National Institute of
Standards and Technology

U.S. Department of Commerce

NIST Interagency Report 7275
Revision 4

Specification for the Extensible Configuration Checklist Description Format (XCCDF) Version 1.2

David Waltermire
Charles Schmidt
Karen Scarfone
Neal Ziring

NIST Interagency Report 7275
Revision 4

Specification for the Extensible Configuration Checklist Description Format (XCCDF) Version 1.2

David Waltermire
Charles Schmidt
Karen Scarfone
Neal Ziring

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

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

March 2012

U.S. Department of Commerce

John E. Bryson, Secretary

National Institute of Standards and Technology

Patrick D. Gallagher, Under Secretary for Standards and Technology and Director

Reports on Computer Systems Technology

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

National Institute of Standards and Technology Interagency Report 7275 Revision 4
80 pages (Mar. 2012)

Acknowledgments

The authors of this report, David Waltermire of the National Institute of Standards and Technology (NIST), Charles Schmidt of The MITRE Corporation, Karen Scarfone of Scarfone Cybersecurity, and Neal Ziring of the National Security Agency (NSA), wish to thank all contributors to this revision of the publication, particularly Adam Halbardier of Booz Allen Hamilton, Vladimir Giszpenc, Kent Landfield and Richard Whitehurst of McAfee, Lisa Nordman of The MITRE Corporation, Joe Wolfkiel of DISA, and Shane Shaffer and Matt Kerr of G2, Inc.

The authors would also like to acknowledge the following individuals who contributed to the initial definition and development of the Extensible Configuration Checklist Description Format (XCCDF): David Proulx, Mike Michnikov, Andrew Buttner, Todd Wittbold, Adam Compton, George Jones, Chris Calabrese, John Banghart, Murugiah Souppaya, John Wack, Trent Pitsenbarger, and Robert Stafford. Stephen D. Quinn, Peter Mell, and Matthew Wojcik contributed to Revisions 1, 2, and 3 of this report. Ryan Wilson of Georgia Institute of Technology also made substantial contributions. Thanks also go to the Defense Information Systems Agency (DISA) Field Security Office (FSO) Vulnerability Management System (VMS)/Gold Disk team for extensive review and many suggestions.

Abstract

This report specifies the data model and Extensible Markup Language (XML) representation for the Extensible Configuration Checklist Description Format (XCCDF) Version 1.2. An XCCDF document is a structured collection of security configuration rules for some set of target systems. The XCCDF specification is designed to support information interchange, document generation, organizational and situational tailoring, automated compliance testing, and scoring. The specification also defines a data model and format for storing results of security guidance or checklist testing. The intent of XCCDF is to provide a uniform foundation for expression of security checklists and other configuration guidance, and thereby foster more widespread application of good security practices.

Audience

The primary audience of the XCCDF specification is government and industry security analysts, and security management product developers.

Trademark Information

Contents

1. INTRODUCTION 1

1.1 PURPOSE AND SCOPE 1
1.2 DOCUMENT STRUCTURE 1
1.3 DOCUMENT CONVENTIONS 1

2. NORMATIVE REFERENCES 2

3. TERMS, DEFINITIONS, AND ABBREVIATIONS 3

3.1 XCCDF TERMINOLOGY 3
3.2 ACRONYMS AND ABBREVIATIONS 3

4. CONFORMANCE 4

4.1 PRODUCT CONFORMANCE 4
4.2 BENCHMARK DOCUMENT CONFORMANCE 4

5. XCCDF OVERVIEW 5

5.1 INTRODUCTION 5
5.2 CHECKLIST STRUCTURE AND TAILORING 6
5.3 TEST RESULTS 7

6. XCCDF DATA MODEL 8

6.1 INTRODUCTION 8
6.2 GENERAL XML INFORMATION 9
6.2.1 XCCDF Namespace and XML Schema 9
6.2.2 Element and Attribute Formatting 9
6.2.3 Element Identifiers 10
6.2.4 <xccdf:metadata> Element 10
6.2.5 Platform Names 11
6.2.6 <xccdf:reference> Element 12
6.2.7 <xccdf:signature> Element 13
6.2.8 Status Tracking 13
6.2.9 Text Substitution 13
6.2.10 @xml:lang Attribute 14
6.3 <XCCDF:BENCHMARK> 15
6.3.1 Basics 15
6.3.2 Properties 16
6.4 ITEM ELEMENTS 18
6.4.1 Properties 18
6.4.2 <xccdf:warning> Element 20
6.4.3 <xccdf:Group> Element 21
6.4.4 <xccdf:Rule> Element 21
6.4.5 <xccdf:Value> Element 30
6.5 <XCCDF:PROFILE> ELEMENT 34
6.5.1 Basics 34
6.5.2 Properties 34
6.5.3 Selectors 35
6.6 <XCCDF:TESTRESULT> ELEMENT 37
6.6.1 Basics 37
6.6.2 Properties 38

