>*, *<xccdf:Value>*, *<xccdf:Group>*, and *<xccdf:Rule>* elements:

1. One or more instances of the *<xccdf:title>* element SHALL be provided. Each instance SHALL contain a text value that briefly indicates the purpose of the containing element.
2. One or more instances of the *<xccdf:description>* element SHALL be provided. Each instance SHALL contain a text value that describes the purpose of the containing element.

XInclude elements SHALL NOT be included in XCCDF content [XINCLUDE].

All remaining OPTIONAL elements in the XCCDF schema MAY be included at the author's discretion unless otherwise noted in this document.

3.2.2. The <xccdf:Benchmark> Element

The following requirements and recommendations apply to the *<xccdf:Benchmark>* element:

1. The *<xccdf:version>* element and the *@id* attribute SHALL be used together to uniquely identify all revisions of a benchmark.
    a. Multiple revisions of a single benchmark SHOULD have the same *@id* attribute value and different *<xccdf:version>* element values so that someone who reviews the revisions can readily identify them as multiple versions of a single benchmark.
    b. Multiple revisions of a single benchmark SHOULD have *<xccdf:version>* element values that indicate the revision sequence so that the history of changes from the original benchmark can be determined.
    c. The *@time* attribute of the *<xccdf:version>* element SHOULD be used for a timestamp of when the benchmark was defined.
2. The *@update* attribute of the *<xccdf:version>* element SHOULD be used for a URI that specifies where updates to the benchmark can be obtained.
3. The *<xccdf:Benchmark>* element SHALL have an *@xml:lang* attribute.
4. The *@style* attribute SHOULD have the value "SCAP_1.4."
5. The *<xccdf:status>* element SHALL indicate the current status of the benchmark document. The associated text value SHALL be "draft" for documents released in public draft state and "accepted" for documents that have been officially released by an organization. The *@date* attribute SHALL be populated with the date of the status change. Additional *<xccdf:status>* elements MAY be included to indicate historic status transitions.
6. The *<xccdf:metadata>* element SHALL be provided and SHALL, at minimum, contain the Dublin Core [DCES] terms from Table 16. If provided, additional Dublin Core terms SHALL follow the required terms within the element sequence.

Table 16. Use of Dublin Core terms in <xccdf:metadata>

Dublin Core Term	Description of Use
<dc:creator>	The person, organization, and/or service that created the benchmark.
<dc:publisher>	The person, organization, and/or service that published the benchmark.
<dc:contributor>	The person, organization, and/or service that contributed to the creation of the benchmark.
<dc:source>	An identifier that indicates the organizational context of the benchmark's *@id* attribute. An organizationally specific URI SHOULD be used.

3.2.3. The <xccdf:Profile> Element

As stated in the XCCDF specification, the use of an *<xccdf:Profile>* element is not required, even though SCAP content commonly it.

Use of the *<xccdf:set-complex-value>* element within the *<xccdf:Profile>* element SHALL NOT be allowed. Use of complex values is disallowed because the behavior for mapping XCCDF complex values to OVAL variables is not defined.

3.2.4. The <xccdf:Rule> Element

The following requirements and recommendations apply to the *<xccdf:Rule>* element. The topics they address are *<xccdf:ident>* elements, *<xccdf:check>* elements, patching up-to-date rules, and CVSS and CCSS scores.

3.2.4.1. The <xccdf:ident> Element

Each *<xccdf:Rule>* element SHALL include an *<xccdf:ident>* element that contains a CVE, CCE, or CPE identifier reference if an appropriate identifier exists. The meaning of the identifier SHALL be consistent with the recommendation implemented by the *<xccdf:Rule>* element. The *<xccdf:ident>* element content SHALL match the corresponding CVE, CCE, or CPE identifier found in the associated OVAL definitions if the rule references an OVAL definition, if an appropriate identifier exists, and if that OVAL definition is the only input to the rule's final result.

When referencing a CVE, CCE, or CPE identifier, an *<xccdf:Rule>* element SHALL have a purpose consistent with one of the rows in Table 17.

Table 17. <xccdf:Rule> and <xccdf:ident> element values

Purpose of the <xccdf:Rule>	OVAL Definition Class	Identifier Type	Value for <xccdf:ident> @system attribute[7]
Check compliance with a configuration setting	compliance	CCE	http://cce.mitre.org
Perform a software inventory check	inventory	CPE	http://cpe.mitre.org
Check for a software flaw vulnerability	vulnerability	CVE	http://cve.mitre.org

Based on the purpose of the *<xccdf:Rule>* element, the *<xccdf:Rule>* SHALL define its *<xccdf:ident>* element's *@system* attribute using the corresponding value from Table 17. Also, if the *<xccdf:Rule>* element references an OVAL definition, it SHALL reference an OVAL definition of the specified class.

Here is a partial example of a rule intended to check compliance with a configuration setting:

```
<xccdf:Rule id="xccdf_gov.nist.fdcc.xp_value_AuditAccountLogonEvents">
    …
    <xccdf:ident system="http://cce.mitre.org">CCE-3867-0</xccdf:ident>
    …
</xccdf:Rule>
```

See Sec. 4.5.1 for information on the meaning of a "pass/fail" rule result relating to each of the identifier types in Table 17. All rules that contain CCE, CPE, or CVE entries in their

[7] The URI values in this column are used to identify the naming system being used and have a MITRE designation due to historic naming conventions.

`<xccdf:ident>` elements SHALL obey these meanings. As a result, such `<xccdf:ident>` elements SHALL only be included if the recommendation is identical to these associated meanings or if they have a *@con:negate* attribute (as described in Sec. 4.5.1) set to comply with the intended meaning (by default, *@con:negate* is set to false). In SCAP, an `<xccdf:ident>` element is not simply a reference to related material — it is a declaration of exact alignment with the described meanings.

An `<xccdf:ident>` element referencing a CVE, CCE, or CPE identifier SHALL be ordered before other `<xccdf:ident>` elements referencing non-SCAP identifiers. Identifiers from previous revisions of CCE or CPE MAY also be specified following the SCAP identifiers.

3.2.4.2. The <xccdf:check> Element

The following requirements and recommendations apply to the `<xccdf:check>` element:

1. The `<xccdf:check-content>` element SHALL NOT be used to embed check content directly into XCCDF content.
2. At least one `<xccdf:check-content-ref>` element SHALL be provided for each `<xccdf:check>` element.
3. When evaluating an `<xccdf:check-content-ref>` element within an `<xccdf:check>` element, its *@href* attribute SHALL contain a "#" and the *@id* of a `<ds:component-ref>` element or SHALL be resolved in the context of the XML Catalog specified as part of the `<ds:component-ref>` element that is referencing this benchmark. In either case, the *@href* attribute SHALL ultimately resolve to a `<ds:component-ref>` element in the data stream referencing the benchmark containing this `<xccdf:check-content-ref>` element. See Sec. 3.1.1 for additional information on `<ds:component-ref>` resolution.

This version of SCAP supports the use of only OVAL and/or OCIL-checking systems in SCAP-conformant content. Use of these checking systems SHALL be restricted as follows:

1. OVAL checking system
   i. Use of the OVAL checking system SHALL be indicated by setting the `<xccdf:check>` element's *@system* attribute to *http://oval.mitre.org/XMLSchema/oval-definitions-5*.
   ii. The *@href* attribute in the `<xccdf:check-content-ref>` element SHALL reference an OVAL source data stream component using the `<ds:component-ref>` approach defined above.
   iii. Use of the *@name* attribute in the `<xccdf:check-content-ref>` element is OPTIONAL. If present, it SHALL reference an OVAL definition in the designated OVAL source data stream component. Otherwise, see Sec. 4.5.2 for information on the use of the *@multi-check* attribute.

2. OCIL-checking system
    i. OCIL questionnaires SHOULD NOT be used if OVAL can perform the same check correctly.
    ii. Use of the OCIL checking system SHALL be indicated by setting the `<xccdf:check>` element's `@system` attribute to `http://scap.nist.gov/schema/ocil/2`.
    iii. The `@href` attribute in the `<xccdf:check-content-ref>` element SHALL reference an OCIL source data stream component using the `<ds:component-ref>` approach defined above.
    iv. Use of the `@name` attribute in the `<xccdf:check-content-ref>` element is OPTIONAL. If present, it SHALL reference an OCIL questionnaire in the designated OCIL source data stream component. Otherwise, see Sec. 4.5.2 for information on the use of the `@multi-check` attribute.
    v. All requirements in Appendix B of NIST Interagency Report (IR) 7692, *Specifications for the Open Checklist Interactive Language (OCIL) Version 2.0* [OCIL], SHALL be followed.

A checking system that is not supported by SCAP MAY be used in XCCDF content. There is no guarantee that an SCAP implementation will be capable of processing any additional checking system data used in this content. To ensure interoperability, SCAP has standardized the use of OVAL and OCIL-checking systems. Content containing the use of checking systems other than the OVAL and OCIL-checking systems SHALL NOT be considered well-formed with regard to SCAP.

3.2.4.3. Use of a Patches Up-To-Date Rule

An OVAL source data stream component MAY be used to represent a series of checks to verify that patches have been installed. Historically, an XCCDF convention has been used to identify such a reference. An XCCDF benchmark MAY include a patches up-to-date rule that SHALL reference an OVAL source data stream component.

