

The attached DRAFT document (provided here for historical purposes), released on July 18, 2016, has been superseded by the following publication:

Publication Number:     **NIST Special Publication (SP) 800-126 Revision 3**

Title:                     **The Technical Specification for the Security Content Automation Protocol (SCAP): SCAP Version 1.3**

Publication Date:       **February 2018**

- Final Publication: <https://doi.org/10.6028/NIST.SP.800-126r3> (which links to <http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-126r3.pdf>).
- Related Information on CSRC:  
Final: <https://csrc.nist.gov/publications/detail/sp/800-126/rev-3/final>
- Additional information:
  - SCAP homepage: <https://scap.nist.gov>
  - NIST cybersecurity publications and programs: <https://csrc.nist.gov>

1 **Draft NIST Special Publication 800-126**  
2 **Revision 3**

---

3 **The Technical Specification for the**  
4 **Security Content Automation Protocol**  
5 **(SCAP)**  
6 *SCAP Version 1.3*

---

7  
8 David Waltermire  
9 Stephen Quinn  
10 Harold Booth  
11 Karen Scarfone  
12 Dragos Prisaca  
13  
14  
15  
16  
17

---

18 **C O M P U T E R   S E C U R I T Y**  
19  
20

---

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

21 **Draft NIST Special Publication 800-126**  
22 **Revision 3**

23  
24 **The Technical Specification for the**  
25 **Security Content Automation Protocol**  
26 **(SCAP)**  
27 *SCAP Version 1.3*  
28

29 David Waltermire  
30 Stephen Quinn  
31 Harold Booth  
32 *Computer Security Division*  
33 *Information Technology Laboratory*  
34

35 Karen Scarfone  
36 *Scarfone Cybersecurity*  
37 *Clifton, VA*  
38

39 Dragos Prisaca  
40 *G2, Inc.*  
41 *Annapolis Junction, MD*  
42

43 July 2016  
44



45  
46  
47  
48 U.S. Department of Commerce  
49 *Penny Pritzker, Secretary*

50  
51 National Institute of Standards and Technology  
52 *Willie May, Under Secretary of Commerce for Standards and Technology and Director*

53

## Authority

54 This publication has been developed by NIST in accordance with its statutory responsibilities  
55 under the Federal Information Security Modernization Act (FISMA) of 2014, 44 U.S.C. § 3541 *et*  
56 *seq.*, Public Law (P.L.) 113-283. NIST is responsible for developing information security  
57 standards and guidelines, including minimum requirements for federal information systems, but  
58 such standards and guidelines shall not apply to national security systems without the express  
59 approval of appropriate federal officials exercising policy authority over such systems. This  
60 guideline is consistent with the requirements of the Office of Management and Budget (OMB)  
61 Circular A-130.

62 Nothing in this publication should be taken to contradict the standards and guidelines made  
63 mandatory and binding on federal agencies by the Secretary of Commerce under statutory  
64 authority. Nor should these guidelines be interpreted as altering or superseding the existing  
65 authorities of the Secretary of Commerce, Director of the OMB, or any other federal official.  
66 This publication may be used by nongovernmental organizations on a voluntary basis and is not  
67 subject to copyright in the United States. Attribution would, however, be appreciated by NIST.

68 National Institute of Standards and Technology Special Publication 800-126 Revision 3  
69 Natl. Inst. Stand. Technol. Spec. Publ. 800-126 Rev. 3, 62 pages (July 2016)  
70 CODEN: NSPUE2

71 Certain commercial entities, equipment, or materials may be identified in this document in order to describe  
72 an experimental procedure or concept adequately. Such identification is not intended to imply  
73 recommendation or endorsement by NIST, nor is it intended to imply that the entities, materials, or  
74 equipment are necessarily the best available for the purpose.

75 There may be references in this publication to other publications currently under development by NIST in  
76 accordance with its assigned statutory responsibilities. The information in this publication, including  
77 concepts and methodologies, may be used by federal agencies even before the completion of such  
78 companion publications. Thus, until each publication is completed, current requirements, guidelines, and  
79 procedures, where they exist, remain operative. For planning and transition purposes, federal agencies may  
80 wish to closely follow the development of these new publications by NIST.

81 Organizations are encouraged to review all draft publications during public comment periods and provide  
82 feedback to NIST. Many NIST cybersecurity publications, other than the ones noted above, are available at  
83 <http://csrc.nist.gov/publications>.

84 **Public comment period: July 18, 2016 through August 19, 2016**

85 National Institute of Standards and Technology  
86 Attn: Computer Security Division, Information Technology Laboratory  
87 100 Bureau Drive (Mail Stop 8930) Gaithersburg, MD 20899-8930  
88 Email: 800-126comments@nist.gov

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

90

91

92

## Reports on Computer Systems Technology

93 The Information Technology Laboratory (ITL) at the National Institute of Standards and  
94 Technology (NIST) promotes the U.S. economy and public welfare by providing technical  
95 leadership for the Nation's measurement and standards infrastructure. ITL develops tests, test  
96 methods, reference data, proof of concept implementations, and technical analyses to advance  
97 the development and productive use of information technology. ITL's responsibilities include the  
98 development of management, administrative, technical, and physical standards and guidelines for  
99 the cost-effective security and privacy of other than national security-related information in  
100 federal information systems. The Special Publication 800-series reports on ITL's research,  
101 guidelines, and outreach efforts in information system security, and its collaborative activities  
102 with industry, government, and academic organizations.

103

104

### Abstract

105 The Security Content Automation Protocol (SCAP) is a suite of specifications that standardize  
106 the format and nomenclature by which software flaw and security configuration information is  
107 communicated, both to machines and humans. This publication defines the technical composition  
108 of SCAP version 1.3 in terms of its component specifications, their interrelationships and  
109 interoperation, and the requirements for SCAP content.

110

111

### Keywords

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

114

115

116

117

## Acknowledgments

118 The authors, David Waltermire, Stephen Quinn, and Harold Booth of the National Institute of Standards  
119 and Technology (NIST); Karen Scarfone of Scarfone Cybersecurity; and Dragos Prisaca of G2, Inc., wish  
120 to thank all contributors to this revision of the publication, particularly Melanie Cook, Lee Badger, and  
121 Jim Foti of NIST; Steve Grubb of Red Hat; Roger Johnson of the Centers for Disease Control and  
122 Prevention; Kent Landfield of Intel Corporation; Adam W. Montville and William K. Munyan of the  
123 Center for Internet Security; and David A. Solin of Joval Continuous Monitoring. The authors also  
124 recognize the contributions of Adam Halbardier of Booz Allen Hamilton, who co-authored Revision 1 of  
125 this publication.

126

127 The authors would like to acknowledge the following contributors to previous versions of this  
128 specification for their keen and insightful assistance: John Banghart, Paul Cichonski, and Blair Heiserman  
129 of NIST; Christopher Johnson of HP Enterprise Services; Paul Bartock of the National Security Agency  
130 (NSA); Jeff Ito, Matt Kerr, Shane Shaffer, and Greg Witte of G2, Inc.; Andy Bove of SecureAcuity; Jim  
131 Ronayne of Varen Technologies; Kent Landfield of McAfee, Inc.; Christopher McCormick, Rhonda  
132 Farrell, Angela Orebaugh, and Victoria Thompson of Booz Allen Hamilton; Alan Peltzman of the  
133 Defense Information Systems Agency (DISA); and Jon Baker, Drew Buttner, Maria Casipe, and Charles  
134 Schmidt of the MITRE Corporation.

135

136

## Trademark Information

137 OVAL is a trademark of the US Department of Homeland Security (DHS).

138 CVE is a registered trademark of The MITRE Corporation.

139 Windows 7 is a registered trademark of Microsoft Corporation.

140 All other registered trademarks or trademarks belong to their respective organizations.

141

142

## Note to Reviewers

143 Feedback on any part of the publication is welcome, but comments on the following are of particular  
144 interest:

- 145 • The content of the new NIST SP 800-126A publication, including changes to OVAL core schema and  
146 platform schema version support and the elimination of the “least version principle”
- 147 • The addition to SCAP of the Software Identification (SWID) Tags 2015 revision specification and  
148 associated requirements
- 149 • The update of the CVSS specification version from 2.0 to 3.0
- 150 • The revisions to requirements involving legacy SCAP content and OVAL content

151

152 Errata

153 This table will contain changes that have been incorporated into NIST Special Publication 800-126  
154 Revision 3 after the publication is finalized. Errata updates can include corrections, clarifications, or other  
155 minor changes in the publication that are either *editorial* or *substantive* in nature.

Date	Type	Change	Pages

156

157

158

## Table of Contents

159	Executive Summary .....	1
160	1. Introduction .....	3
161	1.1 Purpose and Scope .....	3
162	1.2 Audience .....	3
163	1.3 Document Structure .....	3
164	1.4 Document Conventions .....	4
165	2. SCAP 1.3 Conformance .....	6
166	2.1 Product Conformance.....	7
167	2.2 Source Content Conformance .....	7
168	3. SCAP Content Requirements and Recommendations.....	9
169	3.1 SCAP Source Data Stream .....	9
170	3.1.1 Source Data Stream Data Model.....	11
171	3.1.2 Source Data Stream Collection Validation.....	17
172	3.1.3 Globally Unique Identifiers .....	18
173	3.2 Extensible Configuration Checklist Description Format (XCCDF) .....	18
174	3.2.1 General .....	18
175	3.2.2 The <xccdf:Benchmark> Element .....	19
176	3.2.3 The <xccdf:Profile> Element .....	20
177	3.2.4 The <xccdf:Rule> Element .....	20
178	3.2.5 The <xccdf:Value> Element .....	23
179	3.2.6 The <xccdf:Group> Element.....	24
180	3.3 Open Vulnerability and Assessment Language (OVAL).....	24
181	3.4 Open Checklist Interactive Language (OCIL) .....	26
182	3.5 Common Platform Enumeration (CPE).....	27
183	3.6 Software Identification (SWID) Tags.....	28
184	3.7 Common Configuration Enumeration (CCE).....	28
185	3.8 Common Vulnerabilities and Exposures (CVE) .....	29
186	3.9 Common Vulnerability Scoring System (CVSS).....	29
187	3.10 Common Configuration Scoring System (CCSS).....	29
188	3.11 XML Digital Signature.....	29
189	4. SCAP Content Processing Requirements and Recommendations .....	31
190	4.1 Legacy Support .....	31
191	4.2 Source Data Streams .....	31
192	4.3 XCCDF Processing .....	32
193	4.3.1 CPE Applicability Processing .....	32
194	4.3.2 Check System Usage.....	32
195	4.4 SCAP Result Data Streams.....	33
196	4.4.1 The Component Reports .....	34
197	4.4.2 The Target Identification.....	34
198	4.4.3 The Source Data Stream.....	34
199	4.4.4 The Relationships .....	35
200	4.5 XCCDF Results .....	36
201	4.5.1 Assigning Identifiers to Rule Results .....	38
202	4.5.2 Mapping OVAL Results to XCCDF Results .....	39
203	4.6 OVAL Results.....	40

204 4.7 OCIL Results .....42  
 205 4.8 Result Data Stream Signing .....42  
 206 5. Source Data Stream Content Requirements for Use Cases .....44  
 207 5.1 Compliance Checking.....44  
 208 5.2 Vulnerability Scanning .....44  
 209 5.3 Inventory Scanning.....45  
 210 Appendix A— Security Considerations.....47  
 211 Appendix B— Acronyms and Abbreviations .....48  
 212 Appendix C— Glossary .....49  
 213 Appendix D— Normative References.....50  
 214 Appendix E— Change Log.....51

215

216

217

List of Figures

218 Figure 1: SCAP Data Stream Collection..... 9  
 219 Figure 2: SCAP Data Stream .....10  
 220 Figure 3: Sample ARF Report Structure.....36

221

222

List of Tables

223 Table 1: Conventional XML Mappings..... 5  
 224 Table 2: ds:data-stream-collection .....12  
 225 Table 3: ds:data-stream .....13  
 226 Table 4: ds:dictionaries .....13  
 227 Table 5: ds:checklists.....13  
 228 Table 6: ds:checks .....14  
 229 Table 7: ds:extended-components .....14  
 230 Table 8: ds:component-ref .....14  
 231 Table 9: cat:catalog.....15  
 232 Table 10: cat:uri .....15  
 233 Table 11: cat:rewriteURI .....16  
 234 Table 12: ds:component.....16  
 235 Table 13: ds:extended-component.....17  
 236 Table 14: SCAP Source Data Stream Component Document Elements .....17

237 Table 15: Element Identifier Format Convention ..... 18

238 Table 16: Use of Dublin Core Terms in <xccdf:metadata> .....20

239 Table 17: <xccdf:Rule> and <xccdf:ident> Element Values.....20

240 Table 18: XCCDF-OVAL Data Export Matching Constraints .....24

241 Table 19: SCAP Result Data Stream Component Document Elements .....34

242 Table 20: Asset Identification Fields to Populate .....34

243 Table 21: ARF Relationships.....35

244 Table 22: XCCDF Fact Descriptions .....37

245 Table 23: Deriving XCCDF Check Results from OVAL Definition Results .....40

246 Table 24: Specification Locations.....50

247

## 248 Executive Summary

249 The Security Content Automation Protocol (SCAP) is a suite of specifications that standardize the format  
250 and nomenclature by which software flaw and security configuration information is communicated, both  
251 to machines and humans.<sup>1</sup> SCAP is a multi-purpose framework of specifications that support automated  
252 configuration, vulnerability and patch checking, technical control compliance activities, and security  
253 measurement. Goals for the development of SCAP include standardizing system security management,  
254 promoting interoperability of security products, and fostering the use of standard expressions of security  
255 content.

256 SCAP version 1.3 is comprised of twelve component specifications in five categories:

- 257 • **Languages.** The SCAP languages provide standard vocabularies and conventions for expressing  
258 security policy, technical check mechanisms, and assessment results. The SCAP language  
259 specifications are Extensible Configuration Checklist Description Format (XCCDF), Open  
260 Vulnerability and Assessment Language (OVAL®), and Open Checklist Interactive Language  
261 (OCIL™).
- 262 • **Reporting formats.** The SCAP reporting formats provide the necessary constructs to express  
263 collected information in standardized formats. The SCAP reporting format specifications are Asset  
264 Reporting Format (ARF) and Asset Identification. Although Asset Identification is not explicitly a  
265 reporting format, SCAP uses it as a key component in identifying the assets that reports relate to.
- 266 • **Identification schemes.** The SCAP identification schemes provide a means to identify key concepts  
267 such as software products, vulnerabilities, and configuration items using standardized identifier  
268 formats. They also provide a means to associate individual identifiers with additional data pertaining  
269 to the subject of the identifier. The SCAP identification scheme specifications are Common Platform  
270 Enumeration (CPE™), Software Identification (SWID) Tags, Common Configuration Enumeration  
271 (CCE™), and Common Vulnerabilities and Exposures (CVE®).
- 272 • **Measurement and scoring systems.** In SCAP this refers to evaluating specific characteristics of a  
273 security weakness (for example, software vulnerabilities and security configuration issues) and, based  
274 on those characteristics, generating a score that reflects their relative severity. The SCAP  
275 measurement and scoring system specifications are Common Vulnerability Scoring System (CVSS)  
276 and Common Configuration Scoring System (CCSS).
- 277 • **Integrity.** An SCAP integrity specification helps to preserve the integrity of SCAP content and  
278 results. Trust Model for Security Automation Data (TMSAD) is the SCAP integrity specification.

279 SCAP utilizes software flaw and security configuration standard reference data. This reference data is  
280 provided by the National Vulnerability Database (NVD),<sup>2</sup> which is managed by NIST and sponsored by  
281 the Department of Homeland Security (DHS).

282 This publication defines the technical composition of SCAP version 1.3 in terms of its component  
283 specifications, their interrelationships and interoperation, and the requirements for SCAP content. The  
284 technical specification for SCAP in this publication describes the requirements and conventions that are to  
285 be employed to ensure the consistent and accurate exchange of SCAP-conformant content and the ability  
286 to reliably use the content with SCAP-conformant products.

287 The U.S. Federal Government, in cooperation with academia and private industry, has adopted SCAP and  
288 encourages its use in support of security automation activities and initiatives.<sup>3</sup> SCAP has achieved

<sup>1</sup> Products implementing SCAP can also be used to support non-security use cases such as configuration management and software inventory.

<sup>2</sup> <http://nvd.nist.gov/>

<sup>3</sup> Refer to <http://www.whitehouse.gov/omb/memoranda/fy2008/m08-22.pdf>.

289 widespread adoption by major software manufacturers and has become a significant component of large  
290 information security management and governance programs. The protocol is expected to evolve and  
291 expand in support of the growing needs to define and measure effective security controls, assess and  
292 monitor ongoing aspects of that information security, and successfully manage systems in accordance  
293 with risk management frameworks such as NIST Special Publication 800-53<sup>4</sup>, Department of Defense  
294 (DoD) Instruction 8500.2, and the Payment Card Industry (PCI) framework.

295 By detailing the specific and appropriate usage of the SCAP 1.3 components and their interoperability,  
296 NIST encourages the creation of reliable and pervasive SCAP content and the development of a wide  
297 array of products that leverage SCAP.

298 Organizations that develop SCAP 1.3-based content or products should comply with the following  
299 recommendations:

300 **Follow the requirements listed in this document and in the associated component specifications.**

301 Organizations should ensure that their implementation and use of SCAP 1.3 is compliant with the  
302 requirements detailed in each component specification and this document.

303 If requirements are in conflict between component specifications, this document will provide clarification.  
304 If a component specification is in conflict with this document, the requirements in this document take  
305 precedence.

306 **When creating SCAP content, adhere to the conventions specified in this document.**

307 Security products and checklist authors assemble content from SCAP data repositories to create SCAP-  
308 conformant security guidance. For example, a security configuration checklist can document desired  
309 security configuration settings, installed patches, and other system security elements using a standardized  
310 SCAP format. Such a checklist would use XCCDF to describe the checklist, CCE to identify security  
311 configuration settings to be addressed or assessed, and CPE and SWID tags to identify platforms for  
312 which the checklist is valid. The use of CCE and CPE entries within XCCDF checklists is an example of  
313 an SCAP convention—a requirement for valid SCAP usage. These conventions part of the definition of  
314 SCAP 1.3. Organizations producing SCAP content should adhere to these conventions to ensure the  
315 highest degree of interoperability. NIST provides an SCAP Content Validation Tool that organizations  
316 can use to help validate the correctness of their SCAP content. The tool checks that SCAP source and  
317 result content is well-formed, all cross references are valid, and required values are appropriately set.<sup>5</sup>

318

---

<sup>4</sup> The Risk Management Framework is described in Section 3.0 of NIST Special Publication 800-53, available at <http://csrc.nist.gov/publications/PubsSPs.html#800-53>.