6.6.3 <xccdf:fact> Element 41
6.6.4 <xccdf:rule-result> Element 41
6.6.5 <xccdf:tailoring-file> Element 44
6.7 <XCCDF:TAILORING> ELEMENT 45
6.7.1 Basics 45
6.7.2 Properties 45
6.7.3 Profile Shadowing 46
6.7.4 Tailoring Actions and Profile Selectors 47

7. XCCDF PROCESSING 48

7.1 INTRODUCTION 48
7.2 LOADING AND TRAVERSAL 48
7.2.1 Introduction 48
7.2.2 Loading 48
7.2.3 Traversal 51
7.3 ASSESSMENT OUTPUTS 63
7.3.1 Overview 63
7.3.2 Scoring Models 63

APPENDIX A— CONVERTING XCCDF 1.1.4 CONTENT TO XCCDF 1.2 66

A.1 CHANGES TO THE XCCDF XML NAMESPACE 66
A.2 CONVERSION OF IDENTIFIERS 66
A.3 CONVERSION OF <XCCDF:SUB> ELEMENTS 66
A.4 PROPERTIES REMOVED OR DEPRECATED SINCE XCCDF 1.1.4 67

APPENDIX B— CHANGE LOG 68

Tables

TABLE 1: CONVENTIONAL XML MAPPINGS 1
TABLE 2: RECOMMENDED CLASS VALUES 9
TABLE 3: ELEMENT IDENTIFIER FORMAT CONVENTIONS 10
TABLE 4: <XCCDF:BENCHMARK> ELEMENT PROPERTIES 16
TABLE 5: ITEM ELEMENT PROPERTIES 18
TABLE 6: PROPERTIES SPECIFIC TO <XCCDF:GROUP> AND <XCCDF:RULE> ELEMENTS 20
TABLE 7: <XCCDF:WARNING> ELEMENT @CATEGORY ATTRIBUTE VALUES 21
TABLE 8: <XCCDF:GROUP> ELEMENT PROPERTIES 21
TABLE 9: <XCCDF:RULE> ELEMENT PROPERTIES 23
TABLE 10: ASSIGNED VALUES FOR THE @SYSTEM ATTRIBUTE OF AN <XCCDF:IDENT> ELEMENT 24
TABLE 11: <XCCDF:CHECK> ELEMENT PROPERTIES 25
TABLE 12: TRUTH TABLE FOR AND 27
TABLE 13: TRUTH TABLE FOR OR 27
TABLE 14: TRUTH TABLE FOR NEGATION 27
TABLE 15: POSSIBLE PROPERTIES FOR <XCCDF:FIXTEXT> ELEMENT 28
TABLE 16: POSSIBLE PROPERTIES FOR <XCCDF:FIX> ELEMENT 29