When implementing a patches up-to-date XCCDF rule that checks for patches via numerous OVAL patch class definitions, the following approach SHALL be used:

1. The source data stream SHALL include the OVAL source data stream component referenced by the patches up-to-date rule, which contains one or more OVAL patch class definitions and MAY contain other class definitions.
2. The `<xccdf:Rule>` element that references an OVAL source data stream component SHALL have the `@id` attribute value of *xccdf_NAMESPACE_rule_security_patches_up_to_date*, where *NAMESPACE* is the reverse DNS format namespace associated with the content maintainer.
3. Each `<xccdf:check-content-ref>` element SHALL omit the `@name` attribute.

4. The *@multi-check* attribute of the *<xccdf:check>* element SHALL be set to "true." This causes a separate *<xccdf:rule-result>* to be generated for each OVAL patch definition. See Sec. 4.5.2 for more information.

Use of this approach allows for the individual OVAL patch definitions to be easily identified along with the XCCDF rule checking whether patches are up to date.

The following example of a patches up-to-date rule references numerous OVAL patch class definitions:

```
<xccdf:Rule
     id="xccdf_gov.nist.usgcb.win_rule_security_patches_up_to_date"
     selected="true">
   <xccdf:title>Security Patches Up-To-Date</xccdf:title>
   <xccdf:description>Keep systems up to current patch
levels</xccdf:description>
   <xccdf:check system=http://oval.mitre.org/XMLSchema/oval-definitions-5
        multi-check="true">
      <xccdf:check-content-ref href="scap-windows-patches"/>
   </xccdf:check>
</xccdf:Rule>
```

When implementing a patches up-to-date XCCDF rule that checks for patches via a single OVAL patch class definition, the following approach SHALL be used:

1. The source data stream SHALL include the OVAL source data stream component referenced by the patches up-to-date rule, which contains one or more OVAL patch class definitions and MAY contain other class definitions.
2. The *<xccdf:Rule>* element that references an OVAL source data stream component SHALL have the *@id* attribute value of *xccdf_NAMESPACE_rule_security_patches_up_to_date*, where *NAMESPACE* is the reverse DNS format namespace associated with the content maintainer.
3. Each *<xccdf:check-content-ref>* element SHALL refer to the single OVAL definition performing the patches up-to-date check.
4. The *@multi-check* attribute of the *<xccdf:check>* element SHALL be set to "false," which is the default value.

Use of a single OVAL patch definition provides for easier content maintenance and makes it easy to identify both the XCCDF rule and the patch class definition used for checking whether patches are up to date.

Here is a patches up-to-date rule example that references a single OVAL patch class definition:

```
<xccdf:Rule
     id="xccdf_gov.nist.usgcb.win_rule_security_patches_up_to_date"
     selected="true">
   <xccdf:title>Security Patches Up-To-Date</xccdf:title>
   <xccdf:description>Keep systems up to current patch
levels</xccdf:description>
   <xccdf:check system="http://oval.mitre.org/XMLSchema/oval-definitions-5"
        multi-check="false">
```

```
      <xccdf:check-content-ref href="scap-windows-patches"
        name="oval:gov.nist.usgcb.win.patch:def:10101"/>
   </xccdf:check>
</xccdf:Rule>
```

3.2.4.4. CVSS and CCSS Scores

SCAP 1.0 required the inclusion of static CVSS scores in XCCDF vulnerability-related rules. However, CVSS base scores sometimes change over time (e.g., when more information is available about a particular vulnerability), and CVSS temporal and environmental scores are intended to change to reflect current threats, security controls, and other factors. As a result, the practice of embedding CVSS scores in XCCDF content was no longer required starting with SCAP 1.1.

During scoring, current CVSS scores acquired dynamically, such as from a data feed, SHOULD be used in place of the *@weight* attribute within XCCDF vulnerability-related rules. Section 3.8 contains additional requirements for CVSS usage.

CCSS scores are more stable than CVSS scores, but they still may change over time. Accordingly, during scoring, current CCSS scores acquired dynamically, such as from a data feed, MAY be used in place of the *@weight* attribute within XCCDF configuration setting-related rules. Section 3.9 contains additional requirements for CCSS usage.

For both the CVSS and CCSS cases, this specification encourages the use of data feeds that can be updated over time. The specifics around scoring provided in this and referenced sections are intended to prevent potential misuse of the XCCDF *@weight* attribute within an SCAP data stream.

Since the required CVSS version has been updated in SCAP 1.4 to CVSS v3, CVSS v3 scores SHOULD be used instead of CVSS v2 scores when a v3 score is available. This further supports the use of updatable data feeds to provide updated CVSS information. Unfortunately, XCCDF does not provide a means to indicate which CVSS version is used when calculating an XCCDF score. This is a recognized weakness in the XCCDF specification. As a result, tool developers are encouraged not to rely on the scoring information provided within an SCAP checklist.

3.2.5. The <xccdf:Value> Element

Use of the *<xccdf:source>*, *<xccdf:complex-value>*, and *<xccdf:complex-default>* elements within the *<xccdf:Value>* element SHALL NOT be allowed. Within the *<xccdf:choices>* element of the *<xccdf:Value>* element, use of the *<xccdf:complex-choice>* element SHALL NOT be allowed. Use of complex values is disallowed because the behavior for mapping XCCDF complex values to OVAL variables is not defined.

One or more *<xccdf:check-export>* elements MAY be used to define the binding of *<xccdf:Value>* elements to OVAL variables. The format of the *<xccdf:check-export>* element is:

```
   <xccdf:check-export value-id="XCCDF_Value_id"
      export-name="OVAL_External_Variable_id"/>
```

The following `<xccdf:check>` element example demonstrates the use of this convention:

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

The type and value binding of the specified *`<xccdf:Value>`* is constrained to match the lexical representation of the indicated OVAL Variable data type. Table 18 summarizes the constraints regarding data type usage.

Table 18. XCCDF-OVAL data export matching constraints

OVAL Variable Data Type	Matching XCCDF Data Type
int	number
float	number
boolean	boolean
string, evr_string, version, ios_version, fileset_revision, binary	string

Additional information regarding OVAL data types can be found in the OVAL language documentation and the XCCDF specification [XCCDF]. Additional information on OVAL data types may also be added to Sec. 4 of the SCAP 1.4 annex document [SP800-126A].

3.2.6. The <xccdf:Group> Element

The XCCDF group extension SHALL NOT be allowed.

3.3. Open Vulnerability and Assessment Language (OVAL)

This section lists requirements and recommendations for using the Open Vulnerability and Assessment Language (OVAL) to express an OVAL component of an SCAP source data stream (see Table 14).[8] Because SCAP 1.4 supports the use of multiple OVAL source data stream components, an SCAP content creator could choose to divide the OVAL definitions into multiple components. For example, a content creator could create one OVAL source data stream component that contains a set of OVAL definitions for one OVAL version (e.g., 5.10.1) and another set of OVAL definitions for a newer OVAL version (e.g., 5.12) if both versions are valid according to SCAP 1.4 requirements. SCAP 1.4 also supports multiple types of OVAL definitions within a single OVAL source data stream component. For example, a benchmark could

[8] See the SCAP 1.4 annex document [SP800-126A] for requirements regarding which OVAL versions shall or may be used in SCAP 1.4 content.

reference OVAL compliance and vulnerability definitions contained in a single data stream component.

The version of any particular OVAL document instance SHALL be specified using the *<oval:schema_version>* content element of the *<oval:generator>* element, as in this example:

```
   <oval:generator>
       <oval:product_name>The OVAL Repository</oval:product_name>
       <oval:schema_version>5.12</oval:schema_version>
     </oval:generator>
```

The versions that are specified using the *<oval:schema_version>* content element SHALL correspond to the versions specified by the *@xsi:schemaLocation* attribute value for the OVAL schema if an *@xsi:schemaLocation* attribute is specified.

Some OVAL interpreters make use of the OVAL variable format to allow variable values to be passed to the OVAL interpreter. While the OVAL variable format is not part of an SCAP data stream, this format can be used to carry variable information as part of an SCAP product for non-SCAP-related purposes.

If an *<oval-var:oval_variables>* element is used to carry variable values between an XCCDF processor and an OVAL processor, the *<oval:schema_version>* of the *<oval-var:oval_variables>* element SHALL be the same as that of the *<oval-def:oval_definitions>* element whose external variables are bound by the *<oval-var:oval_variables>* element.

Required values for the *@class* attribute of an OVAL definition are as follows:

1. "Compliance" if it represents a check for the system's configuration complying with policy requirements (e.g., having the required value for a specific configuration setting)
2. "Vulnerability" if it represents a check for the presence of a particular software flaw vulnerability on a system
3. "Patch" if it represents a check for whether a discrete patch needs to be installed on the system
4. "Inventory" if it represents a check for the presence of a product of interest on the system

The following requirements apply to particular classes of OVAL definitions:

1. For compliance class definitions:
    a. 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="http://cce.mitre.org"
    ref_id="CCE_identifier"/>
    ```

    The source attribute SHALL be defined using either "*http://cce.mitre.org*" (preferred method) or "CCE."

   b. Definitions that are directly or indirectly extended SHALL be limited to inventory and compliance classes.

2. For inventory class definitions:

   a. If an OVAL inventory class definition maps to one or more CPE identifiers, the definition SHOULD include *`<oval-def:reference>`* elements that reference those identifiers using the following format:

      ```
      <oval-def:reference source="http://cpe.mitre.org"
      ref_id="CPE_identifier"/>
      ```

      The source attribute SHALL be defined using either "*http://cpe.mitre.org"* (preferred method) or "CPE."

   b. Definitions that are directly or indirectly extended SHALL be limited to the inventory class.

3. For patch class definitions:

   a. If an OVAL patch class definition is associated with a source-specific identifier (e.g., Knowledge Base 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"/>
      ```

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

      ```
      <oval-def:reference source="http://cve.mitre.org"
      ref_id="CVE_identifier"/>
      ```

      This recommendation is weaker than its counterparts for the other class definition types because a CVE identifier is not an identifier for a patch; it is more of an association. For example, one patch could fix multiple vulnerabilities, so it would map to multiple CVE identifiers.