<sup>5</sup> <http://scap.nist.gov/revision/1.3/#tools>

## 319 1. Introduction

### 320 1.1 Purpose and Scope

321 This document, NIST Special Publication (SP) 800-126 Revision 3, and its annex, NIST SP 800-126A<sup>6</sup>,  
322 collectively provide the definitive technical specification for version 1.3 of the Security Content  
323 Automation Protocol (SCAP). *SCAP* (pronounced ess-cap) consists of a suite of specifications for  
324 standardizing the format and nomenclature by which software flaw and security configuration information  
325 is communicated, both to machines and humans. This document defines requirements for creating and  
326 processing SCAP source content. These requirements build on the requirements defined within the  
327 individual SCAP component specifications. Each new requirement pertains either to using multiple  
328 component specifications together or to further constraining one of the individual component  
329 specifications. The requirements within the individual component specifications are not repeated in this  
330 document; see those specifications to view their requirements.

331 To extend the contents of this document, an annex has been created. The annex document specifies  
332 additional entities that may be used in SCAP 1.3 conformant content creation and processing:

- 333 • Particular minor version updates to SCAP 1.3 component specifications
- 334 • Particular Open Vulnerability and Assessment Language (OVAL) platform schema versions

335 The scope of this document and its annex are limited to SCAP version 1.3. Other versions of SCAP and  
336 its component specifications are not addressed in these documents.

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

### 339 1.2 Audience

340 This document is intended for three primary audiences:

- 341 • Content authors and editors seeking to ensure that the SCAP source content they produce operates  
342 correctly, consistently, and reliably in SCAP products.
- 343 • Software developers and system integrators seeking to create, use, or exchange SCAP content in their  
344 products or service offerings.
- 345 • Product developers preparing for SCAP validation at an accredited independent testing laboratory.

346 This document assumes that readers already have general knowledge of SCAP and reasonable familiarity  
347 with the SCAP component specifications that their content, products, or services use. Individuals without  
348 this level of knowledge who would like to learn more about SCAP should consult NIST SP 800-117,  
349 *Guide to Adopting and Using the Security Content Automation Protocol*.<sup>7</sup>

### 350 1.3 Document Structure

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

- 352 • Section 2 provides the high-level requirements for claiming conformance with the SCAP 1.3  
353 specification.
- 354 • Section 3 details the requirements and recommendations for SCAP content syntax, structure, and  
355 development.

---

<sup>6</sup> <http://csrc.nist.gov/publications/PubsDrafts.html>

<sup>7</sup> <http://csrc.nist.gov/publications/PubsSPs.html#800-117>

- 356 • Section 4 defines SCAP content processing requirements and recommendations.
- 357 • Section 5 provides additional content requirements and recommendations for particular use cases.
- 358 • Appendix A gives an overview of major security considerations for SCAP implementation.
- 359 • Appendix B contains an acronym and abbreviation list.
- 360 • Appendix C contains a glossary of selected terms used in the document.
- 361 • Appendix D lists references and other resources related to SCAP 1.3.
- 362 • Appendix E provides a change log that documents significant changes to major drafts of this
- 363 specification.

#### 364 1.4 Document Conventions

365 The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”,  
 366 “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be  
 367 interpreted as described in Request for Comment (RFC) 2119 [RFC2119]. When these words appear in  
 368 regular case, such as “should” or “may”, they are not intended to be interpreted as RFC 2119 key words.

369 When a single term within a sentence is italicized, this indicates that the term is being defined. These  
 370 terms and their definitions also appear in Appendix C.

371 Some of the requirements and conventions used in this document reference Extensible Markup Language  
 372 (XML) content [XMLS]. These references come in two forms, inline and indented. An example of an  
 373 inline reference is: a `<cpe2_dict:cpe-item>` may contain `<cpe2_dict:check>` elements that  
 374 reference OVAL Definitions.

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

377 An example of an indented reference is:

378 References to OVAL Definitions are expressed using the following format:

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

383 The general convention used when describing XML attributes within this document is to reference the  
 384 attribute as well as its associated element including the namespace alias, employing the general form  
 385 “@*attributeName* for the `<prefix:localName>`”.

386 Indented references are intended to represent the form of actual XML content. Indented references  
 387 represent literal content by the use of a *fixed-length font*, and parametric (freely replaceable)  
 388 content by the use of an *italic font*. Square brackets ‘[ ]’ are used to designate optional content. Thus  
 389 “[*Oval\_inventory\_definition\_id*]” designates optional parametric content.

390 Both inline and indented forms use qualified names to refer to specific XML elements. A qualified name  
 391 associates a named element with a namespace. The namespace identifies the XML model, and the XML  
 392 schema is a definition and implementation of that model. A qualified name declares this schema to  
 393 element association using the format ‘*prefix:element-name*’. The association of prefix to namespace is  
 394 defined in the metadata of an XML document and varies from document to document. In this  
 395 specification, the conventional mappings listed in Table 1 are used.

396

**Table 1: Conventional XML Mappings**

Prefix	Namespace	Schema
ai	<a href="http://scap.nist.gov/schema/asset-identification/1.1">http://scap.nist.gov/schema/asset-identification/1.1</a>	Asset Identification
arf	<a href="http://scap.nist.gov/schema/asset-reporting-format/1.1">http://scap.nist.gov/schema/asset-reporting-format/1.1</a>	ARF
arf-rel	<a href="http://scap.nist.gov/specifications/arf/vocabulary/relationships/1.0#">http://scap.nist.gov/specifications/arf/vocabulary/relationships/1.0#</a>	ARF relationships
cat	<a href="urn:oasis:names:tc:entity:xmlns:xml:catalog">urn:oasis:names:tc:entity:xmlns:xml:catalog</a>	XML Catalog
con	<a href="http://scap.nist.gov/schema/scap/constructs/1.3">http://scap.nist.gov/schema/scap/constructs/1.3</a>	SCAP Constructs
cpe-dict-ext	<a href="http://scap.nist.gov/schema/cpe-extension/2.3">http://scap.nist.gov/schema/cpe-extension/2.3</a>	CPE Dictionary 2.3 schema extension
cpe2	<a href="http://cpe.mitre.org/language/2.0">http://cpe.mitre.org/language/2.0</a>	Embedded CPE references
cpe2-dict	<a href="http://cpe.mitre.org/dictionary/2.0">http://cpe.mitre.org/dictionary/2.0</a>	CPE dictionaries
cve	<a href="http://scap.nist.gov/schema/vulnerability/0.4">http://scap.nist.gov/schema/vulnerability/0.4</a>	NVD/CVE data feed elements and attributes
dc	<a href="http://purl.org/dc/elements/1.1/">http://purl.org/dc/elements/1.1/</a>	Simple Dublin Core elements
ds	<a href="http://scap.nist.gov/schema/scap/source/1.2">http://scap.nist.gov/schema/scap/source/1.2</a>	SCAP source data stream collection
dt	<a href="http://scap.nist.gov/schema/xml-dsig/1.0">http://scap.nist.gov/schema/xml-dsig/1.0</a>	Security automation digital signature extensions
nvd	<a href="http://scap.nist.gov/schema/feed/vulnerability/2.0">http://scap.nist.gov/schema/feed/vulnerability/2.0</a>	Base schema for NVD data feeds
ocil	<a href="http://scap.nist.gov/schema/ocil/2.0">http://scap.nist.gov/schema/ocil/2.0</a>	OCIL elements and attributes
oval	<a href="http://oval.mitre.org/XMLSchema/oval-common-5">http://oval.mitre.org/XMLSchema/oval-common-5</a>	Common OVAL elements and attributes
oval-def	<a href="http://oval.mitre.org/XMLSchema/oval-definitions-5">http://oval.mitre.org/XMLSchema/oval-definitions-5</a>	OVAL Definitions
oval-res	<a href="http://oval.mitre.org/XMLSchema/oval-results-5">http://oval.mitre.org/XMLSchema/oval-results-5</a>	OVAL results
oval-sc	<a href="http://oval.mitre.org/XMLSchema/oval-system-characteristics-5">http://oval.mitre.org/XMLSchema/oval-system-characteristics-5</a>	OVAL system characteristics
oval-var	<a href="http://oval.mitre.org/XMLSchema/oval-variables-5">http://oval.mitre.org/XMLSchema/oval-variables-5</a>	The elements, types, and attributes that compose the core schema for encoding OVAL Variables. This schema is provided to give structure to any external variables and their values that an OVAL Definition is expecting.
scap-rel	<a href="http://scap.nist.gov/vocabulary/scap/relationships/1.0#">http://scap.nist.gov/vocabulary/scap/relationships/1.0#</a>	SCAP relationships
sch	<a href="http://purl.oclc.org/dsdl/schematron">http://purl.oclc.org/dsdl/schematron</a>	Schematron validation scripts
swid	<a href="http://standards.iso.org/iso/19770/-2/2015/schema.xsd">http://standards.iso.org/iso/19770/-2/2015/schema.xsd</a>	SWID tag documents
xccdf	<a href="http://checklists.nist.gov/xccdf/1.2">http://checklists.nist.gov/xccdf/1.2</a>	XCCDF policy documents
xlink	<a href="http://www.w3.org/1999/xlink">http://www.w3.org/1999/xlink</a>	XML Linking Language
xml	<a href="http://www.w3.org/XML/1998/namespace">http://www.w3.org/XML/1998/namespace</a>	Common XML attributes
xs	<a href="http://www.w3.org/2001/XMLSchema">http://www.w3.org/2001/XMLSchema</a>	XML schema
xxxx-def, xxxx-sc	See the annex document for the mappings for OVAL definition and system characteristic schemas	OVAL elements and attributes specific to an OS, Hardware, or Application type xxxx

397

398

## 399 2. SCAP 1.3 Conformance

400 The *component specifications* included in SCAP 1.3 are as follows:

- 401 • Languages
  - 402 ○ Extensible Configuration Checklist Description Format (XCCDF) 1.2, a language for authoring
  - 403 security checklists/benchmarks and for reporting results of evaluating them [XCCDF]
  - 404 ○ Open Vulnerability and Assessment Language (OVAL), a language for representing system
  - 405 configuration information, assessing machine state, and reporting assessment results<sup>8</sup>
  - 406 ○ Open Checklist Interactive Language (OCIL) 2.0, a language for representing checks that collect
  - 407 information from people or from existing data stores made by other data collection efforts [OCIL]
- 408 • Reporting formats
  - 409 ○ Asset Reporting Format (ARF) 1.1, a format for expressing the transport format of information
  - 410 about assets and the relationships between assets and reports [ARF]
  - 411 ○ Asset Identification 1.1, a format for uniquely identifying assets based on known identifiers
  - 412 and/or known information about the assets [AI]
- 413 • Identification schemes
  - 414 ○ Common Platform Enumeration (CPE) 2.3, a nomenclature and dictionary of hardware, operating
  - 415 systems, and applications [CPE]
  - 416 ○ Software Identification (SWID) Tags 2015 revision, a format for representing software identifiers
  - 417 and associated metadata<sup>9</sup> [SWID]
  - 418 ○ Common Configuration Enumeration (CCE) 5, a nomenclature and dictionary of software
  - 419 security configurations [CCE]
  - 420 ○ Common Vulnerabilities and Exposures (CVE), a nomenclature and dictionary of security-related
  - 421 software flaws<sup>10</sup> [CVE]
- 422 • Measurement and scoring systems
  - 423 ○ Common Vulnerability Scoring System (CVSS) 3.0, a system for measuring the relative severity
  - 424 of software flaw vulnerabilities [CVSS]
  - 425 ○ Common Configuration Scoring System (CCSS) 1.0, a system for measuring the relative severity
  - 426 of system security configuration issues [CCSS]
- 427 • Integrity
  - 428 ○ Trust Model for Security Automation Data (TMSAD) 1.0, a specification for using digital
  - 429 signatures in a common trust model applied to other security automation specifications
  - 430 [TMSAD].

431 All references to these specifications within this document are to the version numbers listed above or in  
 432 the annex unless otherwise explicitly specified.

433 Combinations of these specifications can be used together for particular functions, such as security  
 434 configuration checking. These functions, known as *SCAP use cases*, are ways in which a product can use

<sup>8</sup> See the SCAP 1.3 annex document, NIST SP 800-126A, for the OVAL component specification (core schema) versions and platform schema versions that are supported by SCAP 1.3.

<sup>9</sup> The “2015 revision” refers to ISO/IEC 19770-2:2015, which is the specification for SWID tags.

<sup>10</sup> CVE does not have a version number.

435 SCAP. The collective XML content used for a use case is called an *SCAP data stream*, which is a specific  
436 instantiation of SCAP content. There are two types of SCAP data streams: an *SCAP source data stream*  
437 holds the input content, and an *SCAP result data stream* holds the output content. The major elements of  
438 a data stream, such as an XCCDF benchmark or a set of OVAL Definitions, are referred to as *stream*  
439 *components*.

440 Products and source content may want to claim conformance to one or more of the SCAP use cases,  
441 which are defined in Section 5 of this document, for a variety of reasons. For example, a product may  
442 want to assert that it uses SCAP content properly and can interoperate with other products using valid  
443 SCAP content. Another example is a policy mandating that an organization use SCAP source content for  
444 performing vulnerability assessments and other security operations.

445 This section provides the high-level requirements that a product or source content must meet for  
446 conformance with the SCAP 1.3 specification. Such products and source content are referred to as *SCAP*  
447 *conformant*. Most of the requirements listed in this section reference other sections in the document that  
448 fully define the requirements.

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

## 453 **2.1 Product Conformance**

454 There are two types of SCAP-conformant products: content producers and content consumers. *Content*  
455 *producers* are products that generate SCAP source data stream content, while *content consumers* are  
456 products that accept existing SCAP source data stream content, process it, and produce SCAP result data  
457 streams. Products claiming conformance with the SCAP 1.3 specification SHALL comply with the  
458 following requirements:

- 459 1. Adhere to the requirements detailed in each applicable component specification (for each selected  
460 SCAP component specification, and for each SCAP component specification required to  
461 implement the selected SCAP use cases). The authoritative references for each specification are  
462 listed in Appendix C.
- 463 2. Adhere to the requirements detailed in the errata for this document.
- 464 3. For content producers, generate well-formed SCAP source data streams. This includes following  
465 the source content conformance requirements specified in Section 2.2, and following the  
466 requirements in Section 5 for the use cases that the content producer supports.
- 467 4. For content consumers, consume and process well-formed SCAP source data streams, and  
468 generate well-formed SCAP result data streams. This includes following all of the processing  
469 requirements defined in Section 4 for each selected SCAP component specification and each  
470 SCAP component specification required to implement the selected SCAP use cases.
- 471 5. Make an explicit claim of conformance to this specification in any documentation provided to end  
472 users.

## 473 **2.2 Source Content Conformance**

474 Source content (i.e., source data streams) claiming conformance with the SCAP 1.3 specification SHALL  
475 comply with the following requirements:

- 476 1. Adhere to the requirements detailed in each applicable component specification (for each selected  
477 SCAP component specification, and each SCAP component specification required to implement

- 478 the selected SCAP use cases). The authoritative references for each specification are listed in  
479 Appendix C.
- 480 2. Adhere to the requirements detailed in the errata for this document.
- 481 3. Follow all of the syntax, structural, and other source content design requirements defined in  
482 Section 3 for each selected SCAP component specification and for each SCAP component  
483 specification required to implement the selected SCAP use cases. Also, follow all of the  
484 requirements specified for the content's use cases as defined in Section 5.

485

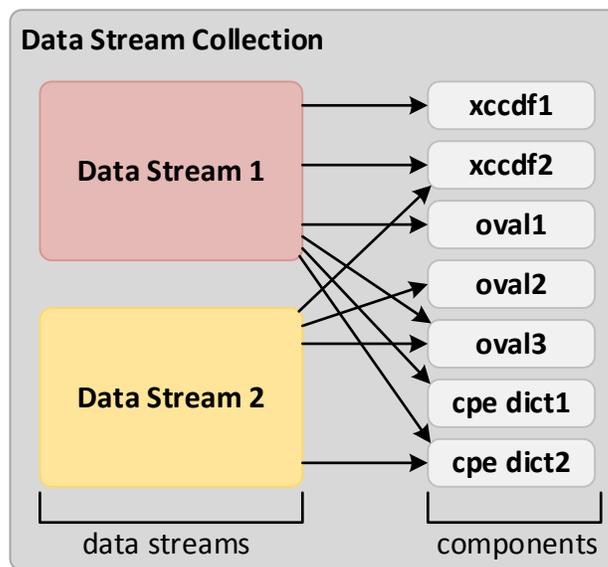
### 3. SCAP Content Requirements and Recommendations

486 This section defines the SCAP 1.3 content syntax, structure, and development requirements and  
 487 recommendations for SCAP-conformant content and products. Organizations are encouraged to adopt the  
 488 optional recommendations to promote stronger interoperability and greater content consistency. The first  
 489 part of the section discusses SCAP source data streams. The middle of the section groups requirements  
 490 and recommendations by specification: XCCDF, OVAL, OCIL, CPE, SWID, CCE, CVE, CVSS, and  
 491 CCSS, in that order. Finally, the last part of the section discusses applying XML digital signatures to  
 492 source data streams.

#### 3.1 SCAP Source Data Stream

494 This section discusses SCAP source data streams only; SCAP result data streams are discussed in Section  
 495 4.4 as part of the requirements for SCAP processing.

496 An *SCAP source data stream collection* is composed of SCAP data streams and SCAP source  
 497 components. See <http://scap.nist.gov/revision/1.3/#example> for a sample of an SCAP source data stream  
 498 collection and its sections. The components section contains an unbounded number of *SCAP source*  
 499 *components*, each consisting of data expressed using one or more of the SCAP specifications. The data  
 500 streams section contains one or more source data streams, each of which references the source  
 501 components in the components section that compose the data stream. This model allows source  
 502 components to be reused across multiple data streams. Many data streams are allowed in a data stream  
 503 collection to allow grouping of related or similar source data streams. For example, NIST currently  
 504 distributes the United States Government Configuration Baseline (USGCB)<sup>11</sup> as a series of SCAP  
 505 bundles. Source data streams that are similar or related (e.g., Microsoft Windows 7 content and Microsoft  
 506 Windows 7 Firewall content) may be bundled into the same source data stream collection. Figure 1 shows  
 507 the relationship between data stream collections, data streams, and components.