TABLE 17: PREDEFINED VALUES FOR @SYSTEM ATTRIBUTE OF <XCCDF:FIX> ELEMENT 30
TABLE 18: <XCCDF:VALUE> ELEMENT PROPERTIES 31
TABLE 19: POSSIBLE PROPERTIES FOR <XCCDF:CHOICES> ELEMENT 32
TABLE 20: PERMITTED OPERATORS BY VALUE TYPE 33
TABLE 21: <XCCDF:PROFILE> ELEMENT PROPERTIES 35
TABLE 22: SELECTORS 36
TABLE 23: <XCCDF:TESTRESULT> ELEMENT PROPERTIES 39
TABLE 24: PREDEFINED @NAME ATTRIBUTE VALUES FOR <XCCDF:FACT> ELEMENTS 41
TABLE 25: <XCCDF:RULE-RESULT> ELEMENT PROPERTIES 42
TABLE 26: POSSIBLE RESULTS FOR A SINGLE TEST 43
TABLE 27: <XCCDF:OVERRIDE> ELEMENT PROPERTIES 44
TABLE 28: <XCCDF:TAILORING-FILE> ELEMENT PROPERTIES 45
TABLE 29: <XCCDF:TAILORING> ELEMENT PROPERTIES 46
TABLE 30: PROFILE SHADOWING BEHAVIOR 47
TABLE 31: TAILORING ACTIONS AND PROFILE SELECTORS 47
TABLE 32: LOADING PROCESSING SEQUENCE SUB-STEPS 48
TABLE 33: INHERITANCE PROCESSING MODEL 50
TABLE 34: BENCHMARK PROCESSING ALGORITHM SUB-STEPS 51
TABLE 35: ITEM PROCESSING ALGORITHM SUB-STEPS 52
TABLE 36: PROFILE SELECTOR EXAMPLE: INITIAL CONFIGURATION 56
TABLE 37: PROFILE SELECTOR EXAMPLE: INITIAL BENCHMARK STATE 56
TABLE 38: PROFILE SELECTOR EXAMPLE: FINAL BENCHMARK STATE 58
TABLE 39: CHECK PROCESSING ALGORITHM SUB-STEPS 59
TABLE 40: DEFAULT MODEL ALGORITHM SUB-STEPS 64
TABLE 41: FLAT MODEL ALGORITHM SUB-STEPS 65
TABLE 42: ALTERNATIVE OPERATIONS FOR REMOVED AND DEPRECATED XCCDF 1.1.4 CONSTRUCTS 67
TABLE 43: MAPPING PREVIOUS RELEASE SECTIONS TO THIS RELEASE 69

Figures

FIGURE 1: TYPICAL STRUCTURE OF A BENCHMARK 15
FIGURE 2: CHECK PROCESSING FLOWCHART (WHEN THE CHECK'S PARENT IS AN <XCCDF:RULE>) 60
FIGURE 3: WORKFLOW FOR ASSESSING BENCHMARK COMPLIANCE 63

1. Introduction

1.1 Purpose and Scope

This report defines the specification for the Extensible Configuration Checklist Description Format (XCCDF) version 1.2. The report also defines and explains the requirements that XCCDF 1.2 documents and products (i.e., software) must meet to claim conformance with the specification. This report only applies to XCCDF version 1.2. All other versions are outside the scope of this report.

1.2 Document Structure

The remainder of this report is composed of the following sections and appendices:

- Section 2 provides a list of normative references for the report.
- Section 3 defines selected terms and abbreviations used in the report.
- Section 4 provides the high-level requirements for claiming conformance with the XCCDF version 1.2 specification.
- Section 5 gives an overview of XCCDF and its capabilities.
- Section 6 provides an introduction to the XCCDF data model and details additional requirements and recommendations for XCCDF's use.
- Section 7 discusses XCCDF processing requirements and recommendations.
- Appendix A explains how to convert XCCDF 1.1.4-specific properties to their XCCDF 1.2 counterparts.
- Appendix B provides a change log that documents significant changes to released drafts of this specification. This includes a section-by-section mapping of how the document was reorganized from the previous drafts to this draft. Readers who are familiar with any previous XCCDF versions may find it helpful to review Appendix B first before the rest of the document.

1.3 Document Conventions

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

Namespace prefixes used in this specification are listed in Table 1.

Table 1: Conventional XML Mappings

Prefix	Namespace	Schema
cpe2	http://cpe.mitre.org/language/2.0	Common Platform Enumeration (CPE) 2.3 Applicability Language
cpe2-dict	http://cpe.mitre.org/dictionary/2.0	CPE 2.3 Dictionary
dc	http://purl.org/dc/elements/1.1/	Simple Dublin Core elements
dsig	http://www.w3.org/2000/09/xmldsig#	Interoperable XML digital signatures
xccdf	http://checklists.nist.gov/xccdf/1.2	XCCDF policy documents
xml	http://www.w3.org/XML/1998/namespace	Common XML attributes
xsd	http://www.w3.org/2001/XMLSchema	XML Schema
xsi	http://www.w3.org/2001/XMLSchema-Instance	XML Schema Instance

2. Normative References

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

[DCES], DCMI (Dublin Core Metadata Initiative), *Dublin Core Metadata Element Set, Version 1.1*, October 2010, available at <http://dublincore.org/documents/dces/>

[DCXML], DCMI, *Guidelines for Implementing Dublin Core in XML*, April 2003, available at <http://dublincore.org/documents/dc-xml-guidelines/>

[ILSR], IANA, *IANA Language Subtag Registry (ILSR)*, available at <http://www.iana.org/assignments/language-subtag-registry>

[IR7693], NIST, NIST IR 7693, *Specification for Asset Identification 1.1*, June 2011, available at <http://csrc.nist.gov/publications/PubsNISTIRs.html>