      The source attribute SHALL be defined using either "*http://cve.mitre.org"* (preferred method) or "CVE."

   c. Definitions that are directly or indirectly extended SHALL be limited to inventory and patch classes.

4. For vulnerability class definitions:

   a. If an OVAL 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="http://cve.mitre.org"
      ref_id="CVE_identifier"/>
      ```

The source attribute SHALL be defined using either "*http://cve.mitre.org*" (preferred method) or "CVE."

b. Definitions that are directly or indirectly extended SHALL be limited to inventory and vulnerability classes.

5. For miscellaneous class definitions, no additional requirements apply.

3.4. Open Checklist Interactive Language (OCIL)

This section lists recommendations for using the Open Checklist Interactive Language (OCIL) to express an OCIL component of an SCAP source data stream (see Table 14).

OCIL content SHOULD be used to check rules that cannot be fully automated with OVAL. For example, a particular software product may not have an application programming interface (API) that supports OVAL use. Another example is performing a check that requires user interaction, such as asking the user to look up information within a management console or to report a serial number affixed to a computing device. OCIL can also be used to collect a user's own information, such as whether the user participated in a recent security training session.

If an `<ocil:questionnaire>` element maps to one or more CCE, CVE, and/or CPE identifiers, it SHOULD include `<ocil:reference>` elements that reference those identifiers using the corresponding following format:

```
<ocil:reference href="http://cce.mitre.org">CCE_identifier</ocil:reference>

<ocil:reference href="http://cve.mitre.org">CVE_identifier</ocil:reference>

<ocil:reference href="http://cpe.mitre.org">CPE_identifier</ocil:reference>
```

3.5. Common Platform Enumeration (CPE)

This section lists requirements and recommendations for using Common Platform Enumeration (CPE) to express a CPE component of an SCAP source data stream (see Table 14).

The Official CPE Dictionary data feed[9] MAY be used by SCAP components to reference CPE names. If use of the Official CPE Dictionary is impractical, a subset of the dictionary MAY be used instead. Creating the reduced official dictionary involves first identifying every CPE in `<xccdf:platform>` and `<cpe2:fact-ref>` elements contained within referenced `<cpe2:platform-specification>` elements in every benchmark in the data stream. These CPEs SHALL then be matched against every entry in the Official CPE Dictionary using the CPE name matching algorithm [CPE-M]. All CPEs matched in the official dictionary with a result of EQUAL or SUPERSET SHALL be included in the reduced official dictionary.

One or more third-party dictionaries MAY be included in a data stream as well. All such third-party dictionaries SHOULD follow the requirements of the CPE Dictionary specification [CPE-D]. If including an entire third-party dictionary is impractical, a subset of the dictionary MAY be

[9] The Official CPE Dictionary is available at https://nvd.nist.gov/products/cpe.

used instead. The reduced dictionary SHALL be created using the same procedure outlined for creating a subset of the official dictionary. In all cases, a dictionary component MAY be remote to the data stream collection.

Each CPE name [CPE-N] in an *<xccdf:platform>* or *<cpe2:fact-ref>* element within an XCCDF document SHALL match at least one CPE entry in a dictionary referenced by the data stream. A match is considered an EQUAL or SUPERSET result when matching the CPE name to a dictionary entry, as defined in the CPE Name Matching specification [CPE-M]. Only non-deprecated names SHOULD be used. Checklist authors SHOULD ensure that each CPE name [CPE-N] they specify in an *<xccdf:platform>* or *<cpe2:fact-ref>* element within an XCCDF document has a check associated with its CPE name. If a corresponding check does not exist, then it will not be possible to fully detect the presence of the product and determine platform applicability.

[CPE-D] provides the defining structure of a CPE dictionary. A *<cpe2_dict:cpe-item>* element MAY contain one or more *<cpe2-dict:check>* elements that reference OVAL inventory class definitions. The referenced OVAL inventory class definition SHALL specify the technical procedure for determining whether or not a specific target asset is an instance of the CPE name specified by the *<cpe2_dict:cpe-item>* element. This usage is encouraged for CPE components. If a *<cpe2_dict:cpe-item>* element contained in a CPE component references an OVAL inventory class definition, then that definition SHALL be resolved by an *@href* attribute that references an OVAL source data stream component in the same data stream.

Because there may be a lag between when a new product is available and when the Official CPE Dictionary is updated to include a CPE name for that product, third-party dictionaries may be needed to compensate. When creating a subset of the Official CPE Dictionary or a third-party dictionary, a *<cpe2_dict:check>* element on an entry MAY be added or modified if the existing check does not provide satisfactory content to test the presence of the CPE name.

3.6. Common Configuration Enumeration (CCE)

To maintain consistency and accuracy, SCAP content that references a configuration setting SHALL use the official CCE identifier if a CCE entry for a particular configuration setting exists in the official CCE list. If no CCE entry exists for the configuration setting of interest, the content author SHOULD seek to have a CCE identifier issued for the configuration setting. See the OVAL compliance class definition requirements in Sec. 3.3 and the *<xccdf:ident>* requirements in Sec. 3.2.4.1 for additional requirements regarding CCE identifier references. The current official CCE list is available at https://nvd.nist.gov/config/cce/index, and new CCEs can be requested from NIST via email at cce@nist.gov.

Use of an official, dynamic data feed is preferred over the static coding of CCE-related supporting information in SCAP data sources. For example, NVD provides a data feed[10] that is the authoritative mapping between CCE identifiers and the control identifiers defined in SP 800-

[10] See https://nvd.nist.gov/config/cce.

53. Embedding control identifiers within SCAP content is strongly discouraged due to the maintenance burden that it imposes on content maintainers when the control identifiers are revised. A preferred technique is to only embed CCE identifiers in SCAP content. When mappings to SP 800-53 control identifiers are needed, dynamically acquire them from the official data feed, and associate them with the SCAP content based on its embedded CCE identifiers.

3.7. Common Vulnerabilities and Exposures (CVE)

CVE references in SCAP content MAY include both “candidate” and “entry” status identifiers. Deprecated CVE identifiers SHALL NOT be used.

If a CVE identifier exists for a particular vulnerability, the official CVE identifier SHALL be used. If no CVE exists for the software flaw, an alternate identifier MAY be used, but the user SHOULD seek to have a CVE identifier issued for the vulnerability. Information on submitting unpublished vulnerabilities and obtaining CVE identifiers is available at https://cve.org.

NIST also 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 https://nvd.nist.gov/vuln/data-feeds#CVE_FEED.

3.8. Common Vulnerability Scoring System (CVSS)

The NIST CVE data feed (see Sec. 3.7) is one source of CVSS base scores and vector data that MAY be used by products to support additional use cases built on SCAP and MAY be used by products along with temporal and environmental scores and vectors from other sources.

3.9. Common Configuration Scoring System (CCSS)

CCSS base, temporal, and environmental scores and vectors MAY be used by products. Adopters should consider CCSS data in the context of organizational security policies and dependencies among vulnerabilities. See [CCSS] for additional information.

3.10. XML Digital Signature

Digitally signing source data streams helps ensure the integrity and trustworthiness of legitimate content by preventing unauthorized content from being processed. Leveraging the TMSAD specification [TMSAD] for SCAP improves the legitimacy of authoritative content. Content authors SHOULD digitally sign SCAP source data stream collections in accordance with [TMSAD] and the requirements provided in this section.

3.10.1. Signature Location

If a digital signature is included in a source data stream collection, it SHALL be placed as the last element within the source data stream collection root element.

3.10.2. Signature Representation

Each digital signature SHALL be represented as a *<ds:Signature>* element and SHALL conform to the W3C XML Signature Syntax and Processing recommendation [DSIG]. The "*ds*" prefix is used in this section for clarity; any equivalent prefix bound to the XMLDSIG namespace MAY be used.

3.10.3. Signature Requirements

The *<ds:Signature>* element SHALL follow the recommendations in [TMSAD] and the following additional requirements:

a) Each *<ds:Signature>* SHALL cover exactly one target:

    1) The source data stream collection root element

    or

    2) A single data stream.

b) If the *<ds:Signature>* element targets the source data stream collection root element, the *<ds:SignedInfo>* element contained within SHALL contain exactly one *<ds:Reference>*. The *<ds:Reference>* SHALL have a target URI of one of the following:

    1) If the URI is an empty string otherwise represented as "", the URI target SHALL be considered to refer to the source data stream collection root element.

    2) If the URI is a defined ID, then it SHALL be a same-document reference (e.g., #id) that resolves to the source data stream collection root element.

c) If the *<ds:Signature>* element targets a single data stream, then the *<ds:SignedInfo>* element SHALL contain a same-document reference (e.g., #id) that resolves to the data stream collection being signed along with either:

    1) A set of same-document references (e.g., #id) that resolve to each local component referenced by the data stream being signed

    or

    2) A same-document reference (e.g., #id) that resolves to a *<ds:Manifest>* element contained within the *<ds:Signature>* as a *<ds:Object>* element. The referenced *<ds:Manifest>* element SHALL have a *<ds:Reference>* element for each local component referenced by the data stream being signed. External components MAY be omitted from the *<ds:Manifest>* element. Each *<ds:Reference>* element referencing a *<ds:component>* or *<ds:extended-component>* SHALL point to the component being signed by identifying the component in the *@URI* attribute using "#" and the *@Id* of the component.