508

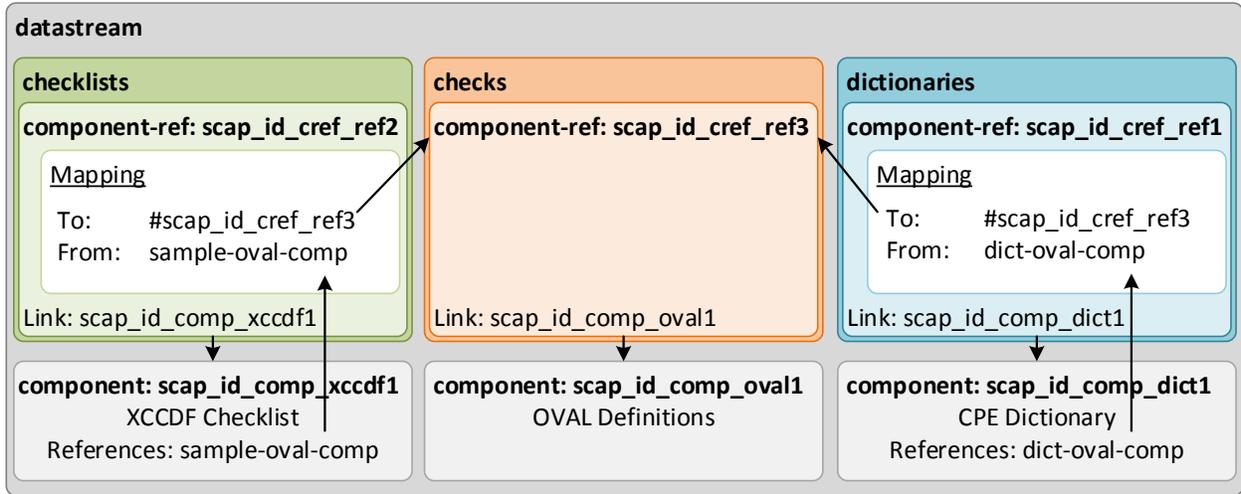
509

Figure 1: SCAP Data Stream Collection

<sup>11</sup> <http://usgcb.nist.gov/>

510 In Figure 1, data stream1 points to xccdf1, xccdf2, oval1, oval3, cpe dict1, and cpe dict2. data stream2  
 511 points to xccdf2, oval2, oval3, and cpe dict2. Each data stream is a collection of links to the components  
 512 that they reference; each logical link encapsulates the information required to allow the content consumer  
 513 to connect the components together within the data stream. Content authors MAY place components in in  
 514 any order. For example, some authors might choose to place dictionary components first to help optimize  
 515 data stream parsing.

516 Links serve two purposes: to indicate which component is being referred to, and to provide a map to  
 517 associate references within a component to other links within the data stream. The latter allows a data  
 518 stream to define context for each component’s references within the bounds of the data stream’s own set  
 519 of links. Figure 2 provides a conceptual example that illustrates how a data stream is constructed.



520

521

Figure 2: SCAP Data Stream

522 In Figure 2, the data stream links to three components. The OVAL component does not reference out to  
 523 external content, so there are no mappings captured for it. The XCCDF and CPE Dictionary components  
 524 reference other components (e.g., scap\_id\_cref\_ref3). When referencing components within the example  
 525 data stream, a mapping indicates that when scap\_id\_comp\_xccdf1 references “sample-oval-comp”, the  
 526 content is found through the link to the component identified as “scap\_id\_comp\_oval1”. Similarly, when  
 527 the scap\_id\_comp\_dict1 component references “dict-oval-comp” the component reference is resolved  
 528 through the link to the component identified as “scap\_id\_comp\_oval1”. This approach associates SCAP  
 529 components within a data stream at the SCAP logical level, allowing components to be reused across data  
 530 streams within the same data stream collection. This reuse can be accomplished irrespective of how  
 531 references are made within a given component.

532 The following is a stripped down example of the source data stream. The details are covered later in this  
 533 specification.

```

534 <ds:data-stream-collection id="scap_datastream_collection_1" schematron-version="1.3">
535   <ds:data-stream id="scap_id_datastream_ds1" scap-version="1.3" use-
536   case="CONFIGURATION">
537     <ds:dictionaries>
538       <ds:component-ref id="scap_id_cref_ref1" xlink:href="#scap_id_comp_dict1">
539         <cat:catalog>
540           <cat:uri name="dict-oval-comp" uri="#scap_id_cref_ref3"/>
541         </cat:catalog>
542       </ds:component-ref>
543     </ds:dictionaries>
544     <ds:checklists>
545       <ds:component-ref id="scap_id_cref_ref2" xlink:href="#scap_id_comp_xccdf1">
    
```

```

546     <cat:catalog>
547       <cat:uri name="sample-oval-comp" uri="#scap_id_cref_ref3"/>
548     </cat:catalog>
549   </ds:component-ref>
550 </ds:checklists>
551 <ds:checks>
552   <ds:component-ref id="scap_id_cref_ref3" xlink:href="#scap_id_comp_ovall"/>
553 </ds:checks>
554 </ds:data-stream>
555 <ds:component id="scap_id_comp_xccdf1" timestamp="2016-01-22T14:00:00">
556   <xccdf:Benchmark id="xccdf_gov.nist_benchmark_SCAP13" style="SCAP_1.3">
557     ...
558     <xccdf:Rule id="xccdf_gov.nist_rule_id-001">
559       <xccdf:check system="http://oval.mitre.org/XMLSchema/oval-definitions-5">
560         <xccdf:check-content-ref href="sample-oval-comp" name="oval:gov.nist:def:1"/>
561       </xccdf:check>
562     </xccdf:Rule>
563   </xccdf:Benchmark>
564 </ds:component>
565 <ds:component id="scap_id_comp_ovall" timestamp="2016-01-22T14:00:00">
566   <oval-def:oval_definitions>...</oval-def:oval_definitions>
567 </ds:component>
568 <ds:component id="scap_id_comp_dict1" timestamp="2016-01-22T14:00:00">
569   <cpe2-dict:cpe-list>
570     <cpe2-dict:cpe-item name="cpe:/a:oracle:database_server:11.1.0.6.0::enterprise">
571       <cpe2-dict:check href="dict-oval-comp"
572         system="http://oval.mitre.org/XMLSchema/oval-definitions-5">
573         oval:gov.nist:def:2</cpe2-dict:check>
574       <cpe-dict-ext:cpe23-item
575         name="cpe:2.3:a:oracle:database_server:11.1.0.6.0::-::-enterprise::-::-"/>
576     </cpe2-dict:cpe-item>
577   </cpe2-dict:cpe-list>
578 </ds:component>
579 </ds:data-stream-collection>

```

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

- 581 1. Individual components may be developed outside of an SCAP data stream where the binding to  
582 other components is not necessarily known at the time the component is created.
- 583 2. The SCAP source data stream creates a binding between different components that were not  
584 necessarily designed to reference each other. For example, XCCDF was not designed to reference  
585 a particular checking system; it can reference OVAL, OCIL, and other checking systems.
- 586 3. The logical link mapping in the data stream places a layer of capability within the data stream to  
587 control the dereferencing of URIs within components, creating a complete solution related to  
588 bundling components.
- 589 4. The SCAP source data stream format will be useful in future communication models such as web  
590 services, transport protocols, tasking mechanisms, etc.
- 591 5. The SCAP source data stream format supports more comprehensive validation of component  
592 content, including interrelationships between components.

### 593 3.1.1 Source Data Stream Data Model

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

- 596 • The “Element Name” field indicates the name for the XML element being described. Each  
597 element name has a namespace prefix indicating the namespace to which the element belongs.  
598 See Table 1 for a mapping of namespace prefixes to namespaces.
- 599 • The “Element Definition” field indicates the prose description of the element. The definition field  
600 MAY contain key words as indicated in [RFC2119].
- 601 • The “Properties” field is broken into four subfields:
  - 602 ○ The “Name” column indicates the name of a property that MAY, SHOULD, or MUST be  
603 included in the described element, in accordance with the cardinality indicated in the “Count”  
604 column and any [RFC2119] requirement words in the “Property Definition” column.
  - 605 ○ The “Type” column indicates the REQUIRED data type for the value of the property. There  
606 are two categories of types: literal and element. A literal type indicates the type of literal as  
607 defined in [XMLS]. An element type references the name of another element that ultimately  
608 defines that property.
  - 609 ○ The “Count” column indicates the cardinality of the property within the element. The  
610 property MUST be included in the element in accordance with the cardinality. If a range is  
611 given, and “n” is the upper bound of the range, then the upper limit SHALL be unbounded.
  - 612 ○ The “Property Definition” column defines the property in the context of the element. The  
613 definition MAY contain key words as indicated in [RFC2119].

614 **Table 2: ds:data-stream-collection**

Element Name: ds:data-stream-collection			
Element Definition	The top-level element for a SCAP data stream collection. It contains the data streams and components that comprise this data stream collection, along with any data stream signatures.		
Properties:			
Name	Type	Count	Property Definition
id	literal – ID	1	The identifier for the data stream collection. This identifier MUST be globally unique (see Section 3.1.3).
schematron-version	literal – token	1	The version of the SCAP Requirements Schematron rule set to which the data stream collection conforms.
data-stream	element – ds:data-stream	1-n	An element that represents a single data stream (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 3.11 and 4.8 define the requirements for this element.

616  
617

618

**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 MUST be globally unique (see Section 3.1.3).
use-case	literal – token	1	The use case represented by the data stream. The value MUST be one of the following: CONFIGURATION, VULNERABILITY, INVENTORY, or OTHER. The value selected MUST indicate which type of content is being represented as defined in Section 5. The value “OTHER” is for content that does not correspond to a specific use case; this content MUST be valid according to the requirements defined in Sections 3 and 4.
scap-version	literal – token	1	The targeted SCAP version. The value MUST be 1.3, 1.2, 1.1, or 1.0. The value MUST indicate which version of SCAP the content is conformant with. 1.3 MUST 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.

619

620

621

**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	MUST contain a reference to a dictionary component (a component containing CPE dictionary content).

622

623

**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	MUST contain a reference to a checklist component (a component containing an <code>&lt;xccdf: Benchmark&gt;</code> or an <code>&lt;xccdf: Tailoring&gt;</code> element).

624  
625

**Table 6: ds:checks**

<b>Element Name: ds:checks</b>			
<b>Element Definition</b>	A container element that holds references to one or more check components.		
<b>Properties</b>			
<b>Name</b>	<b>Type</b>	<b>Count</b>	<b>Property Definition</b>
component-ref	element – component-ref	1-n	MUST contain a reference to a check component (a component containing check content). See Section 3.2.4.2 for information on SCAP check system support and requirements.

626  
627  
628

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

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

629  
630  
631

**Table 8: ds:component-ref**

<b>Element Name: ds:component-ref</b>			
<b>Element Definition</b>	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].		
<b>Properties</b>			
<b>Name</b>	<b>Type</b>	<b>Count</b>	<b>Property Definition</b>
id	literal - ID	1	The identifier for the reference. This identifier MUST be globally unique (see Section 3.1.3).
type	literal – xlink:type	0-1	The type of XLink represented. The <i>&lt;ds:component-ref&gt;</i> is constrained to a simple XLink, so the value of this field MUST 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 MUST be in the form '# + componentId (e.g. "#component1"). When referencing external content, the URI MUST be in the form of scheme:[/][user:password@]host[:port][/]path[?query][#fragment] as specified in [RFC3986] and MUST 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 <i>&lt;ds:component-ref&gt;</i> , and where those URIs should map to within the context of this data stream. See Table 9.

632

633

**Table 9: cat:catalog**

Element Name: cat:catalog			
<b>Element Definition</b>	A catalog element defined by the OASIS XML Catalog specification [XMLCAT]. Within an SCAP source data stream this element SHALL contain one or more <code>&lt;cat:uri&gt;</code> and/or <code>&lt;cat:rewriteURI&gt;</code> elements, and it SHALL NOT contain any other elements or attributes. Refer to Section 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 MUST be provided)	Maps a reference in the enclosing <code>&lt;ds:component-ref&gt;</code> element's component to some other <code>&lt;ds:component-ref&gt;</code> element that MUST be used to resolve the reference. See Table 10.
rewriteURI	element – cat:rewriteURI	0-n (at least 1 of this or uri MUST 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 <code>&lt;ds:component-ref&gt;</code> element's component to some other <code>&lt;ds:component-ref&gt;</code> element that MUST be used to resolve the reference. See Table 11.

634

635

**Table 10: cat:uri**

Element Name: cat:uri			
<b>Element Definition</b>	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 <code>&lt;ds:component-ref&gt;</code> element's component to some other <code>&lt;ds:component-ref&gt;</code> element that MUST be used to resolve the reference. A <code>&lt;cat:uri&gt;</code> element SHALL have a <code>@name</code> attribute and a <code>@uri</code> attribute.		
Properties			
Name	Type	Count	Property Definition
name	literal – xs:anyURI	1	The <code>@name</code> attribute is the source of the mapping and MUST contain a URI that matches a “referenced URI” in the data stream component referenced by the <code>&lt;ds:component-ref&gt;</code> 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 <code>@uri</code> attribute is the destination of the mapping and MUST be populated with the value “#” + <code>@id</code> of a <code>&lt;ds:component-ref&gt;</code> . When resolving the URI in the <code>@name</code> attribute, the <code>&lt;ds:component-ref&gt;</code> pointed to by the <code>@uri</code> attribute SHALL be used.

636

637

638

**Table 11: cat:rewriteURI**

Element Name: cat: rewriteURI			
<b>Element Definition</b>	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 <i>&lt;ds:component-ref&gt;</i> element's component to some other <i>&lt;ds:component-ref&gt;</i> element that MUST be used to resolve the reference. A <i>&lt;cat:rewriteURI&gt;</i> element SHALL have a <i>@uriStartString</i> attribute and a <i>@rewritePrefix</i> attribute specified. See [XMLCAT] for more details.		
Properties			
Name	Type	Count	Property Definition
uriStartString	literal – xs:anyURI	1	The <i>@uriStartString</i> attribute SHALL be populated with the start of a URI of an external link specified within the component referenced by this element's enclosing <i>&lt;ds:component-ref&gt;</i> element that is to be replaced.
rewritePrefix	literal – xs:anyURI	1	The <i>@rewritePrefix</i> attribute SHALL be populated with a string that will replace the matched <i>@uriStartString</i> value. The resulting URI MUST be used to resolve the link.

639

640

**Table 12: ds:component**

Element Name: ds:component			
<b>Element Definition</b>	A container for a single component. The types of components are defined in Section 3.1.2.		
Properties			
Name	Type	Count	Property Definition
id	literal – ID	1	The identifier for the component. This identifier MUST be globally unique (see Section 3.1.3).
timestamp	literal – dateTime	1	Indicates when the <i>&lt;ds:component&gt;</i> was created or last updated.
Benchmark	element – xccdf:Benchmark	1, and only 1, of these elements	XCCDF benchmark
oval_definitions	element – oval-def:oval_definitions		OVAL Definitions
ocil	element – ocil:ocil		OCIL questionnaire
cpe-list	element – cpe2-dict:cpe-list		CPE dictionary
Tailoring	element – xccdf:Tailoring		XCCDF tailoring

641

642

643

**Table 13: ds:extended-component**

Element Name: ds:extended-component			
<b>Element Definition</b>	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 <code>&lt;ds:extended-component&gt;</code> element SHALL make the data stream non-conformant with SCAP.		
<b>Properties</b>			
Name	Type	Count	Property Definition
id	literal – ID	1	The identifier for the component. This identifier MUST be globally unique (see Section 3.1.3).
timestamp	literal – dateTime	1	Indicates when the <code>&lt;ds:extended-component&gt;</code> was created or last updated.

644 **3.1.2 Source Data Stream Collection Validation**

645 The SCAP source data stream collection SHALL validate against the XML schema representation for the  
 646 source data stream, as well as all Schematron rules embedded within that schema. The SCAP components  
 647 referenced by each `<ds:component>` and `<ds:extended-component>` element SHALL validate  
 648 against the corresponding component schema and its embedded Schematron rules. All of the SCAP-  
 649 related schemas are referenced at <http://scap.nist.gov/revision/1.3/#schema>. See Section 2 in NIST SP  
 650 800-126A for a list of SCAP component schema and Schematron file locations.

651 Each SCAP source data stream component SHALL use one of the elements specified in Table 14 as its  
 652 document element. Each SCAP source data stream component SHOULD NOT use any constructs that are  
 653 deprecated in its associated specification. While Section 4.1 requires that products support deprecated  
 654 constructs, these constructs should be avoided to minimize the impact to content use when these  
 655 constructs are removed from future revisions of the associated specifications. Any component in a data  
 656 stream collection SHALL be referenced not more than once by any data stream in that collection.

657

**Table 14: SCAP Source Data Stream Component Document Elements**

Component	Document Element
XCCDF Benchmark	<code>&lt;xccdf: Benchmark&gt;</code>
XCCDF Tailoring	<code>&lt;xccdf: Tailoring&gt;</code>
OVAL	<code>&lt;oval-def: oval_definitions&gt;</code>
OCIL	<code>&lt;ocil: ocil&gt;</code>
CPE Dictionary	<code>&lt;cpe2-dict: cpe-list&gt;</code>

658

659 NIST provides an SCAP Content Validation Tool, which is designed to help validate the correctness of  
 660 SCAP data streams.<sup>12</sup> The SCAP Content Validation Tool is a command-line tool that will check that  
 661 SCAP source and result content is well-formed, cross references are valid, and required values are  
 662 appropriately set. Errors and warnings are returned in both XML and Hypertext Markup Language  
 663 (HTML) formats. Validation of each SCAP source data stream component SHALL be done in accordance  
 664 with the portions of this document that define requirements for the associated component specification.

665 If applicable, each component MUST validate against its associated Schematron stylesheet. For the SCAP  
 666 source data stream collection, it MUST validate against the version of the SCAP Schematron rules as  
 667 specified on the `<ds:data-stream-collection>` element’s `@schematron-version` attribute,  
 668 and it SHOULD also validate against the latest Schematron rules. NIST provides and maintains a set of

<sup>12</sup> The tool can be downloaded from <http://scap.nist.gov/revision/1.3/#tools>.

669 Schematron rules to check well-formed SCAP content. The Schematron files for the SCAP specification  
 670 and its applicable component specifications are located at <http://scap.nist.gov/revision/1.3/#schematron>.  
 671 Source content SHOULD pass all Schematron assertions in the Schematron rule files. When creating  
 672 source content, failed assertions with a “warning” flag MAY be disregarded if the assertion discovers an  
 673 issue in the content that is justifiable and expected based on the needs of the content author. When  
 674 executing source content, all failed assertions with a “warning” flag MUST be disregarded.

675 The Schematron rule sets are interpretations of the specifications, and the implementations of their rules  
 676 are subject to change. Whenever a change is made to a Schematron file, the SCAP errata document will  
 677 be updated and the new Schematron file will be posted. The latest Schematron file SHOULD be used in  
 678 place of any earlier versions. If the latest file is unavailable, the version specified on the `<ds:data-  
 679 stream-collection>` element’s `@schematron-version` attribute SHALL be used instead.  
 680 Also, for the component specifications, the Schematron file on the SCAP website SHALL be used in  
 681 place of any corresponding Schematron file available elsewhere. For example, a particular specification  
 682 may have an official Schematron file available on a different website. In most cases, the copy on the  
 683 SCAP website will be the same, but if issues in a Schematron file are discovered, the SCAP website may  
 684 address these before the individual specification’s maintainers do.