[IR7695], NIST, NIST IR 7695, *Common Platform Enumeration: Naming Specification Version 2.3*, August 2011, available at <http://csrc.nist.gov/publications/PubsNISTIRs.html>

[IR7698], NIST, NIST IR 7698, *Common Platform Enumeration: Applicability Language Specification Version 2.3*, August 2011, available at <http://csrc.nist.gov/publications/PubsNISTIRs.html>

[PCRE], Perl Compatible Regular Expressions (PCRE), available at <http://www.pcre.org>

[RFC2119], IETF, RFC 2119, *Key words for use in RFCs to Indicate Requirement Levels*, March 1997, available at <http://www.ietf.org/rfc/rfc2119.txt>

[RFC5646], IETF, RFC 5646, *Tags for Identifying Languages*, September 2009, available at <http://www.ietf.org/rfc/rfc5646.txt>

[UNICODE], Unicode Technical Recommendation No. 18, *Unicode Regular Expressions*, version 9, January 2004, available at <http://unicode.org/reports/tr18/>

[XHTML], W3C (World Wide Web Consortium), *XHTML Basic*, December 2000, available at <http://www.w3.org/TR/2000/REC-xhtml-basic-20001219/>

[XINCLUDE], W3C, *XML Inclusions (XInclude) Version 1.0 (Second Edition)*, November 2006, available at <http://www.w3.org/TR/xinclude/>

[XMLDSIG], W3C, *XML Signature Syntax and Processing (Second Edition)*, June 2008, available at <http://www.w3.org/TR/xmldsig-core/>

[XMLNAME], W3C, *Namespaces in XML 1.0 (Third Edition)*, December 2009, available at <http://www.w3.org/TR/REC-xml-names/>

[XMLSCHEMA], W3C, *XML Schema Part 2: Datatypes Second Edition*, October 2004, available at <http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/>

[XPATH], W3C, *XML Path Language (XPath) Version 1.0*, November 1999, available at <http://www.w3.org/TR/xpath/>

3. Terms, Definitions, and Abbreviations

For the purposes of this document, the following terms, definitions, and abbreviations apply.

3.1 XCCDF Terminology

Benchmark: The root node of an XCCDF benchmark document; may also be the root node of an XCCDF results document (the results of evaluating the XCCDF benchmark document).

Benchmark Consumer: A product that accepts an existing XCCDF benchmark document, processes it, and produces an XCCDF results document.

Benchmark Producer: A product that generates XCCDF benchmark documents.

Checklist: An organized collection of rules about a particular kind of system or platform.

Group: An item that can hold other items; allows an author to collect related items into a common structure and provide descriptive text and references about them.

Item: A named constituent of a benchmark. The three types of items are groups, rules, and values.

Profile: A named tailoring of a benchmark.

Rule: An element that holds check references and may also hold remediation information.

Tailoring: An element that specifies profiles to modify the behavior of a benchmark; the top-level element of a tailoring document.

TestResult: The container for XCCDF results. May be the root node of an XCCDF results document.

Value: A named data value that can be substituted into other items' properties or into checks.

3.2 Acronyms and Abbreviations

CCE	Common Configuration Enumeration
CPE	Common Platform Enumeration
CVE	Common Vulnerabilities and Exposures
DCMI	Dublin Core Metadata Initiative
DNS	Domain Name System
IANA	Internet Assigned Numbers Authority
IR	Interagency Report
NIST	National Institute of Standards and Technology
OCIL	Open Checklist Interactive Language
OVAL	Open Vulnerability and Assessment Language
PCRE	Perl Compatible Regular Expression
RFC	Request for Comments
SCAP	Security Content Automation Protocol
SP	Special Publication
W3C	World Wide Web Consortium
XCCDF	Extensible Configuration Checklist Description Format
XHTML	Extensible Hypertext Markup Language
XML	Extensible Markup Language

4. Conformance

Products and organizations may want to claim conformance with this specification for a variety of reasons. For example, a software vendor may want to assert that its product generates and/or processes XCCDF benchmark documents properly. Another example is a policy mandating that an organization use XCCDF for documenting and executing its security configuration checklists.

This section provides the high-level requirements that a product or benchmark document MUST meet for conformance with this specification. Most of the requirements listed in this section reference other sections in the report that fully define the requirements.

Other specifications that use XCCDF MAY define additional requirements and recommendations for XCCDF's use. Such requirements and recommendations are outside the scope of this publication.