3.10.4. Key Information

Cryptographic key information SHOULD be provided in the *<ds:Signature>* element through use of the *<ds:KeyInfo>* sub-element.

For example:

```
<scap-ds:source-data-stream-collection xml:id="sds-1" ...>
  <!-- data-streams and components -->
  <ds:Signature Id="sig-sds-1">
    <ds:SignedInfo>
      <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-
exc-c14n#"/>
      <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-
more#rsa-sha256"/>
      <ds:Reference URI="">
        <ds:Transforms>
          <ds:Transform
Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
        </ds:Transforms>
        <ds:DigestMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
        <ds:DigestValue>...</ds:DigestValue>
      </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>...</ds:SignatureValue>
    <ds:KeyInfo>
      <ds:X509Data>
        <ds:X509Certificate>...</ds:X509Certificate>
      </ds:X509Data>
    </ds:KeyInfo>
  </ds:Signature>
</scap-ds:source-data-stream-collection>
```

4. SCAP Content Processing Requirements and Recommendations

This section defines the processing requirements that SCAP content consumers SHOULD follow to correctly process SCAP 1.4 content. This section also provides recommendations that content producers and consumers are encouraged to adopt to promote stronger interoperability and greater consistency.

4.1. Legacy Support

Content consumers that support SCAP 1.4 may be interested in supporting earlier version of SCAP (i.e., SCAP 1.3) if there is little difference between the two content streams.[11] Content consumers that process legacy SCAP content SHOULD be capable of outputting results in the current SCAP revision. Additionally, content consumers MAY output results in the same SCAP version as the source content. For producers of results in legacy formats, legacy results MAY also be converted into results based on the current SCAP revision.

Certain constructs can be deprecated in the SCAP component specifications.[12] SCAP content consumers SHOULD support deprecated constructs unless specifically noted in the annex because they are still valid in SCAP 1.4 and supported legacy SCAP versions. This recommendation ensures that legacy SCAP content that uses these deprecated constructs will continue to be supported.

Content consumers that support OVAL SHOULD support OVAL definition documents written against all versions of OVAL component specifications listed in the annex.

4.2. Source Data Streams

Content consumers SHALL be capable of validating SCAP content against the appropriate schemas and Schematron stylesheets, detecting and reporting errors, and failing gracefully if there are errors. The relevant XML schemas are available at https://scap.nist.gov/revision/1.4/#schema, and the relevant Schematron rule sets are available at https://scap.nist.gov/revision/1.4/#schematron. See Sec. 3.1 for additional information on the Schematron rule sets.

Content consumers SHOULD validate XML digital signatures if they exist in the content. Validating a signature includes confirming that the signature value is valid, all of the reference hashes in the signature and manifest are correct, and the public key used to verify the signature is from a trusted source. A data stream with a signature that does not validate SHOULD NOT be evaluated by a content consumer.

When a *<ds:extended-component>* that is not recognized by the tool is referenced from a *<ds:data-stream>*, *<ds:component>*, or *<ds:extended-component>* element, the tool SHALL issue a warning.

[11] See https://csrc.nist.gov/pubs/sp/800/126/r2/final (SCAP 1.2) and https://csrc.nist.gov/pubs/sp/800/126/r3/final (SCAP 1.3).
[12] The OVAL Language Deprecation policy is available at https://oval-community-guidelines.readthedocs.io/en/base/deprecation.html.

If more than one *<ds:data-stream>* element is specified on the *<ds:data-stream-collection>*, the ID of the *<ds:data-stream>* to execute SHALL be indicated to the content consumer, and the content consumer SHALL use the specified *<ds:data-stream>*. If more than one *<xccdf:Benchmark>* is referenced by a *<ds:data-stream>*, the ID of the *<xccdf:Benchmark>* to execute SHALL be indicated to the content consumer, and the content consumer SHALL process the indicated *<xccdf:Benchmark>*. Because SCAP and its component specifications do not formally define how to designate a particular data stream or benchmark, it is expected that products will implement these capabilities in a proprietary way.

4.3. XCCDF Processing

The following requirements and recommendations pertain to content consumers processing XCCDF benchmark and tailoring components from an SCAP source data stream.

4.3.1. CPE Applicability Processing

CPEs referenced in an *<xccdf:platform>* element directly or by a *<cpe2:fact-ref>* contained within a referenced *<cpe2:platform-specification>* element SHALL be evaluated as follows to determine their presence on a machine:

1. The CPE SHALL be matched against all CPEs in all of the dictionaries referenced by the *<ds:data-stream>* element. All CPEs that return an EQUAL or SUPERSET result as defined in CPE Name Matching [CPE-M] SHALL be used in evaluating the *<xccdf:platform>* or *<cpe2:fact-ref>*.
2. Either a list of CPEs found on the target asset SHALL be known before the scan, or a list SHALL be generated. If a previously known list is used, it SHALL be equivalent to a newly generated list. To generate the list, the *<cpe2_dict:check>* element data associated with the found *<cpe2_dict:cpe-item>* elements SHALL be evaluated against the target using the referenced OVAL inventory class definition. If a *<cpe2_dict:check>* returns “pass,” then the corresponding CPE SHALL be added to the list of CPEs found on the target.
3. The list of CPEs found on the target asset and either the *<xccdf:platform>* or the *<cpe2:platform-specification>* SHALL be used as input to the CPE Applicability Language [CPE-L] algorithm to determine the XCCDF benchmark applicability to the target asset.

4.3.2. Checking System Usage

If an XCCDF component has multiple *<xccdf:check-content-ref>* elements, then check processing SHOULD be performed according to [XCCDF:7.2.3.5.1] with the following changes:

1. For each *<xccdf:check-content-ref>* element, a content consumer either SHALL attempt to retrieve the document referenced by the *<ds:component-ref>* element that is referenced directly by the *<xccdf:check-content-ref>* element’s *@href*

attribute, or it SHALL resolve the *@href* attribute within the context of the XML Catalog specified as part of the *<ds:component-ref>* element used to reference this benchmark. If not resolvable, the next available *<xccdf:check-content-ref>* element SHALL be evaluated. If none of the *<xccdf:check-content-ref>* elements are resolvable, then the result of the rule evaluation SHALL be the XCCDF "notchecked" status, and processing of the check SHALL end.

2. Once a resolvable *<xccdf:check-content-ref>* element is found, then checking system processing SHALL proceed. When evaluating a rule, an *<xccdf:rule-result/xccdf:message>* with the *@severity* attribute value of "info" SHALL be generated, indicating the *<xccdf:check-content-ref>* *@href* attribute and *@name* attribute, if provided.

Content consumers SHOULD implement checking systems that are supported by SCAP, as defined in Sec. 3.2.4.2. Content consumers MAY implement checking systems that are not supported by SCAP. If a tool encounters a checking system it does not support, it SHOULD issue a warning, and it SHOULD continue processing according to the [XCCDF] specification.

4.4. SCAP Result Data Streams

An SCAP result data stream contains the results of the evaluation of one or more SCAP source data streams by an SCAP content consumer. The following requirements and recommendations pertain to content consumers that generate SCAP result data streams.

An SCAP result data stream SHALL conform to the [ARF] specification. The following sections outline the details of the ARF report. In all situations, one or more component results (e.g., XCCDF, check results), the target asset, and/or the SCAP source data stream collection represented as a report request in ARF MAY be represented either as a local component in the ARF or as a remote resource, leveraging the remote resource capability built into ARF. The following is a stripped down ARF example:

```
<arf:asset-report-collection>
   <rc:relationships>
      <rc:relationship type="arf-rel:isAbout" subject="xccdf1">
         <rc:ref>asset1</rc:ref>
      </rc:relationship>
      <rc:relationship type="arf-rel:isAbout" subject="oval1">
         <rc:ref>asset1</rc:ref>
      </rc:relationship>
      <rc:relationship type="scap-rel:checkContext" subject="oval1">
         <rc:ref>xccdf1</rc:ref>
      </rc:relationship>
      <rc:relationship type="scap-rel:fromSource" subject="xccdf1">
         <rc:ref>collection1</rc:ref>
      </rc:relationship>
      <rc:relationship type="scap-rel:fromSource" subject="oval1">
         <rc:ref>collection1</rc:ref>
      </rc:relationship>
   </rc:relationships>
   <arf:report-requests>
      <arf:report-request id="collection1">
```

```
         <arf:content>
            <ds:data-stream-collection>…</ds:data-stream-collection>
         </arf:content>
      </arf:report-request>
   </arf:report-requests>
   <arf:assets>
      <arf:asset id="asset1">
         <ai:computing-device>…</ai:computing-device>
      </arf:asset>
   </arf:assets>
   <arf:reports>
      <arf:report id="xccdf1">
         <arf:content>
            <xccdf:TestResult>…</xccdf:TestResult>
         </arf:content>
      </arf:report>
      <arf:report id="oval1">
         <arf:content>
            <oval-res:oval_results>…</oval-res:oval_results>
         </arf:content>
      </arf:report>
   </arf:reports>
</arf:asset-report-collection>
```

4.4.1. The Component Reports

The ARF report SHALL contain a report object for each XCCDF, OVAL, and OCIL component executed when a source data stream is evaluated against a target. It MAY contain additional report objects for other results, such as *<oval-var:oval_variables>* or extended component results. Each component result SHALL be captured as a separate *<arf:report>* element in the *<arf:asset-report-collection>* element.[13] When reporting on XCCDF, OVAL, or OCIL, each component report SHALL use the element specified in Table 19 as its root element.