685 **3.1.3 Globally Unique Identifiers**

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

694 **Table 15: Element Identifier Format Convention**

Element	Identifier Format Convention
<code>&lt;ds:data-stream-collection&gt;</code>	<code>scap_namespace_collection_name</code>
<code>&lt;ds:data-stream&gt;</code>	<code>scap_namespace_datastream_name</code>
<code>&lt;ds:component-ref&gt;</code>	<code>scap_namespace_cref_name</code>
<code>&lt;ds:component&gt;</code>	<code>scap_namespace_comp_name</code>
<code>&lt;ds:extended-component&gt;</code>	<code>scap_namespace_ecomp_name</code>

695 **3.2 Extensible Configuration Checklist Description Format (XCCDF)**

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

701 **3.2.1 General**

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

704 Descriptive information within XCCDF MAY be used by SCAP products to assist in the selection of the  
 705 appropriate SCAP data stream, ensure that the most recent or correct version of an XCCDF document is

706 used, and provide additional information about the document. The following requirements and  
707 conventions apply to the `<xccdf: Benchmark>`, `<xccdf: Profile>`, `<xccdf: Value>`,  
708 `<xccdf: Group>`, and `<xccdf: Rule>` elements:

- 709 1. One or more instances of the `<xccdf: title>` element SHALL be provided. Each instance  
710 MUST contain a text value that briefly indicates the purpose of the containing element.
- 711 2. One or more instances of the `<xccdf: description>` element SHALL be provided. Each  
712 instance MUST contain a text value that describes the purpose of the containing element.

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

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

### 716 3.2.2 The `<xccdf: Benchmark>` Element

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

- 718 1. The `<xccdf: version>` element and the `@id` attribute SHALL be used together to uniquely  
719 identify all revisions of a benchmark.
  - 720 a. Multiple revisions of a single benchmark SHOULD have the same `@id` attribute value and  
721 different `<xccdf: version>` element values, so that someone who reviews the revisions  
722 can readily identify them as multiple versions of a single benchmark.
  - 723 b. Multiple revisions of a single benchmark SHOULD have `<xccdf: version>` element  
724 values that indicate the revision sequence, so that the history of changes from the original  
725 benchmark can be determined.
  - 726 c. The `@time` attribute of the `<xccdf: version>` element SHOULD be used for a  
727 timestamp of when the benchmark was defined.
- 728 2. The `@update` attribute of the `<xccdf: version>` element SHOULD be used for a URI that  
729 specifies where updates to the benchmark can be obtained.
- 730 3. The `<xccdf: Benchmark>` element SHALL have an `@xml: lang` attribute.
- 731 4. The `@style` attribute SHOULD have the value "SCAP\_1.3".
- 732 5. The `<xccdf: status>` element SHALL indicate the current status of the benchmark  
733 document. The associated text value SHALL be "draft" for documents released in public draft  
734 state and "accepted" for documents that have been officially released by an organization. The  
735 `@date` attribute SHALL be populated with the date of the status change. Additional  
736 `<xccdf: status>` elements MAY be included to indicate historic status transitions.
- 737 6. The `<xccdf: metadata>` element SHALL be provided and SHALL, at minimum, contain the  
738 Dublin Core [DCES] terms from Table 16. If provided, additional Dublin Core terms SHALL  
739 follow the required terms within the element sequence.

740

**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.

741 **3.2.3 The <xccdf:Profile> Element**

742 As stated in the XCCDF specification, the use of an <xccdf:Profile> element is not required. SCAP  
 743 content commonly includes <xccdf:Profile> elements, so people tend to assume that they are  
 744 required, but they are optional.

745 Use of the <xccdf:set-complex-value> element within the <xccdf:Profile> element  
 746 SHALL NOT be allowed.

747 **3.2.4 The <xccdf:Rule> Element**

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

751 **3.2.4.1 The <xccdf:ident> Element**

752 Each <xccdf:Rule> element SHALL include an <xccdf:ident> element containing a CVE, CCE,  
 753 or CPE identifier reference if an appropriate identifier exists. The meaning of the identifier MUST be  
 754 consistent with the recommendation implemented by the <xccdf:Rule> element. If the rule references  
 755 an OVAL Definition, then <xccdf:ident> element content SHALL match the corresponding CVE,  
 756 CCE, or CPE identifier found in the associated OVAL Definition(s) if an appropriate identifier exists and  
 757 if that OVAL Definition is the only input to the rule's final result.

758 When referencing a CVE, CCE, or CPE identifier, an <xccdf:Rule> element MUST have a purpose  
 759 consistent with one of the rows in Table 15. Based on the purpose of the <xccdf:Rule> element, the  
 760 <xccdf:Rule> SHALL define its <xccdf:ident> element's @system attribute using the  
 761 corresponding value from Table 15. Also, if the <xccdf:Rule> element references an OVAL  
 762 Definition, it SHALL reference an OVAL Definition of the specified class.

763

**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
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

764

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

```
766 <xccdf:Rule id="xccdf_gov.nist.fdcc.xp_value_AuditAccountLogonEvents">
767     ...
768     <xccdf:ident system="http://cce.mitre.org">CCE-3867-0</xccdf:ident>
769     ...
770 </xccdf:Rule>
```

771

772 See Section 4.5.1 for information on the meaning of a “pass/fail” rule result relating to each of the  
 773 identifier types in Table 15. All rules that contain CCE, CPE, or CVE entries in their `<xccdf:ident>`  
 774 elements MUST obey these meanings. As a result, such `<xccdf:ident>` elements MUST only be  
 775 included either if the recommendation is identical to these associated meanings or if they have a  
 776 `@con:negate` attribute (as described in Section 4.5.1) set to comply with the intended meaning (by  
 777 default, `@con:negate` is set to false). In SCAP, an `<xccdf:ident>` element is not simply a  
 778 reference to related material – it is a declaration of exact alignment with the described meanings.

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

### 782 3.2.4.2 The `<xccdf:check>` Element

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

- 784 1. The `<xccdf:check-content>` element SHALL NOT be used to embed check content  
 785 directly into XCCDF content.
- 786 2. At least one `<xccdf:check-content-ref>` element MUST be provided for each  
 787 `<xccdf:check>` element.
- 788 3. When evaluating an `<xccdf:check-content-ref>` element within an `<xccdf:check>`  
 789 element, its `@href` attribute either MUST contain a “#” + `@id` of a `<ds:component-ref>`  
 790 element or MUST be resolved in the context of the XML Catalog specified as part of the  
 791 `<ds:component-ref>` element that is referencing this benchmark. In either case, the `@href`  
 792 attribute MUST ultimately resolve to a `<ds:component-ref>` element in the data stream  
 793 referencing the benchmark containing this `<xccdf:check-content-ref>` element. See  
 794 Section 3.1.1 for additional information on `<ds:component-ref>` resolution.

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

- 797 1. OVAL check system
  - 798 i. Use of the OVAL check system SHALL be indicated by setting the `<xccdf:check>`  
 799 element’s `@system` attribute to “`http://oval.mitre.org/XMLSchema/oval-`  
 800 `definitions-5`”.
  - 801 ii. The `@href` attribute in the `<xccdf:check-content-ref>` element MUST reference  
 802 an OVAL source data stream component using the `<ds:component-ref>` approach  
 803 defined above.
  - 804 iii. Use of the `@name` attribute in the `<xccdf:check-content-ref>` element is  
 805 OPTIONAL. If present, it MUST reference an OVAL Definition in the designated OVAL  
 806 source data stream component, otherwise see Section 4.5.2 for information on use of the  
 807 `@multi-check` attribute.
- 808 2. OCIL check system
  - 809 i. OCIL questionnaires SHOULD NOT be used if OVAL can perform the same check  
 810 correctly.
  - 811 ii. Use of the OCIL check system SHALL be indicated by setting the `<xccdf:check>`  
 812 element’s `@system` attribute to “`http://scap.nist.gov/schema/ocil/2`”.
  - 813 iii. The `@href` attribute in the `<xccdf:check-content-ref>` element MUST reference  
 814 an OCIL source data stream component using the `<ds:component-ref>` approach  
 815 defined above.

- 816           iv. Use of the *@name* attribute in the `<xccdf:check-content-ref>` element is  
 817           OPTIONAL. If present, it MUST reference an OCIL questionnaire in the designated OCIL  
 818           source data stream component, otherwise see Section 4.5.2 for information on use of the  
 819           *@multi-check* attribute.  
 820           v. Follow the additional requirements in Appendix B of NIST IR 7692, *Specifications for the*  
 821           *Open Checklist Interactive Language (OCIL) Version 2.0* [OCIL].

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

### 827 3.2.4.3 Use of a Patches Up-To-Date Rule

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

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

- 834           1. The source data stream MUST include the OVAL source data stream component referenced by  
 835           the patches up-to-date rule, which contains one or more OVAL patch class definitions.
- 836           2. The `<xccdf:Rule>` element that references an OVAL source data stream component SHALL  
 837           have the *@id* attribute value of “*xccdf\_NAMESPACE\_rule\_security\_patches\_up\_to\_date*”, where  
 838           *NAMESPACE* is the reverse DNS format namespace associated with the content maintainer.
- 839           3. Each `<xccdf:check-content-ref>` element SHALL omit the *@name* attribute.
- 840           4. The *@multi-check* attribute of the `<xccdf:check>` element SHALL be set to “true”. This  
 841           causes a separate `<xccdf:rule-result>` to be generated for each OVAL Definition. See  
 842           Section 4.5.2 for more information.

843 Here is a patches up-to-date rule example that references numerous OVAL patch class definitions:

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

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

- 857           1. The source data stream MUST include the OVAL source data stream component referenced by  
 858           the patches up-to-date rule, which contains one or more OVAL patch class definitions, and/or  
 859           other class definitions.

- 860 2. The `<xccdf:Rule>` element that references an OVAL source data stream component SHALL  
 861 have the `@id` attribute value of “`xccdf_NAMESPACE_rule_security_patches_up_to_date`”, where  
 862 `NAMESPACE` is the reverse DNS format namespace associated with the content maintainer.
- 863 3. Each `<xccdf:check-content-ref>` element SHALL refer to the single OVAL definition  
 864 performing the patches up-to-date check.
- 865 4. The `@multi-check` attribute of the `<xccdf:check>` element SHALL be set to “false”,  
 866 which is the default value.

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

```
868 <xccdf:Rule
869   id="xccdf_gov.nist.usgcb.win_rule_security_patches_up_to_date"
870   selected="true">
871   <xccdf:title>Security Patches Up-To-Date</xccdf:title>
872   <xccdf:description>Keep systems up to current patch
873 levels</xccdf:description>
874   <xccdf:check system="http://oval.mitre.org/XMLSchema/oval-definitions-5" multi-
875 check="false">
876     <xccdf:check-content-ref href="scap-windows-patches"
877 name="oval:gov.nist.usgcb.win.patch:def:10101" />
878
879   </xccdf:check>
880 </xccdf:Rule>
```

#### 881 3.2.4.4 CVSS and CCSS Scores

882 SCAP 1.0 required the inclusion of static CVSS scores in XCCDF vulnerability-related rules. However,  
 883 CVSS base scores sometimes change over time, such as when more information is available about a  
 884 particular vulnerability, and CVSS temporal and environmental scores are intended to change to reflect  
 885 current threats, security controls, and other factors. During scoring, current CVSS scores acquired  
 886 dynamically, such as from a data feed, SHOULD be used in place of the `@weight` attribute within  
 887 XCCDF vulnerability-related rules. Section 3.9 contains additional requirements for CVSS usage.

888 CCSS scores are more stable than CVSS scores, but they still may change over time. Accordingly, during  
 889 scoring, current CCSS scores acquired dynamically, such as from a data feed, MAY be used in place of  
 890 the `@weight` attribute within XCCDF configuration setting-related rules. Section 3.10 contains  
 891 additional requirements for CCSS usage.

#### 892 3.2.5 The `<xccdf:Value>` Element

893 Use of the `<xccdf:source>`, `<xccdf:complex-value>`, and `<xccdf:complex-default>`  
 894 elements within the `<xccdf:Value>` element SHALL NOT be allowed. Within the  
 895 `<xccdf:choices>` element of the `<xccdf:Value>` element, use of the `<xccdf:complex-`  
 896 `choice>` element SHALL NOT be allowed.

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

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

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

```

903 <xccdf:check system="http://oval.mitre.org/XMLSchema/oval-definitions-5">
904   <xccdf:check-export value-id="xccdf_gov.nist.fdcc.xp_value_NoSlowLink"
905   export-name="oval:gov.nist.fdcc.xp:var:66711" />
906   <xccdf:check-export value-id="xccdf_gov.nist.fdcc.xp_value_NoBackgroundPolicy"
907   export-name="oval:gov.nist.fdcc.xp:var:66712" />
908   <xccdf:check-export value-id="xccdf_gov.nist.fdcc.xp_value_NoGPOListChanges"
909   export-name="oval:gov.nist.fdcc.xp:var:66713" />
910   <xccdf:check-content-ref href="fdcc-winxp-oval.xml"
911   name="oval:gov.nist.fdcc.xp:def:6671" />
912 </xccdf:check>
913

```

914 The type and value binding of the specified `<xccdf:Value>` is constrained to match that lexical  
 915 representation of the indicated OVAL Variable data type. Table 18 summarizes the constraints regarding  
 916 data type usage. Additional information regarding OVAL data types can be found in the OVAL Language  
 917 documentation<sup>13</sup> and the XCCDF specification [XCCDF]. Additional information on OVAL data types  
 918 may also be added to the SCAP 1.3 annex document, NIST SP 800-126A.

919 **Table 18: XCCDF-OVAL Data Export Matching Constraints**

920 OVAL Variable Data Type	921 Matching XCCDF Data Type
922 int	923 number
924 float	925 number
926 boolean	927 boolean
928 string, evr_string, version, ios_version, fileset_revision, binary	929 string

930 **3.2.6 The <xccdf:Group> Element**

931 XCCDF group extension SHALL NOT be allowed.

932 **3.3 Open Vulnerability and Assessment Language (OVAL)**

933 This section lists requirements and recommendations for using the Open Vulnerability and Assessment  
 934 Language (OVAL) to express an OVAL component of an SCAP source data stream (see Table 14).

935 See the SCAP 1.3 annex document, NIST SP 800-126A, for requirements regarding which OVAL  
 936 component specification (core schema) versions and platform schema versions shall or may be used in  
 937 SCAP 1.3 content.

938 Because SCAP 1.3 supports the use of multiple OVAL source data stream components, an SCAP content  
 939 creator could choose to divide the OVAL Definitions into multiple components. For example, a content  
 940 creator could create one OVAL source data stream component for one version of OVAL definitions and  
 941 another for a second version of OVAL definitions if both versions are supported by SCAP 1.3. SCAP 1.3  
 942 also supports multiple types of OVAL Definitions within a single OVAL source data stream component;  
 943 for example, a benchmark could reference OVAL compliance and vulnerability definitions contained in a  
 944 single data stream component.

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

---

<sup>13</sup> <https://github.com/OVALProject/Language/blob/5e853a3867184144284f72bd5b00be6e8c379799/specifications/oval-language-specification.docx> in section 4.2.7

```

938 <oval:generator>
939   <oval:product_name>The OVAL Repository</oval:product_name>
940   <oval:schema_version>5.11</oval:schema_version>
941 </oval:generator>

```

942  
 943 The version that is specified using the `<oval:schema_version>` content element SHALL  
 944 correspond to the version specified by the `<xsi:schemaLocator>` value for the OVAL schema.

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

950 Required values for the `@class` attribute of an OVAL Definition are as follows:

- 951 1. “compliance” if it represents a check for the system’s configuration complying with policy  
 952 requirements (for example, having the required value for a specific configuration setting).
- 953 2. “vulnerability” if it represents a check for the presence of a particular software flaw vulnerability  
 954 on a system.
- 955 3. “patch” if it represents a check for whether a discrete patch needs to be installed on the system.
- 956 4. “inventory” if it represents a check for the presence of a product of interest on the system.

957 The following requirements apply to particular classes of OVAL Definitions:

- 958 1. For compliance class definitions:
  - 959 a. If an OVAL compliance class definition maps to one or more CCE identifiers, the definition  
 960 SHOULD include `<oval-def:reference>` elements that reference those identifiers  
 961 using the following format:
 

```

          962 <oval-def:reference source="http://cce.mitre.org"
          963 ref_id="CCE_identifier"/>
          964
          965 The source attribute SHALL be defined using either “http://cce.mitre.org” (preferred
          966 method) or “CCE”.
        
```
  - 967 b. Definitions that are directly or indirectly extended SHALL be limited to inventory and  
 968 compliance classes.
- 969 2. For inventory class definitions:
  - 970 a. If an OVAL inventory class definition maps to one or more CPE identifiers, the definition  
 971 SHOULD include `<oval-def:reference>` elements that reference those identifiers  
 972 using the following format:
 

```

          973 <oval-def:reference source="http://cpe.mitre.org"
          974 ref_id="CPE_identifier"/>
          975
          976 The source attribute SHALL be defined using either “http://cpe.mitre.org” (preferred
          977 method) or “CPE”.
        
```
  - 978 b. Definitions that are directly or indirectly extended SHALL be limited to the inventory class.
- 979 3. For patch class definitions:

980

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

```
984
985 <oval-def:reference source="www.microsoft.com/Patch"
986 ref_id="KB912919"/>
```

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

```
990
991 <oval-def:reference source="http://cve.mitre.org"
992 ref_id="CVE_identifier"/>
```

993  
 994 This recommendation is weaker than its counterparts for the other class definition types  
 995 because a CVE identifier is not an identifier for a patch; it is more of an association. For  
 996 example, one patch could fix multiple vulnerabilities, so it would map to multiple CVE  
 997 identifiers.

998  
 999 The source attribute SHALL be defined using either “*http://cve.mitre.org*” (preferred  
 1000 method) or “CVE”.

- 1001 c. Definitions that are directly or indirectly extended SHALL be limited to inventory and patch  
 1002 classes.

1003 4. For vulnerability class definitions:

- 1004 a. If an OVAL vulnerability class definition maps to one or more CVE identifiers, the definition  
 1005 SHOULD include `<oval-def:reference>` elements that reference those identifiers  
 1006 using the following format:

```
1007
1008 <oval-def:reference source="http://cve.mitre.org"
1009 ref_id="CVE_identifier"/>
```

1010  
 1011 The source attribute SHALL be defined using either “*http://cve.mitre.org*” (preferred  
 1012 method) or “CVE”.

- 1013 b. Definitions that are directly or indirectly extended SHALL be limited to inventory and  
 1014 vulnerability classes.

### 1015 3.4 Open Checklist Interactive Language (OCIL)

1016 This section lists recommendations for using the Open Checklist Interactive Language (OCIL) to express  
 1017 an OCIL component of an SCAP source data stream (see Table 14).