4.1 Product Conformance

There are two types of XCCDF products: benchmark producers and benchmark consumers. *Benchmark producers* are products that generate XCCDF benchmark documents, while *benchmark consumers* are products that accept an existing XCCDF benchmark document, process it, and produce an XCCDF results document. Products claiming conformance with this specification SHALL adhere to the following requirements:

1. For benchmark producers, generate well-formed XCCDF benchmark documents. This includes following the benchmark document requirements specified in Section 4.2 and all of the pertinent processes defined in Sections 6 and 7.

2. For benchmark consumers, consume and process well-formed XCCDF benchmark documents, and generate well-formed XCCDF results documents. This includes following all of the pertinent processes defined in Sections 6 and 7.

3. Make an explicit claim of conformance to this specification in any documentation provided to end users.

4.2 Benchmark Document Conformance

XCCDF benchmark documents claiming conformance with this specification SHALL follow these requirements:

1. Adhere to the official XCCDF schema as explained in Section 6.

2. Adhere to the syntax, structural, and other XCCDF benchmark document requirements defined in Sections 6 and 7.

5. XCCDF Overview

5.1 Introduction

XCCDF was created to document technical and non-technical security checklists using a standardized format. The general objective is to allow security analysts and IT experts to create effective, interoperable automated checklists, and to support the use of these checklists with a wide variety of tools. A checklist is an organized collection of rules about a particular kind of system or platform. Automation is necessary for consistent and rapid verification of system security because of the sheer number of things to check and the number of hosts within an organization that need to be assessed (often many thousands).

XCCDF enables easier, more uniform creation of security checklists, which in turn helps to improve system security by more consistent and accurate application of sound security practices. Adoption of XCCDF lets security professionals, security tool vendors, and system auditors exchange information more quickly and precisely, and also permits greater automation of security testing and configuration assessment. Additional capabilities provided by XCCDF include the following:

- Ensure compliance to multiple policies (systems subject to the Federal Information Security Management Act [FISMA], Security Technical Implementation Guide [STIG], Health Insurance Portability and Accountability Act [HIPAA], etc.)
- Permit faster, more cooperative, and more automated definition of security rules, procedures, guidance documents, alerts, advisories, and remediation measures
- Permit fast, uniform, manageable administration of security checks and audits
- Permit composition of security rules and tests from different community groups and vendors
- Facilitate scoring, reporting, and tracking of security status and checklist conformance for systems

The XCCDF specification, which is vendor-neutral, is suited for a wide variety of checklist applications. XCCDF has an open, standardized format, amenable to generation by and editing with a variety of tools. In addition, because it is expressed using XML, an XCCDF document is embeddable inside other documents. XCCDF also includes provisions for incorporating other data formats, and it is extensible to include new functionality, features, and data stores without hindering the functionality of existing XCCDF tools.

Since XCCDF's creation, various commercial, government, and community developers have created tools that support XCCDF, allowing a single XCCDF checklist to be used by many organizations and many tools. These tools read an XCCDF checklist and follow it to perform the necessary checks and ask the necessary questions to measure conformance with the checklist and generate corresponding reports.

A common use case for an XCCDF checklist is normalizing security configuration content through automated tools. Such tools accept one or more XCCDF checklists along with supporting system test definitions, and determine whether the specified rules are satisfied by a target system. The XCCDF checklist supports generation of a report, including a weighted score. XCCDF checklists can also be used to test whether or not a system is vulnerable to a particular kind of attack. For this purpose, the XCCDF checklist plays the role of a vulnerability alert, but with the ability to describe the problem, drive automated verification of its presence, and convey recommendations for corrective actions.

The scenarios below illustrate some uses of XCCDF security checklists and tools.

- Scenario 1 – An industry consortium, in conjunction with a product vendor, wants to produce a security checklist for an application server. The core security settings are the same for all OS platforms on which the server runs, but a few settings are OS-specific. The consortium crafts one checklist for the core settings and writes several OS-specific ones that supplement the core settings. Users download the core checklist and the OS-specific checklists that apply to their installations, and then run an assessment tool to score their compliance with the checklists.
- Scenario 2 – An academic group produces a checklist for secure configuration of a particular server operating system version. A government agency issues a set of rules extending the academic checklist to meet more stringent user authorization criteria imposed by statute. A medical enterprise downloads both the academic checklist and the government extension, tailors them to fit their internal security policy, and uses them for an enterprise-wide audit using a commercial security audit tool. Reports outputted by the tool include remediation measures which the IT staff can use to bring their systems into full internal policy compliance. (Note that remediation processes should be carefully planned and implemented.)