Table 19. SCAP result data stream component document elements

Component	Document Element
XCCDF	`<xccdf:TestResult>`
OVAL	`<oval-res:oval_results>`
OCIL	`<ocil:ocil>`

SCAP result data stream components SHOULD NOT use any deprecated constructs in their associated specifications. Validation of each component SHALL be done in accordance with the portions of this document that define requirements for the component. See Sec. 3.1.2 for more information on the SCAP Content Validation Tool, which can help validate the correctness of SCAP result data streams.

[13] For example, if two check components were executed, one referenced by a `<ds:component-ref>` element of a `<ds:dictionaries>` element and another one referenced by a `<ds:component-ref>` element of a `<ds:checklists>` element, the ARF report will include two separate `<arf:report>` elements, one for each executed component.

4.4.2. The Target Identification

The target asset SHALL be represented in the ARF report using the *<ai:assets>* part of ARF. The *<ai:asset>* element populated about a target asset SHOULD include the fields specified in Table 20, where applicable.

Table 20. Asset identification fields to populate

Field	Location Within Asset Identification Computing Device
Ethernet media access control address	connections/connection/mac-address
Internet Protocol version 4 address	connections/connection/ip-address/ip-v4
Internet Protocol version 6 address	connections/connection/ip-address/ip-v6
Host name	hostname
Fully qualified domain name	fqdn

Additional identification information MAY be captured in the *<ai:asset>* element (e.g., asset tag, system GUID). The guidelines specified in [AI] SHALL be followed when populating the asset identification information. Currently, only the target asset of the SCAP evaluation is identified.

4.4.3. The Source Data Stream

The source data stream collection that was used to generate the results against the target SHOULD be included in the ARF report as an *<arf:report-request>*. If the source data stream collection is included in the ARF report, and an *<xccdf:Tailoring>* component was used during processing, the tailoring component SHALL be included as well. The following is a stripped-down example:

```
<arf:asset-report-collection>
    <arf:report-requests>
        <arf:report-request id="request_0">
            <arf:content>
                <ds:data-stream-collection id="..."
                    <!-- Source data stream collection which was tailored -->
                    ...
                </ds:data-stream-collection>
            </arf:content>
        </arf:report-request>

        <arf:report-request id="request_1">
            <arf:content>
                <ds:data-stream-collection id="..."
                    <!-- Source data stream collection with an <xccdf:Tailoring>
component -->
                    ...
                </ds:data-stream-collection>
            </arf:content>
        </arf:report-request>
    </arf:report-requests>
    <arf:assets>...</arf:assets>
    <arf:reports>...</arf:reports>
</arf:asset-report-collection>
```

4.4.4. The Relationships

Table 21 outlines the relationships that SHALL be specified in the ARF report if the stated condition is satisfied.

Table 21. ARF relationships

Relationship	Condition	Cardinality	Definition	Subject	Object
arf-rel:isAbout	None	One for each component report	Each report is reporting about the asset	Component report	Target asset
scap-rel:checkContext	Benchmark report exists	One for each check component report (OVAL or OCIL)	Each check report is reporting in the context of the benchmark report	Check component report	Benchmark component report
scap-rel:fromSource	Report request exists	One for each component report	Each component report was generated from the SCAP source content	Component report	Report request
scap-rel:associatedWith	OVAL variables report is provided	One for each OVAL variables component report	Each OVAL variables report is associated with an OVAL result	Component report of OVAL variables	Component report of OVAL results

Figure 3 gives an example of how the resulting ARF report would look.

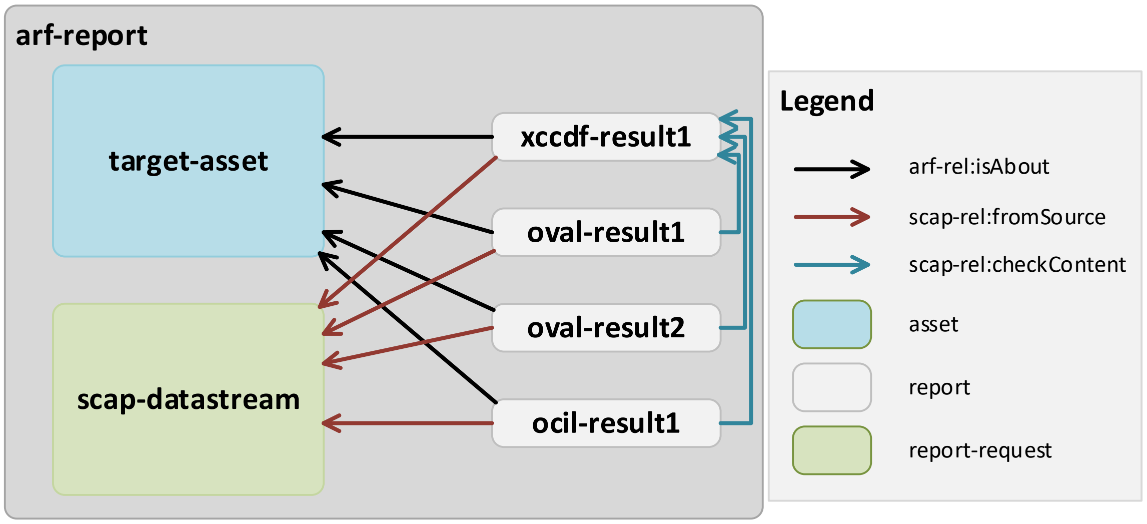

Fig. 3. Sample ARF report structure

4.5. XCCDF Results

The following requirements and recommendations pertain to content consumers that generate XCCDF result data stream components.

Each XCCDF result data stream component SHALL comply with the XCCDF Results schema.

XCCDF test results SHALL be documented as the contents of an *<xccdf:TestResult>* element. To be considered valid SCAP result content, the *<xccdf:TestResult>* element SHALL meet the following conditions:

1. The *@start-time* and *@end-time* attributes SHALL be provided to indicate when the scan started and completed, respectively.
2. The *@test-system* attribute SHALL be provided, and it SHALL be a CPE name value indicating the product that was responsible for generating the results.
3. When the *<xccdf:TestResult>* is the root XCCDF element, then it will include an *<xccdf:benchmark>* element [XCCDF:6.6.2].
    a. The *<xccdf:benchmark>* element SHALL have an *@id* attribute specified. The *@id* attribute SHALL match the value of the *<xccdf:Benchmark>* element's *@id* attribute that was processed.
    b. The *<xccdf:benchmark>* element SHALL have an *@href* attribute specified. The *@href* attribute SHALL hold the URI that references the XCCDF component (i.e., either local to the data stream collection or remote) that was processed. The URI SHALL be in the form specified for the *@href* attribute in Table 8.
4. If a child profile of an *<xccdf:Tailoring>* element was applied during processing, then the *<xccdf:tailoring-file>* element SHALL be present and SHALL provide the following information about the <*xccdf:Tailoring>* element: *@href*, *@id,* *@version*, and *@time*. The *@href* attribute SHALL hold the URI to the XCCDF tailoring component and SHALL comply with the format described above (item 3).
5. The *<xccdf:Profile>* element SHALL be included if a profile was applied during processing. This is also applicable to selected profiles that are part of *<xccdf:Tailoring>*.
6. Regarding the definition and use of *<xccdf:Profile>* elements, reported *<xccdf:set-value>* elements SHALL include all those values that are exported by the reported rules. The specific settings are those determined by the reported *<xccdf:Profile>*.
7. The *<xccdf:identity>* element SHALL identify the security principal used to access rule evaluations on the targets. This will include the identity name or username used to perform the evaluation.
8. Each IP addresses associated with the *<xccdf:target>* SHALL be enumerated using the *<xccdf:target-address>* element.
9. An *<xccdf:target-id-ref>* SHALL be specified with a *@system* attribute of "http://scap.nist.gov/schema/asset-identification/1.1," an *@href* attribute value of "", and a *@name* attribute value of the ID of the *<ai:asset>* element in the ARF that this *<xccdf:TestResult>* is about.
10. The *<xccdf:rule-result>* elements report the result of the application of each selected rule [XCCDF:6.6.2]. The *@role*, *@severity*, and *@weight* attributes of the

<xccdf:rule-result> element SHALL be provided to indicate the values used during assessment. The *<xccdf:check/xccdf:check-content-ref>* element SHALL record the reference to the checking system-specific result component report ID and check name within the result file using the *@href* and *@name* attributes, respectively. The *@href* attribute SHALL contain "#" and the *@id* of the *<arf:report>* that contains the check result. This approach provides traceability between XCCDF and check results. If *@multi-check* is not set to "true," and the *<xccdf:rule-result>* represents a group of checks, then the *@name* attribute SHALL be omitted. See the example below the next requirement.

11. Where applicable to the target system, each of the *<xccdf:fact>* elements in Table 22 SHALL be provided. Previous versions of SCAP required additional facts; these have been incorporated into the use of the Asset Identification specification, as discussed in Sec. 4.4.2.