1018 OCIL content SHOULD be used for checking rules that cannot be fully automated with OVAL. For  
 1019 example, a particular software product may not have an application programming interface (API) that  
 1020 supports OVAL use. Another example is performing a check that requires user interaction, such as asking  
 1021 the user to look up information within a management console or to report a serial number affixed to a  
 1022 computing device. OCIL can also be used to collect a user’s own information, such as whether the user  
 1023 participated in a recent security training session.

1024 If an `<ocil:questionnaire>` element maps to one or more CCE, CVE, and/or CPE identifiers, it  
 1025 SHOULD include `<ocil:reference>` elements that reference those identifiers using the  
 1026 corresponding following format:

1027 <ocil:reference href="http://cve.mitre.org">CCE\_identifier</ocil:reference>  
 1028  
 1029 <ocil:reference href="http://cve.mitre.org">CVE\_identifier</ocil:reference>  
 1030  
 1031 <ocil:reference href="http://cpe.mitre.org">CPE\_identifier</ocil:reference>

### 1032 3.5 Common Platform Enumeration (CPE)

1033 This section lists requirements and recommendations for using Common Platform Enumeration (CPE) to  
 1034 express a CPE component of an SCAP source data stream (see Table 14).

1035 The Official CPE Dictionary data feed<sup>14</sup> MAY be used by SCAP components to reference CPE names. If  
 1036 use of the Official CPE Dictionary is impractical, a subset of the dictionary MAY be used instead.  
 1037 Creating the reduced official dictionary involves first identifying every CPE in <xccdf:platform>  
 1038 and <cpe2:fact-ref> elements contained within referenced <cpe2:platform-  
 1039 specification> elements in every benchmark in the data stream. Then these CPEs MUST be  
 1040 matched against every entry in the Official CPE Dictionary using the CPE name matching algorithm  
 1041 [CPE-M]. All CPEs matched in the official dictionary with a result of EQUAL or SUPERSET MUST be  
 1042 included in the reduced official dictionary.

1043 One or more third-party dictionaries MAY be included in a data stream as well. All such third-party  
 1044 dictionaries SHOULD follow the requirements of the CPE Dictionary specification [CPE-D]. If including  
 1045 an entire third-party dictionary is impractical, a subset of the dictionary MAY be used instead. The  
 1046 reduced dictionary MUST be created using the same procedure outlined for creating a subset of the  
 1047 official dictionary.

1048 In all cases, a dictionary component MAY be remote to the data stream collection.

1049 Each CPE name [CPE-N] in an <xccdf:platform> or <cpe2:fact-ref> element within an  
 1050 XCCDF document SHALL match at least one CPE entry in a dictionary referenced by the data stream. A  
 1051 match is considered an EQUAL or SUPERSET result when matching the CPE name to a dictionary entry,  
 1052 as defined in the CPE Name Matching specification [CPE-M]. Only non-deprecated names SHOULD be  
 1053 used.

1054 Checklist authors SHOULD ensure that each CPE name [CPE-N] they specify in an  
 1055 <xccdf:platform> or <cpe2:fact-ref> element within an XCCDF document has a check  
 1056 associated with its CPE name. If a corresponding check does not exist, then it will not be possible to fully  
 1057 detect the presence of the product and determine platform applicability. Because there may be a lag  
 1058 between the time that a new product is available and the Official CPE Dictionary is updated to include a  
 1059 CPE name for that product, third-party dictionaries would need to be used to compensate for the lag.

1060 [CPE-D] provides the defining structure of a CPE dictionary. A <cpe2\_dict:cpe-item> element  
 1061 MAY contain one or more <cpe2\_dict:check> elements that reference OVAL inventory class  
 1062 definitions using the following format:

```
1063 <cpe2_dict:check system="http://oval.mitre.org/XMLSchema/oval-definitions-5"  
1064     [href="oval_URL"]>oval_inventory_definition_id</cpe2_dict:check>
```

1065 For example:

```
1066 <cpe2_dict:cpe-list xmlns="http://cpe.mitre.org/dictionary/2.0"  
1067     xmlns:cpe2_dict="http://cpe.mitre.org/dictionary/2.0">  
1068   <cpe2_dict:cpe-item  
1069     name="cpe:/a:sun:java_system_messaging_server:6.2::-:sparc">  
1070     <cpe2_dict:title>Sun Java System Messaging Server 6.2 sparc</title>
```

<sup>14</sup> The Official CPE Dictionary is located at <http://nvd.nist.gov/cpe.cfm>.

```

1071     <cpe2_dict:check
1072         system=http://oval.mitre.org/XMLSchema/oval-definitions-5
1073         href="example-sunjavamsg62-oval.xml">oval:org.mitre.oval:def:128
1074     </cpe2_dict:check>
1075     <cpe-dict-ext:cpe23-item
1076         name="cpe:2.3:a:sun:java_system_messaging_server:6.2:-:-:-:-:sparc:-"/>
1077     </cpe2_dict:cpe-item>
1078 </cpe2_dict:cpe-list>

```

1079  
1080 The referenced OVAL inventory class definition SHALL specify the technical procedure for determining  
1081 whether or not a specific target asset is an instance of the CPE name specified by the  
1082 `<cpe2_dict:cpe-item>` element. This usage is encouraged for CPE components.

1083 When creating a subset of the Official CPE Dictionary or a third-party dictionary, a  
1084 `<cpe2_dict:check>` element on an entry MAY be added or modified if the existing check does not  
1085 provide satisfactory content to test the presence of the CPE name.

1086 If a `<cpe2_dict:cpe-item>` element contained in a CPE component references an OVAL inventory  
1087 class definition, then that definition SHALL be resolved by an `@href` attribute referencing an OVAL  
1088 source data stream component in the same data stream.

### 1089 3.6 Software Identification (SWID) Tags

1090 The syntax and construction of a SWID tag is defined in ISO/IEC 19770-2:2015 [SWID] and is further  
1091 refined in NISTIR 8060 [SWID-CYBER]. For a software product that has an associated SWID tag, this  
1092 tag should have been installed along with the software product. SWID tags can also exist for software  
1093 patches. For software patches that have an associated SWID tag, this tag is expected to be installed along  
1094 with the patch. When made available in these ways, a SWID tag provides evidence of the installation of a  
1095 software product or patch.

1096 A SWID tag installed on a target asset SHALL be identified by an OVAL inventory class definition. The  
1097 definition SHOULD use the `<independent-def:xmlfilecontent_object>` to search the file  
1098 system for one or more SWID tags expressed in XML that match a desired XPath expression.

1099 If a SWID tag has been installed on the target endpoint for a software product or patch, then one of the  
1100 following methods SHALL be used to detect the SWID tag on the target asset:

- 1101 1. One or more `<cpe2-dict:check>` elements that reference an OVAL inventory class  
1102 definition that searches for the presence of a matching SWID tag.
- 1103 2. A `<cpe:check-fact-ref>` element that references an OVAL inventory class definition  
1104 that searches for the presence of a matching SWID tag.
- 1105 3. An OVAL definition that references another OVAL inventory class definition using the  
1106 `<oval-def:extend_definition>` element where the extended definition searches for  
1107 the presence of a matching SWID tag.

### 1108 3.7 Common Configuration Enumeration (CCE)

1109 To maintain consistency and accuracy, SCAP content referencing a configuration setting SHALL use the  
1110 official CCE identifier if a CCE entry for a particular configuration setting exists in the official CCE list.  
1111 If no CCE entry exists for the configuration setting of interest, the content author SHOULD seek to have  
1112 a CCE identifier issued for the configuration setting. See the OVAL compliance class definition

1113 requirements in Section 3.3 and the `<xccdf:ident>` requirements in Section 3.2.4.1 for additional  
1114 requirements involving CCE identifier references.

1115 The current official CCE list is available at <https://nvd.nist.gov/cce/index.cfm>, and new CCEs can be  
1116 requested from NIST via email ([cce@nist.gov](mailto:cce@nist.gov)).

1117 Use of an official, dynamic data feed is preferred to static coding of CCE-related supporting information  
1118 in SCAP data sources. For example, NVD provides a data feed<sup>15</sup> that is the authoritative mapping  
1119 between CCE identifiers and the control identifiers defined in NIST SP 800-53. Embedding control  
1120 identifiers within SCAP content is strongly discouraged due to the maintenance burden that it imposes on  
1121 content maintainers when the control identifiers are revised. A preferred technique is to embed only the  
1122 CCE identifiers within SCAP content; when mappings to NIST SP 800-53 control identifiers are needed,  
1123 dynamically acquire them from the official data feed and associate them to the SCAP content based on its  
1124 embedded CCE identifiers.

### 1125 **3.8 Common Vulnerabilities and Exposures (CVE)**

1126 CVE references in SCAP content MAY include both “candidate” and “entry” status identifiers.  
1127 Deprecated CVE identifiers SHALL NOT be used.

1128 If a CVE identifier exists for a particular vulnerability, the official CVE identifier SHALL be used. If no  
1129 CVE exists for the software flaw, an alternate identifier MAY be used, but the user SHOULD seek to  
1130 have a CVE identifier issued for the vulnerability. Information on submitting unpublished vulnerabilities  
1131 and obtaining CVE identifiers is available at [https://cve.mitre.org/cve/request\\_id.html](https://cve.mitre.org/cve/request_id.html).

1132 NIST provides a CVE data feed to support dynamic and current vulnerability information and associated  
1133 metadata (e.g., CVSS values). The current schema is available at <http://nvd.nist.gov/download.cfm>.

### 1134 **3.9 Common Vulnerability Scoring System (CVSS)**

1135 The NIST CVE data feed, discussed in Section 3.8, is one source of CVSS base score and vector data that  
1136 MAY be used by products to support additional use cases built on SCAP usage. In support of these  
1137 additional use cases, CVSS base scores and vectors from this data feed MAY be used by products along  
1138 with temporal and environmental scores and vectors from other sources.

### 1139 **3.10 Common Configuration Scoring System (CCSS)**

1140 CCSS base, temporal, and environmental scores and vectors MAY be used by products. Adopters of  
1141 CCSS should be aware that it has significant differences from CVSS. Unlike CVSS data, which can be  
1142 used by itself to aid in prioritizing vulnerability remediation efforts, CCSS data is not directly useful in  
1143 the same way. Instead, CCSS data needs to be considered in the context of each organization’s security  
1144 policies and in the context of dependencies among vulnerabilities. See [CCSS] for additional information.

### 1145 **3.11 XML Digital Signature**

1146 Digitally signing source data streams is important to ensuring the integrity and trustworthiness of  
1147 legitimate content, while preventing rogue content from being executed. Leveraging the Trust Model for  
1148 Security Automation Data (TMSAD) specification [TMSAD] for SCAP can improve the legitimacy of  
1149 authoritative content and create a more secure environment. As such, content authors MAY digitally sign  
1150 source content following the guidelines in [TMSAD], along with the following requirements.

1151 One or more XML digital signatures MAY be included as the last elements in the SCAP source data  
1152 stream collection root element. Each signature MUST be represented as a `<dsig:Signature>`

---

<sup>15</sup> <http://nvd.nist.gov/cce.cfm>

1153 element and follow the W3C recommendation [DSIG]. Each *<dsig:Signature>* element MUST sign  
1154 only one data stream.

1155 The *<dsig:Signature>* element MUST follow the recommendations in [TMSAD] and these  
1156 additional requirements:

- 1157 1. A *<dsig:Manifest>* element MUST be included within the *<dsig:Signature>* element  
1158 as a *<dsig:Object>* element. The *<dsig:Manifest>* element MUST have a  
1159 *<dsig:Reference>* element for each local component referenced by the data stream being  
1160 signed. External components MAY be omitted from the *<dsig:Manifest>* element. Each  
1161 *<dsig:Reference>* element referencing a *<ds:component>* or *<ds:extended-*  
1162 *component>* element MUST point to the component being signed by identifying the  
1163 component in the *@URI* attribute using “#” + *@Id* of the component.
- 1164 2. A *<dsig:SignatureProperties>* element MUST be included within the  
1165 *<dsig:Signature>* element as a *<dsig:Object>* element. At least one  
1166 *<dsig:SignatureProperty>* element MUST be populated with *<dt:signature-*  
1167 *info>* as specified in [TMSAD].
- 1168 3. The first *<dsig:Reference>* element in a *<dsig:Signature>* element MUST be to the  
1169 *<ds:data-stream>* element being signed. The *<ds:data-stream>* element MUST be  
1170 referenced in the *@URI* attribute using “#” + *@Id* of the *<ds:data-stream>* element.
- 1171 4. The second *<dsig:Reference>* element in a *<dsig:Signature>* element MUST be to  
1172 the *<dsig:SignatureProperties>* element captured in a *<dsig:Object>* element  
1173 within the *<dsig:Signature>* element. The *<dsig:SignatureProperties>* element  
1174 MUST be referenced in the *@URI* attribute using “#” + *@Id* of  
1175 the *<dsig:SignatureProperties>* element.
- 1176 5. The third *<dsig:Reference>* element MUST be to the *<dsig:Manifest>* element  
1177 captured in a *<dsig:Object>* element with the *<dsig:Signature>* element. The  
1178 *<dsig:Manifest>* element MUST be referenced in the *@URI* attribute using “#” + *@Id*  
1179 attribute of the *<dsig:Manifest>* element.
- 1180 6. *<dsig:Reference>* elements on the *<dsig:Manifest>* element SHOULD be in the same  
1181 order as the *<ds:component-ref>* elements on the data stream being signed.
- 1182 7. Key information SHOULD be provided on the *<dsig:Signature>* element.

1183

## 1184 4. SCAP Content Processing Requirements and Recommendations

1185 This section defines the processing requirements that SCAP content consumers MUST follow in order to  
1186 correctly process SCAP 1.3 content. This section also provides recommendations that are not mandatory;  
1187 organizations are encouraged to adopt them to promote stronger interoperability and greater consistency.  
1188 The topics covered in the first part of this section are legacy support, source data streams, and XCCDF  
1189 processing. The end of the section covers result-related topics: SCAP result data streams, XCCDF results,  
1190 OVAL results, OCIL results, and result data stream signing.

### 1191 4.1 Legacy Support

1192 Content consumers supporting SCAP 1.3 SHALL be capable of processing SCAP 1.3, 1.2, and 1.1  
1193 content. Content consumers SHALL process SCAP content as defined under the corresponding version of  
1194 NIST SP 800-126 (for SCAP 1.3, this revision; for SCAP 1.2, revision 2; for SCAP 1.1, revision 1).<sup>16</sup>  
1195 Content consumers that process legacy SCAP content MUST be capable of outputting results in the same  
1196 SCAP version as the source content, and MAY convert the legacy SCAP results into results based on a  
1197 newer SCAP version.

1198 Within the SCAP component specifications, certain constructs may be deprecated.<sup>17</sup> SCAP content  
1199 consumers MUST support all deprecated constructs because they are still valid. This requirement ensures  
1200 that legacy content that made use of these deprecated constructs continues to be supported.

1201 Content consumers supporting OVAL SHALL support OVAL Definition documents written against all  
1202 versions of OVAL component specifications listed in the annex.

### 1203 4.2 Source Data Streams

1204 Content consumers SHALL be capable of validating SCAP content against the appropriate schemas and  
1205 Schematron stylesheets, detecting and reporting errors, and failing gracefully if there are errors. The  
1206 relevant XML schemas are located at <http://scap.nist.gov/revision/1.3/#schema>, and the relevant  
1207 Schematron rule sets at <http://scap.nist.gov/revision/1.3/#schematron>. See Section 3.1 for additional  
1208 information on the Schematron rule sets.

1209 Content consumers SHOULD validate XML digital signatures if they exist in the content. Validating a  
1210 signature includes confirming that the signature value is valid, all of the reference hashes in the signature  
1211 and manifest are correct, and the public key used to verify the signature is from a trusted source. A data  
1212 stream with a signature that does not validate SHOULD NOT be evaluated by a content consumer.

1213 Whenever a `<ds:extended-component>` that is not recognized by the tool is referenced from a  
1214 `<ds:data-stream>`, `<ds:component>`, or `<ds:extended-component>` element, the tool  
1215 SHALL issue a warning.

1216 If more than one `<ds:data-stream>` element is specified on the `<ds:data-stream-`  
1217 `collection>`, the ID of the `<ds:data-stream>` to execute MUST be indicated to the content  
1218 consumer, and the content consumer MUST use the specified `<ds:data-stream>`. If more than one  
1219 `<xccdf:Benchmark>` is referenced by a `<ds:data-stream>`, the ID of the  
1220 `<xccdf:Benchmark>` to execute MUST be indicated to the content consumer, and the content  
1221 consumer MUST process the indicated `<xccdf:Benchmark>`. Because SCAP and its component  
1222 specifications do not formally define how to designate a particular data stream, benchmark, etc. in these  
1223 cases, it is expected that products will implement these capabilities in a proprietary way.

<sup>16</sup> <http://csrc.nist.gov/publications/PubsSPs.html#800-126>

<sup>17</sup> The OVAL Language Deprecation policy is available here: <http://oval.mitre.org/language/about/deprecation.html>

## 1224 4.3 XCCDF Processing

1225 The following requirements and recommendations pertain to content consumers processing XCCDF  
1226 benchmark and tailoring components from an SCAP source data stream.

### 1227 4.3.1 CPE Applicability Processing

1228 CPEs referenced in an `<xccdf:platform>` element directly or by a `<cpe2:fact-ref>` contained  
1229 within a referenced `<cpe2:platform-specification>` element SHALL be evaluated as follows  
1230 to determine their presence on a machine:

- 1231 1. The CPE SHALL be matched against all CPEs in all of the dictionaries referenced by the  
1232 `<ds:data-stream>` element. All CPEs that return an EQUAL or SUPERSET result as  
1233 defined in CPE Name Matching [CPE-M] SHALL be used in evaluating the  
1234 `<xccdf:platform>` or `<cpe2:fact-ref>`.
- 1235 2. Either a list of CPEs found on the target asset MUST be known before the scan, or a list SHALL  
1236 be generated. If a previously known list is used, it MUST be equivalent to a newly generated list.  
1237 To generate the list, the `<cpe2_dict:check>` element data associated with the found  
1238 `<cpe2_dict:cpe-item>` elements SHALL be evaluated against the target using the  
1239 referenced OVAL inventory class definition. If a `<cpe2_dict:check>` returns “pass”, then  
1240 the corresponding CPE SHALL be added to the list of CPEs found on the target.
- 1241 3. The list of CPEs found on the target asset, along with the `<xccdf:platform>` or  
1242 `<cpe2:platform-specification>` SHALL be used as input to the CPE Applicability  
1243 Language [CPE-L] algorithm to determine the XCCDF Benchmark applicability to the target  
1244 asset.