These scenarios demonstrate some of XCCDF's range of capabilities. XCCDF can represent complex conditions and relationships about the systems to be assessed, and it can incorporate descriptive material and remediative measures. It is also designed to be modular; for example, XCCDF benchmarks acquire programmatically ascertainable information through lower-level check system languages.

5.2 Checklist Structure and Tailoring

The basic unit of structure for a checklist is a rule. A rule simply describes a state or condition which the target of the document should exhibit. A simple checklist might consist of a list of rules, but richer ones require additional structure. XCCDF allows checklist authors to impose organization within the checklist, such as putting related rules into named groups and designating the order for processing rules and groups.

Checklist users can employ tailoring tools to customize a checklist's rules for their local environment or policies. For example, an auditor might need to set the password policy requirement to be more stringent than the default recommendation. Another example is that an organization may have trouble applying particular settings because of legacy systems or conflicts with other software. In cases such as these, the checklist users may need to tailor the checklist. The following customization options are available:

- **Selectability** – A tailoring action selects or deselects a rule or group of rules. For example, an entire group of rules that relate to physical security might not apply to a network scan, so that group could be deselected. In the case of NIST Special Publication (SP) 800-53, certain rules apply according to the impact rating of the system. For example, systems that have an impact rating of *low* might not have all of the same access control requirements as a system with a *high* impact rating, so the rules that are not applicable for the *low* system can be deselected.
- **Value Modification** – A tailoring action substitutes a locally-significant value for a general value in an XCCDF variable (`<xccdf:Value>`). This locally-significant value then gets used wherever the variable is referenced. For example, at a site where all logs are sent to a single host, the address of that log server could be substituted into an audit configuration variable. Using the NIST SP 800-53 example, a system with a *moderate* impact rating might require a 12-character password, whereas a system with a *low* impact rating might only require an 8-character password.
- **Property Modification** – A tailoring action modifies a property for an element not addressed by selectability or value modification. For example, an author could alter the relative weight of particular rules or groups of rules.

XCCDF 1.2 supports the creation and use of tailoring documents, which define tailoring profiles available for use with a particular benchmark document. Having a tailoring document allows sets of checklist customizations to be recorded in a consistent manner.

XCCDF allows checklists to include descriptive and interrogative text to help checklist users make tailoring decisions, even directing users through the process. Some combinations of rules within the same checklist might conflict or be mutually exclusive. To avert problems, the checklist author can identify particular tailoring choices as incompatible. Checklist authors can also designate the modes (e.g., Gold, Platinum, High Impact Rating, Level 1) under which a rule should be processed. Checklist authors should include the appropriate text in their checklists to aid in tailoring.

Another important facet of customization is extension. XCCDF supports mechanisms for authors to extend (inherit from) existing rules, values, and groups, in addition to expressing rules, values, and groups in their entirety. For example, in XCCDF checklists, it is desirable to share descriptive material among several rules, and to allow a specialized rule to be created by extending a base rule. (Note that group extension has been deprecated in XCCDF 1.2. Use of group extension is strongly discouraged because it has known problems, such as those described in Section 7.2.2, and content that uses it is unlikely to be interoperable between XCCDF products.)

5.3 Test Results

Many organizations use several security products to determine the security of IT systems and their compliance to various policies. Unfortunately, if the outputs from these products are not standardized, costly customization and integration can be required for trending, aggregation, and reporting. Addressing this, XCCDF provides a standardized reporting format for storing the results of the rule checking subsystem. Security tools sometimes include only the results of the test or tests in the form of a pass/fail status. Other tools provide additional information (e.g., instead of simply indicating that more than one privileged account exists on a system, certain tools also provide the list of privileged accounts). The following information is basic to all XCCDF results:

- The security guidance document or checklist used, along with any adaptations via customization or tailoring applied
- Information about the target system to which the test was applied, including arbitrary identification and configuration information about the target system
- The time interval of the test, and the time instant at which each individual rule was evaluated
- One or more scores
- References to lower-level details possibly stored in other output files.