Table 22. XCCDF fact descriptions

XCCDF Fact	Description of Use
`urn:scap:fact:asset:identifier:ein`	Equipment identification number or other inventory tag number
`urn:scap:fact:asset:identifier:guid`	Globally unique identifier for the asset, if assigned
`urn:scap:fact:asset:environmental_information:owning_organization`	Organization that tracks the asset on its inventory
`urn:scap:fact:asset:environmental_information:current_region`	Geographic region where the asset is located
`urn:scap:fact:asset:environmental_information:administration_unit`	Name of the organization that does system administration for the asset

The following is a stripped-down example that illustrates the above requirements:

```
<arf:asset-report-collection>
    <rc:relationships>...</rc:relationships>
    <arf:report-requests>...</arf:report-requests>
    <arf:assets>...</arf:assets>
    <arf:reports>
        <arf:report id="scap_gov.nist_comp_r3005-xccdf_01">
            <arf:content>
                <xccdf:TestResult start-time="2016-03-10T10:07:11" version="1-2.1.0.0"
test-system="cpe:/a:vendor:product_name:version"
                    end-time="2016-03-10T10:07:11"
                    id="xccdf_gov.nist_testresult_...">
                    <xccdf:benchmark href="file:r3005-datastream-
01.xml#scap_gov.nist_comp_r3005-xccdf_01" id="xccdf_gov.nist_benchmark_r3005_id_01"/>

                    <xccdf:tailoring-file href="#scap_gov.nist_comp_r3005-
xccdf_tailoring_03" id="xccdf_gov.nist_tailoring_r3005_03" time="2016-01-22T14:00:00"
version="1-2.1.0.0"/>

                    <xccdf:organization>...</xccdf:organization>
                    <xccdf:identity privileged="true"
authenticated="true">...</xccdf:identity>
```

```
                    <xccdf:profile
idref="xccdf_gov.nist.validation_profile_r3005_tailoring_03"/>
                    <xccdf:target>...</xccdf:target>
                    <xccdf:target-address>...</xccdf:target-address>
                    <xccdf:target-facts>...</xccdf:target-facts>
                    <xccdf:target-id-ref system="http://scap.nist.gov/schema/asset-
identification/1.1" href="" name="..."/>
                    <xccdf:set-value
idref="xccdf_gov.nist_value_validation.r3005_for_rule_6">test0</xccdf:set-value>
                    ...
                    <xccdf:rule-result time="2016-03-10T10:07:11"
idref="xccdf_gov.nist_rule_validation.r3005_rule_1" weight="10" severity="medium"
role="full">
                        <xccdf:result>pass</xccdf:result>
                        <xccdf:check system="http://oval.mitre.org/XMLSchema/oval-
definitions-5" selector="sel1">
                            <xccdf:check-content-ref href="#scap_gov.nist_comp_r3005-
oval" name="oval:nist.validation.r3005:def:2"/>
                        </xccdf:check>
                    </xccdf:rule-result>
                </xccdf:TestResult>

            </arf:content>
        </arf:report>

        <arf:report id="scap_gov.nist_comp_r3005-oval">
            <arf:content>
                <arf:oval-res:oval_results>...</oval-res:oval_results>
            </arf:content>
        </arf:report>
        ...
    </arf:reports>
</arf:asset-report-collection>
```

4.5.1. Assigning Identifiers to Rule Results

The *<xccdf:rule-result>* element provides data that indicates the result of assessing a system using the identified *<xccdf:Rule>* element. If the target *<xccdf:Rule>* identified by the <xccdf:*rule-result>* element's *@idref* attribute has one or more *<xccdf:ident>* elements with a *@system* attribute value listed in Sec. 3.2.4.1, then each *<xccdf:ident>* element SHALL also appear within the *<xccdf:rule-result>* element.

If the *<xccdf:ident>* element is included, it is important for tracking purposes that produced XCCDF results have specific meanings. If an *<xccdf:ident>* element is present, and it identifies a CVE, CCE, or CPE entry, then an *<xccdf:rule-result>* of "pass" SHALL indicate that the check content evaluated within the rule complied with one of the following:

- For a CVE entry, the target platform satisfies all of the conditions of the XCCDF rule and is unaffected by the vulnerability or exposure referenced by the CVE.
- For a CCE entry, the target platform complies with the configuration-setting guidance expressed in the XCCDF rule.
- For a CPE entry, the target platform was identified on the system.

These interpretations of `<xccdf:ident>` elements must be preserved. For example, consider two policy recommendations. One is that a particular piece of software be installed, and the second is that another piece of software should not be installed. Both rules for these policy recommendations could use the same CPE entry in their `<xccdf:ident>` elements. However, because the interpretation of a CPE entry is that a "pass" result indicates that software was installed, the second policy recommendation's rule would violate this. This can be corrected by using the `@con:negate` attribute, which is a Boolean attribute that inverts the rule result. The second rule could check for the software being installed and then negate that result, thus giving a result that is consistent in meaning with the first rule. For rules that cannot have their interpretations preserved through the use of the `@con:negate` attribute, an alternative is to have a CCE entry correspond to the recommendation. Rules that do not use `<xccdf:ident>` elements have no such restrictions.

4.5.2. Mapping OVAL Results to XCCDF Results

When evaluating an `<xccdf:Rule>` element that references an OVAL definition, the `<xccdf:rule-result>` element SHALL be used to capture the result of this evaluation. This result SHALL be determined by evaluating the referenced OVAL definition on a target host. The resulting value of an individual `<xccdf:check>` SHALL be mapped from the OVAL definition result produced during evaluation. The corresponding `<xccdf:rule-result/xccdf:result>` value is then computed based on the result values of all relevant `<xccdf:check>` elements.[14] While the OVAL specification permits limiting result status reporting, SCAP-conformant content SHALL provide full status reporting, including error, unknown, not applicable, not evaluated, true, and false.

Content consumers SHALL apply the mapping illustrated in Table 23 when deriving `<xccdf:check>` results from OVAL definition processing.

Table 23. Deriving XCCDF check results from OVAL definition results

OVAL Definition Result		XCCDF Check Result (@negate is set to "false")	XCCDF Check Result (@negate is set to "true")
error		error	error
unknown		unknown	unknown
not applicable		notapplicable	notapplicable
not evaluated		notchecked	notchecked
Definition Class	**Definition Result**	pass	fail
compliance	true		
vulnerability	false		
inventory	true		
patch	false		

[14] Normally, only a single `<xccdf:check>` element is needed. However, if an `<xccdf:complex-check>` element is used, there may be multiple results that must be combined, as outlined in the XCCDF specification.

OVAL Definition Result		XCCDF Check Result (@negate is set to “false”)	XCCDF Check Result (@negate is set to “true”)
Definition Class	**Definition Result**		
compliance	False	fail	pass
vulnerability	true		
inventory	false		
patch	true		

The corresponding result value SHALL be recorded based on the `@class` attribute of the OVAL definition and the `@negate` attribute of the `<xccdf:check>` element, where applicable.

The mappings in Table 23 are specific to each OVAL definition class. For example, if an OVAL compliance class definition is processed, and OVAL returns a result of “true,” the content consumer is conveying the fact that the system was found to be compliant with that check and therefore returns a “pass” result for that check. 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 check. Negations of check results or their combination in complex checks may result in additional modification before the final corresponding `<xccdf:rule-result/xccdf:result>` value is known.

If the `<xccdf:Rule>` element under evaluation has an `<xccdf:check-content-ref>` element with the `@name` attribute omitted and an `<xccdf:check>` element with its `@multi-check` attribute set to “true,” then the result of each evaluated OVAL definition SHALL be recorded as a separate `<xccdf:rule-result>` element. In this case, the `<xccdf:rule-result>/<xccdf:check-content-ref>` element SHALL identify the specific check result of each evaluated OVAL definition using the `@href` and `@name` attributes, as described in Sec. 4.5, item 8.

According to [XCCDF:Table 9;Table 35;Table 39], if the `<xccdf:Rule>` element under evaluation is selected, and its `@role` attribute is set to ”unchecked,” then the rule result SHALL be set to “notchecked.” If the `<xccdf:Rule>` element under evaluation is selected, and its `@role` attribute is set to “unscored,” then the rule result SHALL be set to “informational.”

4.6. OVAL Results

The following requirements and recommendations pertain to content consumers that generate OVAL result data stream components. See the annex for additional requirements and recommendations.

Each OVAL result data stream component SHALL validate against at least one version of the OVAL results schema that corresponds to an OVAL component specification version that was specified in Sect. 2 of the annex, regardless of the version of the OVAL definitions document that was evaluated.

An SCAP OVAL result data stream component SHALL include the results of every OVAL definition used to generate the reported results.

In order to be SCAP-conformant, an SCAP content consumer SHALL be able to produce all types of OVAL result outputs described below. The specific result output SHALL be configurable within the SCAP content consumer.

In order to support SCAP instances where OVAL thin content (i.e., only the ID of the definition and the results) is preferred, SCAP content consumers SHALL support all valid values for the *<oval-res:directives>* that control the expected content of the results file.

To support the ability for results to be consumed by appropriate products, data results SHALL be expressed as a single machine without system characteristics, single machine with system characteristics, or single machine with thin results as follows:

1. Single machine without system characteristics — A single result file that includes the results of all OVAL definitions evaluated and “full” result types, as described in the *<oval-res:ContentEnumeration>* element, without system characteristics.

   For this format, the values for the *<oval-res:directives>* element SHALL be:

   ```
   <oval-res:directives include_source_definitions="false">
      <oval-res:definition_true content="full" reported="true"/>
      <oval-res:definition_false content="full" reported="true"/>
      <oval-res:definition_unknown content="full" reported="true"/>
      <oval-res:definition_error content="full" reported="true"/>
      <oval-res:definition_not_evaluated content="full" reported="true"/>
      <oval-res:definition_not_applicable content="full" reported="true"/>
   </oval-res:directives>
   ```

   When creating the OVAL system characteristics, as defined by the *<oval-sc:oval_system_characteristics>* element, the *<oval-sc:collected_objects>* and *<oval-sc:system_data>* elements SHALL NOT be provided.

2. Single machine with system characteristics — A single result file that includes the results of all OVAL definitions evaluated and “full” result types, as described in the *<oval-res:ContentEnumeration>* element and the system characteristics of the target evaluated.