### 1245 4.3.2 Check System Usage

1246 If an XCCDF component has multiple `<xccdf:check-content-ref>` elements, then check  
1247 processing SHALL be performed according to [XCCDF:7.2.3.5.1] with the following changes:

- 1248 1. For each `<xccdf:check-content-ref>` element, a content consumer either MUST attempt  
1249 to retrieve the document referenced by the `<ds:component-ref>` element that is referenced  
1250 directly by the `<xccdf:check-content-ref>` element’s `@href` attribute, or it MUST  
1251 resolve the `@href` attribute within the context of the XML Catalog specified as part of the  
1252 `<ds:component-ref>` element used to reference this benchmark. If not resolvable, the next  
1253 available `<xccdf:check-content-ref>` element SHALL be evaluated. If none of the  
1254 `<xccdf:check-content-ref>` elements are resolvable, then the result of the rule  
1255 evaluation SHALL be the XCCDF “notchecked” status and processing of the check SHALL end.
- 1256 2. Once a resolvable `<xccdf:check-content-ref>` element is found, then check system  
1257 processing SHALL proceed. When evaluating a rule, an `<xccdf:rule-  
1258 result/xccdf:message>` with the `@severity` attribute value of “info” SHALL be  
1259 generated, indicating the `<xccdf:check-content-ref>` `@href` attribute and `@name`  
1260 attribute, if provided.

1261 Content consumers SHALL implement check systems supported by SCAP as defined in Section 3.2.4.2.  
1262 Content consumers MAY implement check systems that are not supported by SCAP. If a tool encounters  
1263 a check system it does not support, it MUST issue a warning and it MUST continue processing according  
1264 to the [XCCDF] specification.

1265 When processing a patches-up-to-date rule, only OVAL patch class definitions SHALL be evaluated; all  
 1266 other classes of definitions (e.g., inventory class definitions) SHALL NOT be evaluated except when they  
 1267 serve, directly or indirectly, as criteria (extended definitions) of patch definitions.

#### 1268 4.4 SCAP Result Data Streams

1269 An SCAP result data stream contains the results of the evaluation of one or more SCAP source data  
 1270 streams by an SCAP content consumer. The following requirements and recommendations pertain to  
 1271 content consumers generating SCAP result data streams.

1272 An SCAP result data stream SHALL conform to the [ARF] specification. The following sections outline  
 1273 the details of the ARF report. In all situations, one or more component results (e.g., XCCDF, check  
 1274 results), the target asset, and/or the SCAP source data stream collection represented as a report request in  
 1275 ARF MAY be represented either as a local component in the ARF or as a remote resource, leveraging the  
 1276 remote resource capability built into ARF. This is a stripped down ARF example:

```

1277 <arf:asset-report-collection>
1278   <rc:relationships>
1279     <rc:relationship type="arf-rel:isAbout" subject="xccdf1">
1280       <rc:ref>asset1</rc:ref>
1281     </rc:relationship>
1282     <rc:relationship type="arf-rel:isAbout" subject="oval1">
1283       <rc:ref>asset1</rc:ref>
1284     </rc:relationship>
1285     <rc:relationship type="scap-rel:checkContext" subject="oval1">
1286       <rc:ref>xccdf1</rc:ref>
1287     </rc:relationship>
1288     <rc:relationship type="scap-rel:fromSource" subject="xccdf1">
1289       <rc:ref>collection1</rc:ref>
1290     </rc:relationship>
1291     <rc:relationship type="scap-rel:fromSource" subject="oval1">
1292       <rc:ref>collection1</rc:ref>
1293     </rc:relationship>
1294   </rc:relationships>
1295   <arf:report-requests>
1296     <arf:report-request id="collection1">
1297       <arf:content>
1298         <ds:data-stream-collection>...</ds:data-stream-collection>
1299       </arf:content>
1300     </arf:report-request>
1301   </arf:report-requests>
1302   <arf:assets>
1303     <arf:asset id="asset1">
1304       <ai:computing-device>...</ai:computing-device>
1305     </arf:asset>
1306   </arf:assets>
1307   <arf:reports>
1308     <arf:report id="xccdf1">
1309       <arf:content>
1310         <xccdf:TestResult>...</xccdf:TestResult>
1311       </arf:content>
1312     </arf:report>
1313     <arf:report id="oval1">
1314       <arf:content>
1315         <xccdf-res:oval-results>...</xccdf-res:oval-results>
1316       </arf:content>
1317     </arf:report>
1318   </arf:reports>
1319 </arf:asset-report-collection>

```

#### 1320 4.4.1 The Component Reports

1321 The ARF report MUST contain a report object for each XCCDF, OVAL, and OCIL component executed  
 1322 when a source data stream is evaluated against a target. It MAY contain additional report objects for other  
 1323 results, such as `<oval-var:oval_variables>` or extended component results. Each component  
 1324 result MUST be captured as a separate `<arf:report>` element in the `<arf:asset-report-`  
 1325 `collection>` element, and when reporting on XCCDF, OVAL, or OCIL, each component report  
 1326 SHALL use the element specified in Table 19 as its root element.

1327 **Table 19: SCAP Result Data Stream Component Document Elements**

Component	Document Element
XCCDF	<code>&lt;xccdf:TestResult&gt;</code>
OVAL	<code>&lt;oval-res:oval_results&gt;</code>
OCIL	<code>&lt;ocil:ocil&gt;</code>

1328  
 1329 Each SCAP result data stream component SHOULD NOT use any constructs that are deprecated in its  
 1330 associated specification. Validation of each component SHALL be done in accordance with the portions  
 1331 of this document that define requirements for the component. See Section 3.1.2 for more information on  
 1332 the SCAP Content Validation Tool, which can help validate the correctness of SCAP result data streams.

#### 1333 4.4.2 The Target Identification

1334 The target asset MUST be represented in the ARF report using the `<ai:assets>` part of ARF. The  
 1335 `<ai:asset>` element populated about a target asset SHOULD include the fields specified in Table 20,  
 1336 where applicable.

1337 **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

1338  
 1339 Additional identification information MAY be captured in the `<ai:asset>` element (asset tag, system  
 1340 GUID, etc.) The guidelines specified in [AI] MUST be followed when populating the asset identification  
 1341 information.

1342 Currently, only the target asset of the SCAP evaluation is identified.

#### 1343 4.4.3 The Source Data Stream

1344 The source data stream collection that was used to generate the results against the target SHOULD be  
 1345 included in the ARF report as an `<arf:report-request>`. If the source data stream collection is  
 1346 included in the ARF report and an `<xccdf:Tailoring>` component was used during processing, the  
 1347 tailoring component SHALL be included as well. This is a stripped down example:

1348

```

1349 <arf:asset-report-collection>
1350   <arf:report-requests>
1351     <arf:report-request id="request_0">
1352       <arf:content>
1353         <ds:data-stream-collection id="..."
1354           <!-- Source data stream collection which was tailored -->
1355           ...
1356         </ds:data-stream-collection>
1357       </arf:content>
1358     </arf:report-request>
1359
1360     <arf:report-request id="request_1">
1361       <arf:content>
1362         <ds:data-stream-collection id="..."
1363           <!-- Source Data Stream Collection with an <xccdf:Tailoring>
1364 component -->
1365           ...
1366         </ds:data-stream-collection>
1367       </arf:content>
1368     </arf:report-request>
1369   </arf:report-requests>
1370   <arf:assets>...</arf:assets>
1371   <arf:reports>...</arf:reports>
1372 </arf:asset-report-collection>

```

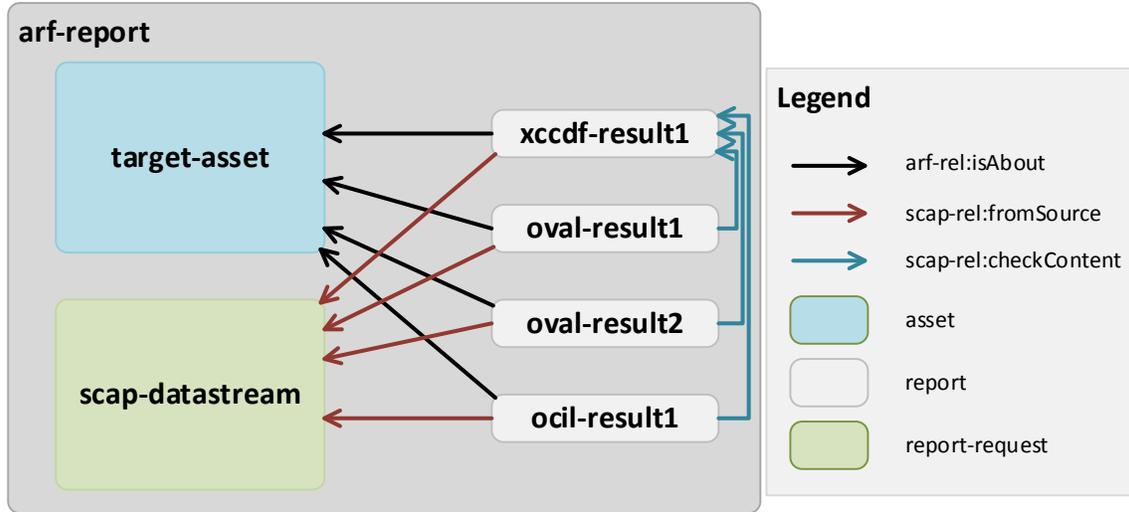
1373 **4.4.4 The Relationships**

1374 Table 21 outlines the relationships that MUST be specified in the ARF report if the stated condition is  
 1375 satisfied.

1376 **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

1377  
 1378 Figure 3 gives an example of how the resulting ARF report would look.



1379  
1380

Figure 3: Sample ARF Report Structure

1381 **4.5 XCCDF Results**

1382 The following requirements and recommendations pertain to content consumers generating XCCDF result  
1383 data stream components.

1384 Each XCCDF result data stream component SHALL comply with the XCCDF Results schema.

1385 XCCDF test results SHALL be documented as the contents of an `<xccdf:TestResult>` element. To  
1386 be considered valid SCAP result content, the `<xccdf:TestResult>` element SHALL meet the  
1387 following conditions:

- 1388 1. The `@start-time` and `@end-time` attributes SHALL be provided to indicate when the scan  
1389 started and completed, respectively.
- 1390 2. The `@test-system` attribute SHALL be provided, and it SHALL be a CPE name value  
1391 indicating the product that was responsible for generating the results.
- 1392 3. When the `<xccdf:TestResult>` is the root XCCDF element, then it will include an  
1393 `<xccdf:benchmark>` element [XCCDF:6.6.2].
  - 1394 a. The `<xccdf:benchmark>` element MUST have an `@id` attribute specified. The `@id`  
1395 attribute SHALL match the value of the `<xccdf:Benchmark>` element's `@id`  
1396 attribute that was processed.
  - 1397 b. The `<xccdf:benchmark>` element MUST have an `@href` attribute specified. The  
1398 `@href` attribute SHALL hold the URI to the XCCDF component (either local to the data  
1399 stream collection or remote) that was processed. The URI MUST be in the form specified  
1400 for the `@href` attribute in Table 8.
- 1401 4. If a child profile of an `<xccdf:Tailoring>` element was applied during processing, then the  
1402 `<xccdf:tailoring-file>` element SHALL be present and SHALL provide the following  
1403 information about the `<xccdf:Tailoring>` element: `@href`, `@id`, `@version`, and  
1404 `@time`. The `@href` attribute SHALL hold the URI to the XCCDF Tailoring component and  
1405 SHALL comply with the format described above (item 3).
- 1406 5. The `<xccdf:Profile>` element SHALL be included if a profile was applied during  
1407 processing. This is also applicable to selected profiles part of `<xccdf:Tailoring>`.

- 1408 6. Regarding the definition and use of `<xccdf:Profile>` elements, reported `<xccdf:set-`  
 1409 `value>` elements SHALL include all those values that are exported by the reported rules. The  
 1410 specific settings are those determined by the reported `<xccdf:Profile>`.
- 1411 7. The `<xccdf:identity>` element SHALL identify the security principal used to access rule  
 1412 evaluation on the target(s). This will include the identity name or username used to perform the  
 1413 evaluation.
- 1414 8. Each IP address(es) associated with the `<xccdf:target>` SHALL be enumerated using the  
 1415 `<xccdf:target-address>` element.
- 1416 9. An `<xccdf:target-id-ref>` SHALL be specified with a `@system` attribute of  
 1417 `"http://scap.nist.gov/schema/asset-identification/1.1"`, an `@href` attribute value of `"`, and a  
 1418 `@name` attribute value of the ID of the `<ai:asset>` element in the ARF that this  
 1419 `<xccdf:TestResult>` is about.
- 1420 10. The `<xccdf:rule-result>` elements report the result of the application of each selected  
 1421 rule [XCCDF:6.6.2]. The `<xccdf:check/xccdf:check-content-ref>` element  
 1422 SHALL record the reference to the check system specific result component report ID and check  
 1423 name within the result file using the `@href` and `@name` attributes, respectively. The `@href`  
 1424 attribute SHALL contain `"#"` + the `@id` of the `<arf:report>` containing the check result. This  
 1425 approach provides traceability between XCCDF and check results. Note that if `@multi-check`  
 1426 is not set to `"true"` and the `<xccdf:rule-result>` represents a group of checks, then the  
 1427 `@name` attribute SHALL be omitted. See the example below the next requirement.
- 1428 11. Where applicable to the target system, each of the `<xccdf:fact>` elements in Table 22  
 1429 SHALL be provided. Previous versions of SCAP required additional facts; these have been  
 1430 incorporated into the use of the Asset Identification specification, as discussed in Section 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

1433  
 1434  
 1435 Here is a stripped down example illustrating the above requirements:

```

1436 <arf:asset-report-collection>
1437   <rc:relationships>...</rc:relationships>
1438   <arf:report-requests>...</arf:report-requests>
1439   <arf:assets>...</arf:assets>
1440   <arf:reports>
1441     <arf:report id="scap_gov.nist_comp_r3005-xccdf_01">
```

```

1443         <arf:content>
1444             <xccdf:TestResult start-time="2016-03-10T10:07:11" version="1-2.1.0.0"
1445 test-system="cpe:/a:vendor:product_name:version"
1446             end-time="2016-03-10T10:07:11"
1447             id="xccdf_gov.nist_testresult_...">
1448                 <xccdf:benchmark href="file:r3005-datastream-
1449 01.xml#scap_gov.nist_comp_r3005-xccdf_01" id="xccdf_gov.nist_benchmark_r3005_id_01"/>
1450
1451                 <xccdf:tailoring-file href="#scap_gov.nist_comp_r3005-
1452 xccdf_tailoring_03" id="xccdf_gov.nist_tailoring_r3005_03" time="2016-01-22T14:00:00"
1453 version="1-2.1.0.0"/>
1454
1455                 <xccdf:organization>...</xccdf:organization>
1456                 <xccdf:identity privileged="true"
1457 authenticated="true">...</xccdf:identity>
1458                 <xccdf:profile
1459 idref="xccdf_gov.nist.validation_profile_r3005_tailoring_03"/>
1460                 <xccdf:target>...</xccdf:target>
1461                 <xccdf:target-address>...</xccdf:target-address>
1462                 <xccdf:target-facts>...</xccdf:target-facts>
1463                 <xccdf:target-id-ref system="http://scap.nist.gov/schema/asset-
1464 identification/1.1" href="" name="...">
1465                 <xccdf:set-value
1466 idref="xccdf_gov.nist_value_validation.r3005_for_rule_6">test0</xccdf:set-value>
1467                 ...
1468                 <xccdf:rule-result time="2016-03-10T10:07:11"
1469 idref="xccdf_gov.nist_rule_validation.r3005_rule_1" weight="10" severity="medium">
1470                 <xccdf:result>pass</xccdf:result>
1471                 <xccdf:check system="http://oval.mitre.org/XMLSchema/oval-
1472 definitions-5" selector="sell">
1473                 <xccdf:check-content-ref href="#scap_gov.nist_comp_r3005-
1474 oval" name="oval:nist.validation.r3005:def:2"/>
1475                 </xccdf:check>
1476                 </xccdf:rule-result>
1477             </xccdf:TestResult>
1478
1479         </arf:content>
1480     </arf:report>
1481     <arf:report id="scap_gov.nist_comp_r3005-oval">
1482         <arf:content>
1483             <arf:xccdf-res:oval-results>...</arf:xccdf-res:oval-results>
1484         </arf:content>
1485     </arf:report>
1486     ...
1487 </arf:reports>
1488 </arf:asset-report-collection>
1489

```

#### 1490 4.5.1 Assigning Identifiers to Rule Results

1491 The `<xccdf:rule-result>` element provides data indicating the result of assessing a system using  
1492 the identified `<xccdf:Rule>` element. If the target `<xccdf:Rule>` identified by the  
1493 `<xccdf:rule-result>` element's `@idref` attribute has one or more `<xccdf:ident>` elements  
1494 with a `@system` attribute value listed in Section 3.2.4.1, then each `<xccdf:ident>` element SHALL  
1495 also appear within the `<xccdf:rule-result>` element.

1496 Here is an example for a CVE entry:

```

1497 <xccdf:rule-result idref="java-upgrade-278" weight="10.0">
1498     <xccdf:result>pass</xccdf:result>
1499     ...

```

```

1500     <xccdf:ident system="http://cve.mitre.org">CVE-2006-0614</xccdf:ident>
1501     ...
1502 </xccdf:rule-result>

```

1503  
 1504 If the `<xccdf:ident>` element is included, for tracking purposes it is important that produced XCCDF  
 1505 results have specific meanings. If an `<xccdf:ident>` element is present and it identifies a CVE, CCE,  
 1506 or CPE entry, then an `<xccdf:rule-result>` of “pass” SHALL indicate that the check content  
 1507 evaluated within the rule complied with one of the following:

- 1508 • For a CVE entry, the target platform satisfies all the conditions of the XCCDF rule and is  
 1509 unaffected by the vulnerability or exposure referenced by the CVE.
- 1510 • For a CCE entry, the target platform complies with the configuration setting guidance expressed  
 1511 in the XCCDF rule.
- 1512 • For a CPE entry, the target platform was identified on the system.

1513 It is important that these interpretations of `<xccdf:ident>` elements be preserved. For example,  
 1514 consider two policy recommendations. One is that a particular piece of software be installed, and the  
 1515 second that another piece of software not be installed. Both rules for these policy recommendations could  
 1516 use the same CPE entry in their `<xccdf:ident>` elements. However, because the interpretation of a  
 1517 CPE entry is that a “pass” result indicates software was installed, the second policy recommendation’s  
 1518 rule would violate this. This can be corrected by using the `@con:negate` attribute, a Boolean attribute  
 1519 that inverts the rule result. The second rule could check for the software being installed and then negate  
 1520 that result, thus giving a result consistent in meaning with the first rule. For rules that cannot have their  
 1521 interpretations preserved through the use of the `@con:negate` attribute, an alternative is to have a CCE  
 1522 entry corresponding to the recommendation. Rules that do not use `<xccdf:ident>` elements have no  
 1523 such restrictions.