6. XCCDF Data Model

6.1 Introduction

The XCCDF data model and its XML representation are intended to be platform-independent and portable to foster broad adoption and sharing of rules. The fundamental data model for XCCDF consists of the following main element data types:

- **Benchmark.** An XCCDF benchmark document holds an *<xccdf:Benchmark>* element, which acts as a container for other elements, including *<xccdf:Group>*, *<xccdf:Rule>*, *<xccdf:Value>*, *<xccdf:Profile>*, and *<xccdf:TestResult>* elements.
- **Item.** An item is a named constituent of an *<xccdf:Benchmark>*. There are three types of items:
  - **Group.** An *<xccdf:Group>* holds other items. An *<xccdf:Group>* collects related *<xccdf:Rule>* and *<xccdf:Value>* elements into a common structure and can provide descriptive text and references about them. An *<xccdf:Group>* allows benchmark users to select and deselect related *<xccdf:Rule>* elements together; since a deselected *<xccdf:Group>* is not processed, none of its contained items are processed either. Selection of an *<xccdf:Group>* allows its children to be processed normally based on their individual selection states.
  - **Rule.** An *<xccdf:Rule>* element holds check references and can also hold remediation information.
  - **Value.** An *<xccdf:Value>* element is a named data value that can be substituted into other items' properties or into checks. It can have an associated data type and metadata that express how the value should be used and how it can be tailored.
- **Profile.** An *<xccdf:Profile>* element is a named tailoring of a benchmark using a collection of attributed references to *<xccdf:Rule>*, *<xccdf:Group>*, and *<xccdf:Value>* elements. It allows definition of named levels or baselines in a benchmark (see Section 5.2).
- **TestResult.** An *<xccdf:TestResult>* element holds the results of performing a test or check against a single target device or system. An *<xccdf:TestResult>* element references *<xccdf:Rule>* and *<xccdf:Value>* elements and may also reference an *<xccdf:Profile>* element.
- **Tailoring.** A tailoring document holds exactly one *<xccdf:Tailoring>* element, which contains *<xccdf:Profile>* elements to modify the behavior of an *<xccdf:Benchmark>*.

The rest of this section explains the relationships between these main element data types and examines each of the types in more detail. Section 6.2 discusses general information that applies to most or all of the main element data types. Sections 6.3 through 6.7 cover *<xccdf:Benchmark>*, item (*<xccdf:Group>*, *<xccdf:Rule>*, *<xccdf:Value>*), *<xccdf:Profile>*, *<xccdf:TestResult>*, and *<xccdf:Tailoring>* elements, respectively.

6.2 General XML Information

6.2.1 XCCDF Namespace and XML Schema

The namespace URI for this specification SHALL be "http://checklists.nist.gov/xccdf/1.2". Applications that process XCCDF SHOULD use the namespace URI to decide whether or not they can process a given document. The XML representation of XCCDF is expressed as an XML schema at http://scap.nist.gov/specifications/xccdf/#resource-1.2. The XML schema implementation of XCCDF SHALL be the authoritative XML binding definition for XCCDF.

6.2.2 Element and Attribute Formatting

The structure of an XCCDF document supports transformation into HTML and various XML formats to promote portability and interoperability. The XCCDF language also allows for the inclusion of content that does not contribute directly to the technical content, such as an introduction, a rationale, warnings, and external references. XCCDF also provides mechanisms to document authors for formatting text, including images, and referencing other information resources (e.g., prose publications), but these mechanisms are separable from the text itself so they can be filtered out by applications that do not support or require them.

Throughout the rest of this section, there are tables that show the possible properties of each main element in the XCCDF data model. In the tables, properties of type "identifier" MUST be strings obeying the definition of "NCName" from [XMLSCHEMA]. Properties of type "string" and "text" MUST NOT include XHTML formatting. Properties of type "HTML-enabled text" are string data that MAY include embedded formatting, presentation, and hyperlink structure; if present, these MUST be expressed using Extensible Hypertext Markup Language (XHTML) Basic tags [XHTML]. The core modules noted in [XHTML] and the Presentation module MAY be used for this.

XHTML markup allows authors to specify how the content of certain fields SHOULD be displayed to a user. However, while this allows authors to specify every detail of a field's display, authors MAY use classifiers in these fields, using class attributes in or elements. These tell products the type of content contained within a text block and allow the products to display this content according to their own conventions. Authors may wish to do this so their content fits better with other automated formatting choices made by a product or a stylesheet, and also because some products may not support the complete range of HTML tags in their displays, thus causing some explicit formatting to be ignored. A list of recommended class values appears in Table 2. Authors MAY use class values not included in this list, and it is OPTIONAL for products to have special formatting for any of the listed classifiers. However, both authors and products SHOULD use these class designators when appropriate to provide more effective display of content. Use of these class attribute values SHALL be applicable wherever HTML content can be included in XCCDF documents.