   For this format, the values for the *<oval-res:directives>* element SHALL be:

   ```
   <oval-res:directives include_source_definitions="false">
      <oval-res:definition_true content="full" reported="true"/>
      <oval-res:definition_false content="full" reported="true"/>
      <oval-res:definition_unknown content="full" reported="true"/>
      <oval-res:definition_error content="full" reported="true"/>
      <oval-res:definition_not_evaluated content="full" reported="true"/>
      <oval-res:definition_not_applicable content="full" reported="true"/>
   </oval-res:directives>
   ```

   When creating the OVAL system characteristics, as defined by the *<oval-sc:oval_system_characteristics>* element, the *<oval-sc:collected_objects>* and *<oval-sc:system_data>* elements SHALL be provided.

3. Single machine with thin results — A single result file that includes the results of all OVAL definitions evaluated and “thin” result types, as described in the OVAL results

schema. A value of "thin" means that the minimal amount of information will be provided.

For this format, the values for the *<oval-res:directives>* element SHALL be:

```
<oval-res:directives include_source_definitions="false">
   <oval-res:definition_true content="thin" reported="true"/>
   <oval-res:definition_false content="thin" reported="true"/>
   <oval-res:definition_unknown content="thin" reported="true"/>
   <oval-res:definition_error content="thin" reported="true"/>
   <oval-res:definition_not_evaluated content="thin" reported="true"/>
   <oval-res:definition_not_applicable content="thin" reported="true"/>
</oval-res:directives>
```

When specifying OVAL system characteristics, a reference SHOULD be made to the target asset in the ARF report collection. Specifically, the *<oval-sc:oval_system_characteristics>/<oval-sc:system_info>* SHOULD be populated with a *<con:asset-identification>* element. That element SHALL be populated with a single *<arf:object-ref>* element that points to the *<ai:asset>* element in the ARF report collection pertaining to the OVAL result. See [ARF] for details on populating the *<arf:object-ref>* element.

4.7. OCIL Results

The following requirements and recommendations pertain to content consumers that generate OCIL result data stream components.

An SCAP OCIL result data stream component SHALL include the results of every *<ocil:questionnaire>*, *<ocil:question_test_action>*, and *<ocil:question>* element used to generate the reported results.

4.8. Result Data Stream Signing

Digitally signing SCAP result content is important for establishing integrity and provenance and for enabling content consumers to make trust decisions in accordance with [TMSAD]. Content consumers SHOULD digitally sign SCAP result content in accordance with [TMSAD] and the requirements in this section.

4.8.1. Signature Location

If a digital signature is included within a source data stream collection, it SHALL be placed in an *<arf:extended-info>* element within the ARF report.

4.8.2. Signature Representation

Each digital signature SHALL be represented as a *<ds:Signature>* element and SHALL conform to the W3C XML Signature Syntax and Processing recommendation [DSIG]. The "*ds*" prefix is used in this section for clarity; any equivalent prefix bound to the XMLDSIG namespace MAY be used.

4.8.3. Signature Requirements

The *<ds:Signature>* element SHALL follow the recommendations in [TMSAD] along with the following additional requirements.

a) The *<ds:Signature>* SHALL cover exactly one target: the *<arf:asset-report-collection>* element (i.e., the ARF report).

b) The first *<ds:Reference>* element in a *<ds:Signature>* element SHALL be to the *<arf:asset-report-collection>* element. The element SHALL be referenced in the *@URI* attribute using the empty string convention "".

c) Two XPath Filter 2 transforms SHALL exist on the first *<dsig:Reference>* element in a *<dsig:Signature>* element. Both SHALL specify a filter type of "subtract." The first transform SHALL specify the XPath */arf:asset-report-collection/arf:extended-infos[count(arf:extended-info[dsig:Signature]) = count(*)]*. The second transform SHALL specify the XPath */arf:asset-report-collection/arf:extended-infos/arf:extended-info[dsig:Signature]*. In both cases, the namespace prefix "arf" SHALL map to the ARF namespace specified in this document.

d) The second *<dsig:Reference>* element MAY be to the *<dsig:SignatureProperties>* element captured in a *<dsig:Object>* element with the *<dsig:Signature>* element. The *<dsig:SignatureProperties>* element SHALL be referenced in the *@URI* attribute using "#" and the *@Id* of the *<dsig:SignatureProperties>* element.

4.8.4. Key information

Key information SHOULD be provided on the *<dsig:Signature>* element.

4.8.5. Countersigning

If countersigning the result is desired (e.g., a system signs automatically, and a human reviewer later adds a signature), the following SHALL apply:

a) The *<arf:extended-info>* element that contains the original signature SHALL be removed from the resulting document.

b) The original signature SHALL be captured as a *<dsig:Object>* element on the new *<dsig:Signature>* element.

c) The first *<dsig:Reference>* element in the new *<dsig:Signature>* element SHALL reference the *<dsig:Object>* element that contains the original signature. The

<dsig:Object> element SHALL be referenced in the *@URI* attribute using "#" and the *@Id* of the *<dsig:Object>* element.

d) If the *<dsig:SignatureProperties>* element existed within the original signature, the second *<dsig:Reference>* element SHALL be to the *<dsig:SignatureProperties>* element captured in a *<dsig:Object>* element with the *<dsig:Signature>* element. The *<dsig:SignatureProperties>* element SHALL be referenced in the *@URI* attribute using "#" and the *@Id* of the *<dsig:SignatureProperties>* element.

e) A *<dsig:SignatureProperties>* element MAY be included in the *<dsig:Signature>* element. At least one *<dsig:SignatureProperty>* element MAY be populated with *<dt:signature-info>*, as specified in [TMSAD].

f) Key information SHOULD be provided on the *<dsig:Signature>* element in accordance with [TMSAD].

g) The new *<dsig:Signature>* element SHALL be placed in a new *<arf:extended-info>* element in the ARF report collection.

A signature that has countersigned another signature (also known as an enveloping signature) MAY be countersigned. When doing so, the requirements above SHALL apply to the new signature creation.

5. Source Data Stream Content Requirements for Use Cases

This section discusses additional requirements for the following SCAP-conformant content use cases: compliance checking, vulnerability scanning, and inventory scanning. As stated in Table 3, each data stream is required to have a *`@use-case`* attribute in its *`<ds:data-stream>`* element with a value that corresponds to either one of the content types defined in this section or to "OTHER" for data streams that do not correspond to a defined use case. The required value for each content type is specified in the following subsection.

Each use case is subject to the requirements presented in this section as well as all applicable requirements in Sec. 3 and 4.

5.1. Compliance Checking

SCAP content can be used to compare system characteristics and settings against an SCAP-conformant checklist in an automated fashion. This can verify that operating systems and applications comply with security checklists and identify any deviations from those checklists.

The SCAP source data stream component that SHALL be included for compliance checking is the XCCDF benchmark, which expresses the checklist. Each rule in the XCCDF benchmark SHALL reference one of the following:

- **An OVAL compliance definition.** This definition SHALL be contained in an OVAL component, which holds definitions of compliance checks used by the checklist. An XCCDF benchmark's rules MAY reference one or more OVAL compliance class definitions in an OVAL component.
- **An OCIL questionnaire.** This questionnaire SHALL be contained in an OCIL component, which holds questionnaires that collect information that OVAL is not being used to collect, such as posing questions to users or harvesting configuration information from an existing database. An XCCDF benchmark's rules MAY reference one or more OCIL questionnaires in an OCIL component.
- **An OVAL patch definition.** This definition SHALL be contained in an OVAL component, which holds definitions for patch compliance checks. These checks may be needed if an organization includes patch verification in its compliance activities. An XCCDF benchmark MAY reference an OVAL patch definition through a patches up-to-date rule in a manner consistent with Sec. 3.2.4.3.

Each XCCDF benchmark SHALL have at least one rule that references either an OVAL compliance class definition in an OVAL component or an OCIL questionnaire in an OCIL component.

All OVAL components and OCIL components referenced by the XCCDF benchmark SHALL be included in the SCAP source data stream.

If the XCCDF benchmark component references any CPE names, then the SCAP source data stream SHALL include a CPE component, which specifies the products or platforms of interest, and SHALL include one or more OVAL inventory class definitions in an OVAL component that contain the technical procedures for determining whether or not a specific target asset has a product or platform of interest.

The `@use-case` attribute in the `<ds:data-stream>` element SHALL be set to "CONFIGURATION."

5.2. Vulnerability Scanning

SCAP content can be used to scan operating systems and applications to look for known software flaws that introduce security exposures. The content enables the consistent detection and reporting of these flaws.

The SCAP source data stream component that SHALL be included for vulnerability scanning is the XCCDF benchmark, which expresses the checklist of the flaws to be checked for. Each rule in the XCCDF benchmark SHALL reference one of the following:

- **An OVAL vulnerability definition.** This definition SHALL be contained in an OVAL component, which holds definitions of vulnerability checks used by the checklist. An XCCDF benchmark's rules MAY reference one or more OVAL vulnerability class definitions in an OVAL component.
- **An OCIL questionnaire.** This questionnaire SHALL be contained in an OCIL component, which holds questionnaires that collect information that OVAL is not being used to collect. An example of OCIL use is to give step-by-step directions for manually examining a system for a vulnerability that cannot be detected with OVAL. In such a case, OCIL is used for capturing information collected using manual examination. An XCCDF benchmark's rules MAY reference one or more OCIL questionnaires in an OCIL component.
- **An OVAL patch definition.** This definition SHALL be contained in an OVAL component, which holds definitions for patch compliance checks. These checks may be needed if an organization includes patch verification in its vulnerability scanning activities. An XCCDF benchmark MAY reference an OVAL patch definition through a patches up-to-date rule in a manner consistent with Sec. 3.2.4.3.