#### 1524 4.5.2 Mapping OVAL Results to XCCDF Results

1525 When evaluating an `<xccdf:Rule>` element that references an OVAL Definition, the  
 1526 `<xccdf:rule-result>` element SHALL be used to capture the result of this evaluation. This result  
 1527 SHALL be determined by evaluating the referenced OVAL Definition on a target host. The result value  
 1528 of an individual `<xccdf:check>` SHALL be mapped from the OVAL Definition result produced  
 1529 during evaluation. The corresponding `<xccdf:rule-result/xccdf:result>` value is then  
 1530 computed based on the result values of all relevant `<xccdf:check>` elements. (Normally only a single  
 1531 `<xccdf:check>` element is needed, but where an `<xccdf:complex-check>` element is used,  
 1532 there may be multiple results that must be combined, as outlined in the XCCDF specification.) While the  
 1533 OVAL specification permits limiting result status reporting, SCAP-conformant content SHALL include  
 1534 full status reporting, including Error, Unknown, Not Applicable, Not Evaluated, True, and False.

1535 Content consumers SHALL apply the mapping illustrated in Table 23 when deriving `<xccdf:check>`  
 1536 results from OVAL Definition processing. The corresponding result value SHALL be recorded based on  
 1537 the `@class` attribute of the OVAL Definition and the `@negate` attribute of the `<xccdf:check>`  
 1538 element where applicable.

1539

**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		
Definition Class	Definition Result	fail	pass
compliance	False		
vulnerability	true		
inventory	false		
patch	true		

1540

1541 The mappings in Table 23 are specific to each OVAL Definition class. For example, if an OVAL  
 1542 compliance class definition is processed and OVAL returns a result of "true", the content consumer is  
 1543 conveying the fact that the system was found to be compliant with that check and therefore returns a  
 1544 "pass" result for that check. A similar definition for a vulnerable condition will return results of "false" if  
 1545 that vulnerability was not found on the examined devices, resulting in a "pass" from the XCCDF check.  
 1546 Negations of check results or their combination in complex-checks may result in additional modification  
 1547 before the final corresponding `<xccdf:rule-result/xccdf:result>` value is known.

1548 If the `<xccdf:Rule>` element under evaluation has an `<xccdf:check-content-ref>` element  
 1549 with the `@name` attribute omitted and an `<xccdf:check>` element with its `@multi-check` attribute  
 1550 set to "true", then the result of each evaluated OVAL Definition SHALL be recorded as a separate  
 1551 `<xccdf:rule-result>` element. In this case the `<xccdf:rule-result/xccdf:check-`  
 1552 `content-ref>` element SHALL identify the specific check result of each evaluated OVAL Definition  
 1553 using the `@href` and `@name` attributes as described in Section 4.5, item 8.

1554 According to [XCCDF:Table 9;Table 35;Table 39], if the `<xccdf:Rule>` element under evaluation is  
 1555 selected and its `@role` attribute is set to "unchecked", then the rule result SHALL be set to  
 1556 "notchecked". If the `<xccdf:Rule>` element under evaluation is selected and its `@role` attribute is set  
 1557 to "unscored", then the rule result SHALL be set to "informational".

1558 **4.6 OVAL Results**

1559 The following requirements and recommendations pertain to content consumers generating OVAL result  
 1560 data stream components. See the annex for additional requirements and recommendations.

1561 Each OVAL result data stream component SHALL validate against at least one version of the OVAL  
 1562 Results schema that corresponds to an OVAL component specification version specified in Section 2 of  
 1563 the annex, regardless of the version of the OVAL Definitions document that was evaluated.

1564 An SCAP OVAL result data stream component SHALL include the results of every OVAL Definition  
 1565 used to generate the reported results.

1566 In order to be SCAP conformant, an SCAP content consumer SHALL be able to produce all the types of  
 1567 OVAL Results output described below. The specific result output SHALL be configurable within the  
 1568 SCAP content consumer.

1569 In order to support SCAP instances where OVAL thin content (only the ID of the definition and the  
 1570 results) is preferred, SCAP content consumers SHALL support all valid values for the `<oval-  
 1571 res:directives>` controlling the expected content of the results file.

1572 To support the ability for results to be consumed by the appropriate product(s), data results SHALL be  
 1573 expressed as Single Machine Without System Characteristics, Single Machine With System  
 1574 Characteristics, or Single Machine With Thin Results as follows:

- 1575 1. Single Machine Without System Characteristics – A single result file that includes the results of  
 1576 all OVAL Definitions evaluated and “full” results types as described in the `<oval-  
 1577 res:ContentEnumeration>` element, without system characteristics.

1578 For this format, the values for the `<oval-res:directives>` element SHALL be:

```
1579 <oval-res:directives include_source_definitions="false">
1580   <oval-res:definition_true content="full" reported="true"/>
1581   <oval-res:definition_false content="full" reported="true"/>
1582   <oval-res:definition_unknown content="full" reported="true"/>
1583   <oval-res:definition_error content="full" reported="true"/>
1584   <oval-res:definition_not_evaluated content="full" reported="true"/>
1585   <oval-res:definition_not_applicable content="full" reported="true"/>
1586 </oval-res:directives>
```

1587  
 1588 When creating the OVAL System Characteristics as defined by the `<oval-  
 1589 sc:oval_system_characteristics>` element, the `<oval-sc:collected_objects>` and  
 1590 `<oval-sc:system_data>` elements SHALL NOT be provided.

- 1591 2. Single Machine With System Characteristics – A single result file that includes the results of all  
 1592 OVAL Definitions evaluated and “full” results types as described in the `<oval-  
 1593 res:ContentEnumeration>` element and the System Characteristics of the target evaluated.

1594 For this format, the values for the `<oval-res:directives>` element SHALL be:

```
1595 <oval-res:directives include_source_definitions="false">
1596   <oval-res:definition_true content="full" reported="true"/>
1597   <oval-res:definition_false content="full" reported="true"/>
1598   <oval-res:definition_unknown content="full" reported="true"/>
1599   <oval-res:definition_error content="full" reported="true"/>
1600   <oval-res:definition_not_evaluated content="full" reported="true"/>
1601   <oval-res:definition_not_applicable content="full" reported="true"/>
1602 </oval-res:directives>
```

1603  
 1604  
 1605 When creating the OVAL System Characteristics as defined by the `<oval-  
 1606 sc:oval_system_characteristics>` element, the `<oval-sc:collected_objects>` and  
 1607 `<oval-sc:system_data>` elements SHALL be provided.

- 1608 3. Single Machine With Thin Results – A single result file that includes the results of all OVAL  
 1609 Definitions evaluated and “thin” results types as described in the OVAL Results schema. A value  
 1610 of “thin” means only the minimal amount of information will be provided.

1611 For this format, the values for the `<oval-res:directives>` element SHALL be:

```
1612 <oval-res:directives include_source_definitions="false">
1613   <oval-res:definition_true content="thin" reported="true"/>
1614   <oval-res:definition_false content="thin" reported="true"/>
```

```

1615     <oval-res:definition_unknown content="thin" reported="true"/>
1616     <oval-res:definition_error content="thin" reported="true"/>
1617     <oval-res:definition_not_evaluated content="thin" reported="true"/>
1618     <oval-res:definition_not_applicable content="thin" reported="true"/>
1619 </oval-res:directives>
1620

```

1621 When specifying OVAL system characteristics, a reference SHOULD be made to the target asset in the  
 1622 ARF report collection. Specifically, the `<oval-sc:oval_system_characteristics>/<oval-`  
 1623 `sc:system_info>/##any` SHOULD be populated with a `<con:asset-identification>`  
 1624 element. That element MUST be populated with a single `<arf:object-ref>` element that points to  
 1625 the `<ai:asset>` element in the ARF report collection pertaining to the OVAL result. See [ARF] for  
 1626 details on populating the `<arf:object-ref>` element.

#### 1627 4.7 OCIL Results

1628 The following requirements and recommendations pertain to content consumers generating OCIL result  
 1629 data stream components.

1630 An SCAP OCIL result data stream component SHALL include the results of every  
 1631 `<ocil:questionnaire>`, `<ocil:question_test_action>`, and `<ocil:question>`  
 1632 element used to generate the reported results.

#### 1633 4.8 Result Data Stream Signing

1634 Digitally signing result data stream content is important to ensuring the integrity and trustworthiness of  
 1635 results. Leveraging [TMSAD] for SCAP can improve the legitimacy of results of SCAP content and  
 1636 create a more secure environment. As such, content consumers MAY digitally sign result content  
 1637 following the guidelines in [TMSAD], along with the following requirements.

1638 One XML digital signature MAY be included in an `<arf:extended-info>` element in the ARF  
 1639 report. The signature MUST be represented as a `<dsig:Signature>` element and MUST follow the  
 1640 W3C recommendation [DSIG]. The `<dsig:Signature>` element MUST sign the ARF report  
 1641 collection root element.

1642 The `<dsig:Signature>` element MUST follow the recommendations in [TMSAD] and these  
 1643 additional requirements:

- 1644 1. A `<dsig:SignatureProperties>` element MUST be included in the  
 1645 `<dsig:Signature>` element. At least one `<dsig:SignatureProperty>` element  
 1646 MUST be populated with `<dt:signature-info>` as specified in [TMSAD].
- 1647 2. The first `<dsig:Reference>` element in a `<dsig:Signature>` element MUST be to the  
 1648 `<arf:asset-report-collection>` element. The element MUST be referenced in the  
 1649 `@URI` attribute using the empty string convention “”.
- 1650 3. Two XPath Filter 2 transforms MUST exist on the first `<dsig:Reference>` element in a  
 1651 `<dsig:Signature>` element. Both MUST specify a filter type of “subtract”. The first  
 1652 transform MUST specify the XPath “/arf:asset-report-collection/arf:extended-  
 1653 `infos[count(arf:extended-info[dsig:Signature]) = count(*)]”. The second transform MUST specify`  
 1654 `the XPath “/arf:asset-report-collection/arf:extended-infos/arf:extended-info[dsig:Signature]”. In`  
 1655 `both cases, the namespace prefix “arf” MUST map to the ARF namespace specified in this`  
 1656 `document.`

1657 4. The second *<dsig:Reference>* element MUST be to the  
1658 *<dsig:SignatureProperties>* element captured in a *<dsig:Object>* element with  
1659 the *<dsig:Signature>* element. The *<dsig:SignatureProperties>* element MUST  
1660 be referenced in the *@URI* attribute using “#” + *@Id* of the  
1661 *<dsig:SignatureProperties>* element.

1662 5. Key information SHOULD be provided on the *<dsig:Signature>* element.

1663 In situations where it is desirable to countersign a result data stream (e.g., when a content consumer  
1664 automatically signs a result data stream and then a person also wants to sign the results), the following  
1665 requirements apply.

1666 1. The *<arf:extended-info>* element containing the original signature SHALL be removed  
1667 from the resulting document.

1668 2. The original signature SHALL be captured as a *<dsig:Object>* element on the new  
1669 *<dsig:Signature>* element.

1670 3. The first *<dsig:Reference>* element on the new *<dsig:Signature>* element SHALL  
1671 reference the *<dsig:Object>* element containing the original signature. The  
1672 *<dsig:Object>* element MUST be referenced in the *@URI* attribute using “#” + *@Id* of the  
1673 *<dsig:Object>* element.

1674 4. The second *<dsig:Reference>* element MUST be to the  
1675 *<dsig:SignatureProperties>* element captured in a *<dsig:Object>* element with  
1676 the *<dsig:Signature>* element. The *<dsig:SignatureProperties>* element MUST  
1677 be referenced in the *@URI* attribute using “#” + *@Id* of the  
1678 *<dsig:SignatureProperties>* element.

1679 5. A *<dsig:SignatureProperties>* element MUST be included in the  
1680 *<dsig:Signature>* element. At least one *<dsig:SignatureProperty>* element  
1681 MUST be populated with *<dt:signature-info>* as specified in [TMSAD].

1682 6. Key information SHOULD be provided on the *<dsig:Signature>* element in accordance  
1683 with [TMSAD].

1684 7. The new *<dsig:Signature>* element MUST be placed in a new *<arf:extended-info>*  
1685 element in the ARF report collection.

1686 A signature that has countersigned another signature (also known as an enveloping signature) MAY be  
1687 countersigned. When doing so, the requirements above SHALL apply to the new signature creation.

1688 When signing a result data stream, the source data stream collection SHOULD be captured in the ARF  
1689 report being signed.

## 1690 5. Source Data Stream Content Requirements for Use Cases

1691 This section discusses additional requirements for the following SCAP-conformant content use cases:  
1692 compliance checking, vulnerability scanning, and inventory scanning. Note that as stated in Table 3 in  
1693 Section 3.1, each data stream is required to have a *@use-case* attribute in its *<ds:data-stream>*  
1694 element with a value corresponding either to one of the content types defined in this section or to  
1695 “OTHER”, for data streams not corresponding to a defined use case. The required value for each content  
1696 type is specified below in the appropriate subsection.

1697 Each use case is subject not only to the requirements presented in this section, but also to all applicable  
1698 requirements in Sections 3 and 4.

### 1699 5.1 Compliance Checking

1700 SCAP content can be used to compare system characteristics and settings against an SCAP-conformant  
1701 checklist in an automated fashion. This can verify that operating systems and applications comply with  
1702 security checklists and identify any deviations from those checklists.

1703 The SCAP source data stream component that **MUST** be included for compliance checking is the XCCDF  
1704 benchmark, which expresses the checklist. Each rule in the XCCDF benchmark **SHALL** reference one of  
1705 the following:

- 1706 • An OVAL compliance definition. This definition **SHALL** be contained in an OVAL component,  
1707 which holds definitions of compliance checks used by the checklist. An XCCDF benchmark’s rules  
1708 **MAY** reference one or more OVAL compliance class definitions in an OVAL component.
- 1709 • An OCIL questionnaire. This questionnaire **SHALL** be contained in an OCIL component, which  
1710 holds questionnaires that collect information that OVAL is not being used to collect, such as posing  
1711 questions to users or harvesting configuration information from an existing database. An XCCDF  
1712 benchmark’s rules **MAY** reference one or more OCIL questionnaires in an OCIL component.
- 1713 • An OVAL patch definition. This definition **SHALL** be contained in an OVAL component, which  
1714 holds definitions for patch compliance checks. These checks may be needed if an organization  
1715 includes patch verification in its compliance activities. An XCCDF benchmark **MAY** reference an  
1716 OVAL patch definition through a patches up-to-date rule in a manner consistent with Section 3.2.4.3.

1717 Each XCCDF benchmark **SHALL** have at least one rule that references either an OVAL compliance class  
1718 definition in an OVAL component or an OCIL questionnaire in an OCIL component.

1719 All OVAL components and OCIL components referenced by the XCCDF benchmark **SHALL** be  
1720 included in the SCAP source data stream.

1721 If the XCCDF benchmark component references any CPE names, then the SCAP source data stream  
1722 **MUST** include a CPE component, which specifies the products or platforms of interest, and **MUST**  
1723 include one or more OVAL inventory class definitions in an OVAL component that contain the technical  
1724 procedures for determining whether or not a specific target asset has a product or platform of interest.

1725 The *@use-case* attribute in the *<ds:data-stream>* element **MUST** be set to  
1726 “CONFIGURATION”.

### 1727 5.2 Vulnerability Scanning

1728 SCAP content can be used to scan operating systems and applications to look for known software flaws  
1729 that introduce security exposures. The content enables consistent detection and reporting of these flaws.

1730 The SCAP source data stream component that MUST be included for vulnerability scanning is the  
1731 XCCDF benchmark, which expresses the checklist of the flaws to be checked for. Each rule in the  
1732 XCCDF benchmark SHALL reference one of the following:

- 1733 • An OVAL vulnerability definition. This definition SHALL be contained in an OVAL component,  
1734 which holds definitions of vulnerability checks used by the checklist. An XCCDF benchmark's rules  
1735 MAY reference one or more OVAL vulnerability class definitions in an OVAL component.
- 1736 • An OCIL questionnaire. This questionnaire SHALL be contained in an OCIL component, which  
1737 holds questionnaires that collect information that OVAL is not being used to collect, such as giving a  
1738 system administrator step-by-step directions for manually examining a system for a vulnerability that  
1739 cannot be detected with OVAL, and then collecting information on the results of that manual  
1740 examination. An XCCDF benchmark's rules MAY reference one or more OCIL questionnaires in an  
1741 OCIL component.
- 1742 • An OVAL patch definition. This definition SHALL be contained in an OVAL component, which  
1743 holds definitions for patch compliance checks. These checks may be needed if an organization  
1744 includes patch verification in its vulnerability scanning activities. An XCCDF benchmark MAY  
1745 reference an OVAL patch definition through a patches up-to-date rule in a manner consistent with  
1746 Section 3.2.4.3.

1747 Each XCCDF benchmark SHALL have at least one rule that references either an OVAL vulnerability  
1748 class definition in an OVAL component or an OCIL questionnaire in an OCIL component.

1749 All OVAL components and OCIL components referenced by the XCCDF benchmark SHALL be  
1750 included in the SCAP source data stream.

1751 If the XCCDF benchmark component references any CPE names, then the SCAP source data stream  
1752 MUST include a CPE component, which specifies the products or platforms of interest, and MUST  
1753 include one or more OVAL inventory class definitions in an OVAL component that contain the technical  
1754 procedures for determining whether or not a specific target asset has a product or platform of interest.

1755 The *@use-case* attribute in the *<ds:data-stream>* element MUST be set to  
1756 "VULNERABILITY".

### 1757 5.3 Inventory Scanning

1758 SCAP content can be used to collect information on the software installed on systems. One example of  
1759 how this could be used is to verify that a group of systems all have required security software programs  
1760 installed. This could help verify compliance with technical security control requirements. Another  
1761 example is to collect software inventory data on devices that are not directly connected to the enterprise  
1762 network, such as smart phones.

1763 Inventory scanning can also be applied to collect information on the presence of software artifacts on  
1764 systems, such as malware or characteristics of malware that indicate its presence. SCAP content authored  
1765 for this purpose can be used to detect classes or categories of malware based on system state that may be  
1766 common across multiple malware instances. For example, it is a common practice to reuse malware code,  
1767 making modifications to address available detection methods, change propagation characteristics, etc. It is  
1768 also possible to author content that detects a specific instantiation of malware. For example, hashing of  
1769 files can be used to identify a malicious executable or library.

1770 The SCAP source data stream component that MUST be included for inventory scanning is the XCCDF  
1771 benchmark, which references the inventory checks and captures the results. Each rule in the XCCDF  
1772 benchmark SHALL reference one of the following:

- 1773 • An OVAL inventory definition. This definition SHALL be contained in an OVAL component, which  
1774 holds definitions of technical procedures for determining whether or not a specific target asset has  
1775 software (product, platform, malware, etc.) of interest. An XCCDF benchmark's rules MAY  
1776 reference one or more OVAL inventory class definitions in an OVAL component.
- 1777 • An OCIL questionnaire. This questionnaire SHALL be contained in an OCIL component, which  
1778 holds questionnaires that collect information that OVAL is not being used to collect, such as posing  
1779 questions to users or harvesting inventory information from an existing database. An XCCDF  
1780 benchmark's rules MAY reference one or more OCIL questionnaires in an OCIL component.
- 1781 The *@use-case* attribute in the *<ds:data-stream>* element MUST be set to "INVENTORY".  
1782

**Appendix A—Security Considerations**

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

- 1785 • **Confidentiality.** SCAP does not define any mechanisms for protecting the confidentiality of SCAP  
1786 content or results. Organizations can add on such protections as they deem appropriate, such as  
1787 encrypting results files that contain sensitive information regarding system vulnerabilities.
- 1788 • **Malicious content.** While SCAP does provide mechanisms for ensuring integrity of SCAP content  
1789 and verifying content signatures, SCAP does not have any features specifically for handling malicious  
1790 SCAP content (benchmarks, tailoring files, etc.) At a minimum, organizations should generate  
1791 signatures for their content and verify signatures on all content before using it to ensure that the  
1792 content has not been maliciously altered. Also, organizations should not process content that fails  
1793 validation, and for stronger assurance may choose not to use any content that has not been signed.
- 1794 • **Security value of content.** It is outside the scope of SCAP’s capabilities to make any assertions or  
1795 assessments regarding the security value of SCAP checklists and other forms of SCAP content.  
1796 People and organizations may determine security value through their own methods, such as applying  
1797 checklists to test systems and evaluating the results of those tests, but SCAP itself does not have any  
1798 way of ensuring the security value of its content.
- 1799 • **Component security.** Be aware of security considerations of all of the component protocols,  
1800 specifications, standards, etc. used by SCAP. SCAP does not impose any additional security  
1801 requirements on these.

1802

1803 **Appendix B—Acronyms and Abbreviations**

1804 Selected acronyms and abbreviations used in the guide are defined below.

<b>API</b>	Application Programming Interface
<b>ARF</b>	Asset Reporting Format
<b>CCE</b>	Common Configuration Enumeration
<b>CCSS</b>	Common Configuration Scoring System
<b>CPE</b>	Common Platform Enumeration
<b>CVE</b>	Common Vulnerabilities and Exposures
<b>CVSS</b>	Common Vulnerability Scoring System
<b>DHS</b>	Department of Homeland Security
<b>DoD</b>	Department of Defense
<b>FISMA</b>	Federal Information Security Management Act
<b>IR</b>	Interagency Report
<b>IT</b>	Information Technology
<b>ITL</b>	Information Technology Laboratory
<b>NIST</b>	National Institute of Standards and Technology
<b>NVD</b>	National Vulnerability Database
<b>OCIL</b>	Open Checklist Interactive Language
<b>OMB</b>	Office of Management and Budget
<b>OS</b>	Operating System
<b>OVAL</b>	Open Vulnerability and Assessment Language
<b>PCI</b>	Payment Card Industry
<b>RFC</b>	Request for Comments
<b>SCAP</b>	Security Content Automation Protocol
<b>SP</b>	Service Pack
<b>SP</b>	Special Publication
<b>SWID</b>	Software Identification
<b>TMSAD</b>	Trust Model for Security Automation Data
<b>URI</b>	Uniform Resource Identifier
<b>URL</b>	Uniform Resource Locator
<b>XCCDF</b>	Extensible Configuration Checklist Description Format
<b>XML</b>	Extensible Markup Language

1805

1806 **Appendix C—Glossary**

1807 This appendix contains definitions for selected terms used within the document.

<b>Component schema</b>	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.
<b>Component specification</b>	One of the individual specifications that comprises SCAP.
<b>Content consumer</b>	A product that accepts existing SCAP source data stream content, processes it, and produces SCAP result data streams
<b>Content producer</b>	A product that generates SCAP source data stream content.
<b>Globally unique identifier</b>	An identifier formatted following special conventions to support uniqueness within an organization and across all organizations creating identifiers. See Section 3.1.3 for the conventions.
<b>Result content</b>	Part or all of one or more SCAP result data streams.
<b>Security Content Automation Protocol (SCAP)</b>	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.
<b>SCAP component</b>	A logical unit of data expressed using one or more of the SCAP component specifications.
<b>SCAP conformant</b>	A product or SCAP data stream that meets the requirements of this specification.
<b>SCAP content</b>	Part or all of one or more SCAP data streams.
<b>SCAP data stream</b>	A specific instantiation of SCAP content.
<b>SCAP data stream collection</b>	A container for SCAP data streams and components.
<b>SCAP result data stream</b>	An SCAP data stream that holds output (result) content.
<b>SCAP source data stream</b>	An SCAP data stream that holds input (source) content.
<b>SCAP source data stream collection</b>	A container for SCAP data streams and components.
<b>SCAP use case</b>	A pre-defined way in which a product can use SCAP. See Section 5 for the definitions of the SCAP use cases.
<b>Source content</b>	Part or all of SCAP source data streams.
<b>Stream component</b>	A major element of a data stream, such as an XCCDF benchmark or a set of OVAL definitions.
<b>Well-formed</b>	An SCAP-conformant data stream or stream component.

1808

1809

1810 **Appendix D—Normative References**

1811 This appendix provides normative references to the specifications that are required to implement the  
 1812 SCAP 1.3 components. See the annex for normative references to the schema and schematron locations  
 1813 related to these specifications.

1814 Table 24 lists the normative references to specifications. Please see the annex for additional normative  
 1815 references to specifications cited in the annex.

1816

1817

**Table 24: Specification Locations**

Abbreviation	Name	URL
[AI]	Asset Identification	<a href="http://csrc.nist.gov/publications/nistir/ir7693/NISTIR-7693.pdf">http://csrc.nist.gov/publications/nistir/ir7693/NISTIR-7693.pdf</a>
[ARF]	ARF	<a href="http://csrc.nist.gov/publications/nistir/ir7694/NISTIR-7694.pdf">http://csrc.nist.gov/publications/nistir/ir7694/NISTIR-7694.pdf</a>
[CCE]	CCE	N/A
[CCSS]	CCSS	<a href="http://csrc.nist.gov/publications/nistir/ir7502/nistir-7502_CCSS.pdf">http://csrc.nist.gov/publications/nistir/ir7502/nistir-7502_CCSS.pdf</a>
[CPE]	CPE	See [CPE-D], [CPE-L], [CPE-M], and [CPE-N]
[CPE-D]	CPE Dictionary	<a href="http://csrc.nist.gov/publications/nistir/ir7697/NISTIR-7697-CPE-Dictionary.pdf">http://csrc.nist.gov/publications/nistir/ir7697/NISTIR-7697-CPE-Dictionary.pdf</a>
[CPE-L]	CPE Applicability Language	<a href="http://csrc.nist.gov/publications/nistir/ir7698/NISTIR-7698-CPE-Language.pdf">http://csrc.nist.gov/publications/nistir/ir7698/NISTIR-7698-CPE-Language.pdf</a>
[CPE-M]	CPE Name Matching	<a href="http://csrc.nist.gov/publications/nistir/ir7696/NISTIR-7696-CPE-Matching.pdf">http://csrc.nist.gov/publications/nistir/ir7696/NISTIR-7696-CPE-Matching.pdf</a>
[CPE-N]	CPE Naming	<a href="http://csrc.nist.gov/publications/nistir/ir7695/NISTIR-7695-CPE-Naming.pdf">http://csrc.nist.gov/publications/nistir/ir7695/NISTIR-7695-CPE-Naming.pdf</a>
[CVE]	CVE	N/A
[CVSS]	CVSS	<a href="https://www.first.org/cvss/cvss-v30-specification-v1.7.pdf">https://www.first.org/cvss/cvss-v30-specification-v1.7.pdf</a>
[DCES]	Dublin Core metadata version 1.1	<a href="http://dublincore.org/documents/2012/06/14/dces/">http://dublincore.org/documents/2012/06/14/dces/</a>
[DSIG]	DSIG	<a href="http://www.w3.org/TR/xmlsig-core/">http://www.w3.org/TR/xmlsig-core/</a>
[OCIL]	OCIL	<a href="http://csrc.nist.gov/publications/nistir/ir7692/nistir-7692.pdf">http://csrc.nist.gov/publications/nistir/ir7692/nistir-7692.pdf</a>
[RFC2119]	RFC 2119	<a href="http://www.ietf.org/rfc/rfc2119.txt">http://www.ietf.org/rfc/rfc2119.txt</a>
[RFC3986]	RFC 3986	<a href="http://www.ietf.org/rfc/rfc3986.txt">http://www.ietf.org/rfc/rfc3986.txt</a>
[SWID]	ISO/IEC 19770-2:2015	<a href="http://www.iso.org/iso/catalogue_detail.htm?csnumber=65666">http://www.iso.org/iso/catalogue_detail.htm?csnumber=65666</a>
[SWID-CYBER]	NISTIR 8060	<a href="http://dx.doi.org/10.6028/NIST.IR.8060">http://dx.doi.org/10.6028/NIST.IR.8060</a>
[TMSAD]	TMSAD	<a href="http://csrc.nist.gov/publications/nistir/ir7802/NISTIR-7802.pdf">http://csrc.nist.gov/publications/nistir/ir7802/NISTIR-7802.pdf</a>
[XCCDF]	XCCDF	<a href="http://csrc.nist.gov/publications/PubsNISTIRs.html#NIST-IR-7275-r4">http://csrc.nist.gov/publications/PubsNISTIRs.html#NIST-IR-7275-r4</a>
[XINCLUDE]	XInclude specification	<a href="https://www.w3.org/TR/2006/REC-xinclude-20061115/">https://www.w3.org/TR/2006/REC-xinclude-20061115/</a>
[XLINK]	XLink specification	<a href="https://www.w3.org/TR/2001/REC-xlink-20010627/">https://www.w3.org/TR/2001/REC-xlink-20010627/</a>
[XMLCAT]	XML Catalog specification	<a href="https://www.oasis-open.org/committees/download.php/14809/xml-catalogs.html">https://www.oasis-open.org/committees/download.php/14809/xml-catalogs.html</a>
[XMLS]	W3C XML Schema	<a href="https://www.w3.org/TR/2004/REC-xmlschema-1-20041028/">https://www.w3.org/TR/2004/REC-xmlschema-1-20041028/</a> , <a href="https://www.w3.org/TR/2004/REC-xmlschema-2-20041028/">https://www.w3.org/TR/2004/REC-xmlschema-2-20041028/</a>

1818

1819 **Appendix E—Change Log**1820 **Revision 3 Release 0**

- 1821 • Revised and reformatted front matter for the document.
- 1822 • Added an errata table to the document for future use.
- 1823 • Made minor editorial and formatting changes throughout document.
- 1824 • Updated SCAP version from 1.2 to 1.3. Changes included the following:
  - 1825 ○ Added the SWID specification to SCAP and created a new Section 3.6 with related
  - 1826 requirements.
  - 1827 ○ Updated the CVSS specification version from 2.0 to 3.0.
  - 1828 ○ Revised the property definition for the `<ds:data-stream>` element's `@scap-version`
  - 1829 property in Table 3 to include "1.3" as a possible value.
  - 1830 ○ Revised Section 3.2.2, item 4 so that the `<xccdf:Benchmark>` element's `@style`
  - 1831 attribute should have the value "SCAP\_1.3" instead of "SCAP\_1.2".
- 1832 • Created the annex document (NIST SP 800-126A) and moved all OVAL version information
- 1833 there. Additional changes related to OVAL include the following:
  - 1834 ○ Eliminated the "least version principle" approach.
  - 1835 ○ Added a requirement to Section 3.3 of this document regarding the consistency of the
  - 1836 `<oval:schema_version>` element and the `<xsi:schemaLocator>` value for the
  - 1837 OVAL schema.
  - 1838 ○ Changed the Section 4.6 (OVAL Results) requirement related to which version of the OVAL
  - 1839 Results schema each OVAL result data stream must validate against.
- 1840 • Section 2 (SCAP 1.3 Conformance):
  - 1841 ○ Changed the "Enumeration" component specification category to "Identification Scheme."
- 1842 • Section 3.1 (SCAP Source Data Stream):
  - 1843 ○ Revised Figure 2 and the example below it to illustrate the current conventions for SCAP data
  - 1844 stream component references.
  - 1845 ○ Expanded the property definition in Table 8 for the `<ds:component-ref>` element's
  - 1846 `@href` property to more clearly define the required URI form for referencing external
  - 1847 content
  - 1848 ○ Rewrote the property definition in Table 9 for the `<cat:catalog>` element's
  - 1849 `@rewriteURI` property
  - 1850 ○ Moved the details of the `<cat:uri>` element and the `<cat:rewriteURI>` element from
  - 1851 Table 9 (`<cat:catalog>`) to Table 10 and Table 11, respectively
- 1852 • Section 3.2 (XCCDF):
  - 1853 ○ Modified wording in Section 3.2.4.2 on the `<xccdf:check>` element to clarify how the
  - 1854 use of check systems other than OVAL and OCIL affects SCAP content
  - 1855 ○ Revised Section 3.2.4.3 to distinguish checking for patches using a rule that references
  - 1856 numerous patch class definitions versus a single OVAL definition
- 1857 • Section 4.1 (Legacy Support):
  - 1858 ○ Revised requirements related to legacy SCAP content and OVAL content.
- 1859 • Section 4.4 (SCAP Result Data Streams):
  - 1860 ○ Expanded Section 4.4.3 on the source data stream to provide an additional requirement for
  - 1861 tailoring component usage and to include an example.
- 1862 • Section 4.5 (XCCDF Results):
  - 1863 ○ Expanded requirement 3 on `<xccdf:benchmark>` element attributes.
  - 1864 ○ Created a new requirement 4 on the `<xccdf:tailoring-file>` element.
  - 1865 ○ Created a new requirement 5 on the `<xccdf:Profile>` element.

- 1866 ○ Removed the example from requirement 10 (formerly requirement 8) and added a new
- 1867 example after requirement 11.
- 1868 ○ Revised the second paragraph of Section 4.5.2 and Table 23 on mapping OVAL results to
- 1869 XCCDF results to take into account the *@negate* attribute of the `<xccdf:check>`
- 1870 element. Also added a new paragraph at the end of Section 4.5.2 to clarify the application of
- 1871 the *@role* attribute's value to the rule result.
- 1872 ● Appendix D (Normative References):
- 1873 ○ Changed URLs to point directly to specification locations.
- 1874 ○ Moved the contents of the original Table 22 on schema and Schematron file locations to
- 1875 NIST SP 800-126A.

1876

### 1877 **Revision 2 Release 1 – 28 September 2011**

- 1878 ● Final version released.
- 1879 ● Made editorial changes throughout document, including extensive addition of cross references.
- 1880 ● Section 3.1 (SCAP Source Data Stream):
- 1881 ○ Improved explanations of source data streams; added XML example and updated diagrams.
- 1882 ○ Added *@schematron-version* attribute to `<ds:data-stream-collection>`.
- 1883 ○ Added `<ds:Tailoring>` element to `<ds:component>` (was previously being treated as
- 1884 an element of `<ds:extended-component>`).
- 1885 ○ Expanded the discussion of Schematron files.
- 1886 ○ Added conventions for globally unique identifiers for `<scap:data-stream-`
- 1887 `collection>`, `<scap:data-stream>`, `<scap:component-ref>`,
- 1888 `<scap:component>`, and `<scap:extended-component>`.
- 1889 ● Section 3.2 (XCCDF):
- 1890 ○ Prohibited use of XInclude elements in XCCDF content, use of the `<xccdf:set-`
- 1891 `complex-value>` element within the `<xccdf:Profile>` element, and use of XCCDF
- 1892 group extension.
- 1893 ○ Clarified use of `<xccdf:ident>` elements and added the *@con:negate* attribute.
- 1894 ○ Clarified use of `<xccdf:check-content-ref>` elements.
- 1895 ● Section 4.1 (Legacy Support):
- 1896 ○ Added explicit information and requirements regarding deprecated constructs in SCAP
- 1897 component specifications.
- 1898 ● Section 4.2 (Source Data Streams):
- 1899 ○ Added a Schematron requirement.
- 1900 ○ Clarified what warnings tools must issue for an unrecognized `<ds:extended-`
- 1901 `component>`.
- 1902 ● Section 4.3 (XCCDF Processing):
- 1903 ○ Clarified the CPE applicability processing requirements.
- 1904 ○ Clarified requirements regarding the use of check systems not supported by SCAP.
- 1905 ● Section 4.4 (SCAP Result Data Streams):
- 1906 ○ Added an ARF example.
- 1907 ○ Added an *scap-ref:associatedWith* relationship requirement for ARF reports.
- 1908 ● Section 4.5 (XCCDF Results):
- 1909 ○ Deleted several facts from the XCCDF Fact Descriptions table.
- 1910 ○ Deleted redundant requirements (present in the latest XCCDF specification).
- 1911 ○ Clarified processing of `<xccdf:ident>` elements and added the *@con:negate* attribute.
- 1912 ○ Removed the requirements for the FDCC XCCDF results format.
- 1913 ● Section 5 (Source Data Stream Content Requirements for Use Cases):
- 1914 ○ Removed the OVAL-only use case.

- 1915 • Appendices:
- 1916 ○ Added a new Appendix A containing security considerations for this version of SCAP.
- 1917 ○ Added a new Appendix C containing a glossary with key terms.
- 1918 ○ Added a list of SCAP schema and Schematron file locations to Appendix D.

1919

**1920 Revision 2 Release 0 – 12 July 2011**

- 1921 • Complete draft specification for version 1.2 released for public comment.
- 1922 • Made editorial changes throughout the document.
- 1923 • Added the following component specifications to SCAP: ARF 1.1, Asset Identification 1.1,
- 1924 CCSS 1.0, and TMSAD 1.0. Updated the following component specifications from SCAP 1.1:
- 1925 XCCDF from 1.1.4 to 1.2; OVAL from 5.8 to 5.10; and CPE from 2.2 to 2.3. Added and revised
- 1926 requirements throughout the specification to use these component specification versions.
- 1927 • In Section 2, rewrote the conformance requirements and defined “content producer” and “content
- 1928 consumer” terms.
- 1929 • Section 3:
- 1930 ○ Added an SCAP source data stream subsection and a subsection on digitally signing source
- 1931 data stream content.
- 1932 ○ Added identifier use requirements for `<xccdf:Rule>` and `<xccdf:ident>` elements.
- 1933 ○ Added requirements for the `<xccdf:Value>` element.
- 1934 ○ Added requirements related to Schematron rules.
- 1935 • Section 4:
- 1936 ○ Revised legacy support requirements for SCAP content and OVAL definition documents.
- 1937 ○ Added an SCAP result data stream subsection. Added source and result data stream
- 1938 requirements throughout the section. Also added a subsection on digitally signing result data
- 1939 stream content.
- 1940 ○ Added a declaration of the FDCC Reporting Format.
- 1941 • In Section 5, added malware detection material to the Inventory Scanning use case.
- 1942 • Updated the normative references.
- 1943 • Added Appendix C (change log).