Each XCCDF benchmark SHALL have at least one rule that references either an OVAL vulnerability class definition in an OVAL component or an OCIL questionnaire in an OCIL component.

All OVAL components and OCIL components referenced by the XCCDF benchmark SHALL be included in the SCAP source data stream.

If the XCCDF benchmark component references any CPE names, then the SCAP source data stream SHALL include a CPE component, which specifies the products or platforms of interest, and SHALL include one or more OVAL inventory class definitions in an OVAL component that contain the technical procedures for determining whether or not a specific target asset has a product or platform of interest.

The *@use-case* attribute in the *<ds:data-stream>* element SHALL be set to "VULNERABILITY."

5.3. Inventory Scanning

SCAP content can be used to collect information on the software installed on systems. One example of how this could be used is to verify that a group of systems all have required security software programs installed. This could help verify compliance with technical security control requirements. Another example is to collect software inventory data on devices that are not directly connected to the enterprise network, such as smart phones.

Inventory scanning can also be applied to collect information on the presence of software artifacts on systems, such as malware or characteristics of malware that indicate its presence. SCAP content authored for this purpose can be used to detect classes or categories of malware based on system states that may be common across multiple malware instances. For example, it is a common practice to reuse malware code while making modifications to address available detection methods or change propagation characteristics. It is also possible to author content that detects a specific instantiation of malware. For example, the hashing of files can be used to identify a malicious executable or library.

The SCAP source data stream component that SHALL be included for inventory scanning is the XCCDF benchmark, which references the inventory checks and captures the results. Each rule in the XCCDF benchmark SHALL reference one of the following:

- **An OVAL inventory definition.** This definition SHALL be contained in an OVAL component, which holds definitions of technical procedures for determining whether or not a specific target asset has software (e.g., product, platform, malware) of interest. An XCCDF benchmark's rules MAY reference one or more OVAL inventory class definitions in an OVAL component.
- **An OCIL questionnaire.** This questionnaire SHALL be contained in an OCIL component, which holds questionnaires that collect information that OVAL is not being used to collect, such as posing questions to users or harvesting inventory information from an existing database. An XCCDF benchmark's rules MAY reference one or more OCIL questionnaires in an OCIL component.

The *@use-case* attribute in the *<ds:data-stream>* element SHALL be set to "INVENTORY."

Appendix A. Security Considerations

Major security considerations for this version of SCAP include the following:

- **Confidentiality.** SCAP does not define any mechanisms for protecting the confidentiality of SCAP content or results. Organizations can add on such protections as they deem appropriate, such as encrypting results files that contain sensitive information regarding system vulnerabilities.
- **Malicious content.** While SCAP provides mechanisms for ensuring the integrity of SCAP content and verifying content signatures, SCAP does not have any features specifically for handling malicious SCAP content (e.g., benchmarks, tailoring files). At a minimum, organizations should generate signatures for their content and verify the signatures on all content before using it to ensure that the content has not been maliciously altered. Additionally, organizations should not process content that fails validation and may choose not to use any content that has not been signed for stronger assurance.
- **Security value of content.** Assertions or assessments regarding the security value of SCAP checklists and other forms of SCAP content is outside of the scope of SCAP's capabilities. People and organizations may determine security values through their own methods (e.g., applying checklists to test systems and evaluating the results of those tests), but SCAP itself does not have any way of ensuring the security value of its content.
- **Component security.** SCAP does not impose any additional security requirements on these on components that use SCAP (e.g., protocols, specifications, standards).

Appendix B. List of Symbols, Abbreviations, and Acronyms

Selected acronyms and abbreviations used in the guide are defined below.

AI
Asset Identification

API
Application Programming Interface

ARF
Asset Reporting Format

CCE
Common Configuration Enumeration

CCSS
Common Configuration Scoring System

CPE
Common Platform Enumeration

CVE
Common Vulnerabilities and Exposures

CVSS
Common Vulnerability Scoring System

DHS
Department of Homeland Security

DoD
Department of Defense

DSIG
Digital Signature

FISMA
Federal Information Security Modernization Act

IR
Interagency Report

IT
Information Technology

ITL
Information Technology Laboratory

NIST
National Institute of Standards and Technology

NVD
National Vulnerability Database

OASIS
Organization for the Advancement of Structured Information Standards

OCIL
Open Checklist Interactive Language

OMB
Office of Management and Budget

OS
Operating System

OVAL
Open Vulnerability and Assessment Language

PCI
Payment Card Industry

RFC
Request for Comments

SCAP
Security Content Automation Protocol

SP
Service Pack

SP
Special Publication

SWID
Software Identification

TMSAD
Trust Model for Security Automation Data

URI
Uniform Resource Identifier

URL
Uniform Resource Locator

XCCDF
Extensible Configuration Checklist Description Format

XML
Extensible Markup Language

Appendix C. Glossary

This appendix contains definitions for selected terms used within the document.

component schema
The schema for an SCAP component specification (e.g. XCCDF, CPE, CVSS). Within this document, this term is distinct from "OVAL component schema," which is defined by the OVAL specification.

component specification
One of the individual specifications that comprises SCAP.

content consumer
A product that accepts existing SCAP source data stream content, processes it, and produces SCAP result data streams

content producer
A product that generates SCAP source data stream content.

globally unique identifier
An identifier formatted following special conventions to support uniqueness within an organization and across all organizations creating identifiers. See Sec. 3.1.3 for the conventions.

result content
Part or all of one or more SCAP result data streams.

Security Content Automation Protocol (SCAP)
A suite of specifications that standardize the format and nomenclature by which software flaw and security configuration information is communicated to machines and humans.

SCAP component
A logical unit of data expressed using one or more of the SCAP component specifications.

SCAP-conformant
A product or SCAP data stream that meets the requirements of this specification.

SCAP content
Part or all of one or more SCAP data streams.

SCAP data stream
A specific instantiation of SCAP content.

SCAP data stream collection
A container for SCAP data streams and components.

SCAP result data stream
An SCAP data stream that holds output (result) content.

SCAP source data stream
An SCAP data stream that holds input (source) content.

SCAP source data stream collection
A container for SCAP data streams and components.

SCAP use case
A pre-defined way in which a product can use SCAP. See Sec. 5 for the definitions of the SCAP use cases.

source content
Part or all of SCAP source data streams.

stream component
A major element of a data stream, such as an XCCDF benchmark or a set of OVAL definitions.

well-formed
An SCAP-conformant data stream or stream component.

Appendix D. Normative References

This appendix provides normative references to the specifications that are required to implement the SCAP 1.4 components. See the annex for normative references to the XML and Schematron schema locations related to these specifications.

Table 24 lists the normative references to specifications.

Table 24. Specification locations

Abbreviation	Name	URL
[AI]	Asset Identification	https://doi.org/10.6028/NIST.IR.7693
[ARF]	ARF	https://doi.org/10.6028/NIST.IR.7694
[CCE]	CCE	https://nvd.nist.gov/config/cce/
[CCSS]	CCSS	https://doi.org/10.6028/NIST.IR.7502
[CPE]	CPE	See [CPE-D], [CPE-L], [CPE-M], and [CPE-N]
[CPE-D]	CPE Dictionary	https://doi.org/10.6028/NIST.IR.7697
[CPE-L]	CPE Applicability Language	https://doi.org/10.6028/NIST.IR.7698
[CPE-M]	CPE Name Matching	https://doi.org/10.6028/NIST.IR.7696
[CPE-N]	CPE Naming	https://doi.org/10.6028/NIST.IR.7695
[CVE]	CVE	https://cve.org/
[CVSS]	CVSS	https://www.first.org/cvss/specification-document
[DCES]	Dublin Core metadata version 1.1	http://dublincore.org/documents/2012/06/14/dces/
[DSIG]	DSIG	https://www.w3.org/TR/xmldsig-core/
[OCIL]	OCIL	https://doi.org/10.6028/NIST.IR.7692
[OVAL]	OVAL	https://github.com/OVAL-Community/OVAL
[RFC2119]	RFC 2119	https://doi.org/10.17487/RFC2119
[RFC3986]	RFC 3986	https://doi.org/10.17487/RFC3986
[SCHEMATRON]	**ISO/IEC 19757-3:2020**	https://www.iso.org/obp/ui/#iso:std:iso-iec:19757:-3:ed-3:v1:en
[SP800-126A]	NIST SP 800-126A	https://doi.org/10.6028/NIST.SP.800-126A
[SWID]	ISO/IEC 19770-2:2015	http://www.iso.org/iso/catalogue_detail.htm?csnumber=65666
[TMSAD]	TMSAD	https://doi.org/10.6028/NIST.IR.7802
[XCCDF]	XCCDF	https://csrc.nist.gov/CSRC/media/Publications/nistir/7275/rev-4/final/documents/nistir-7275r4_updated-march-2012_clean.pdf
[XINCLUDE]	XInclude specification	https://www.w3.org/TR/2006/REC-xinclude-20061115/
[XLINK]	XLink specification	https://www.w3.org/TR/2001/REC-xlink-20010627/
[XMLCAT]	XML Catalog specification	https://www.oasis-open.org/committees/download.php/14809/xml-catalogs.html
[XMLS]	W3C XML Schema	https://www.w3.org/TR/2004/REC-xmlschema-1-20041028/, https://www.w3.org/TR/2004/REC-xmlschema-2-20041028/

Appendix E. Change Log

<u>Revision 4 Release 1 – TBD</u>

- Updated all references of SCAP 1.3 to SCAP 1.4
- Removed backward compatibility requirements for SCAP 1.1 and 1.0
- Revised digital signature requirements
- Removed SWID requirements
- Removed references to OVAL ‘core’ and ‘platform’ schema versions