Table 2: Recommended Class Values

Class Value	Meaning
license	Licensing and use information
copyright	Copyright and ownership information
tangent	A block of text that contains tangentially related information (possibly appropriate for inclusion as a sidebar or a pop-up)
warning	Pitfalls or cautions relative to the surrounding text. High-level and general warnings SHOULD appear in designated warning fields, if available.
critical	Content of critical importance to the user

Class Value	Meaning
example	An example of some kind
instructions	Special instructions to the user
default	General information. Empty or absent class attributes also imply "default" appearance. This tag allows authors to explicitly indicate that text should appear in the default format.

6.2.3 Element Identifiers

The elements listed in Table 3 have special conventions around the format of their identifiers (*@id* attribute). Authors MUST follow these conventions because they make it easier to preserve the global uniqueness of the resulting identifiers.

Table 3: Element Identifier Format Conventions

Element	Format Convention
Benchmark	xccdf_*namespace*_benchmark_*name*
Profile	xccdf_*namespace*_profile_*name*
Group	xccdf_*namespace*_group_*name*
Rule	xccdf_*namespace*_rule_*name*
Value	xccdf_*namespace*_value_*name*
TestResult	xccdf_*namespace*_testresult_*name*
Tailoring	xccdf_*namespace*_tailoring_*name*

In Table 3, *namespace* contains a valid reverse-DNS style string (limited to letters, numbers, periods, and the hyphen character) that is associated with the content author. Examples include "com.acme.finance" and "gov.tla". These *namespace* strings MAY have any number of parts, and benchmark consumers processing them SHALL treat them as case-insensitive. (That is, com.ABC is considered identical to com.abc.) This association with the content author is solely to ensure that different content authors will not use the same namespace value, as this could lead to identifier collisions. Readers should not automatically infer any special authority or trust of the content based on the namespace since reuse of the element or changes to referenced content may not align with the author's original intent. The *name* component in the format conventions MAY be any NCName-compliant string [XMLSCHEMA]. Identifier fields other than those noted in Table 3 have no additional formatting conventions beyond compliance with the NCName data type.

See the example *<xccdf:Profile>* in Section 6.5.1 for several examples of the element identifier format conventions.

6.2.4 <xccdf:metadata> Element

XCCDF supports inclusion of metadata about a document, including title, name of author(s), organization providing the guidance, version number, release date, update URL, and a description. This is particularly useful for facilitating the discovery and retrieval of XCCDF checklists from public repositories.

When used, the *<xccdf:metadata>* element SHALL contain document metadata expressed in XML. The metadata element can appear one or more times as a child of the *<xccdf:Benchmark>*, *<xccdf:Rule>*, *<xccdf:Group>*, *<xccdf:Value>*, *<xccdf:Profile>*, *<xccdf:TestResult>*, *<xccdf:rule-result>*, and *<xccdf:Tailoring>* elements. The *<xccdf:Benchmark>* element SHOULD contain metadata information formatted using the Dublin

Core Metadata Initiative (DCMI) Simple DC Element specification, as described in [DCES] and [DCXML]. Benchmark consumers SHOULD be prepared to process Dublin Core metadata in the *<xccdf:metadata>* element. An example of the Dublin Core format is shown below.

```
<xccdf:metadata xmlns:dc="http://purl.org/dc/elements/1.1/">
    <dc:title>Security Benchmark for Ethernet Hubs</dc:title>
    <dc:creator>James Smith</dc:creator>
    <dc:publisher>Center for Internet Security</dc:publisher>
    <dc:subject>network security for layer 2 devices</dc:subject>
</xccdf:metadata>
```

Elements that support the *<xccdf:metadata>* element MAY use any desired metadata format; for *<xccdf:Benchmark>*, this is in addition to the Dublin Core recommendations already mentioned. Any XML metadata structures, including ad hoc structures, MAY be included in an *<xccdf:metadata>* element. Because any structures can appear in this field, authors SHOULD tag metadata with the *xmlns* prefix [XMLNAME] and, if available, the *@xsi:schemaLocation* attribute in order to identify the metadata structure utilized. Metadata SHOULD comply with existing commercial or government metadata specifications to allow benchmarks to be discovered and indexed.

Metadata is a powerful feature for authors. However, XCCDF puts some limits on the use of metadata:

- Metadata SHOULD NOT replace the functionality of existing fields within XCCDF. If the information matches the common use of some other XCCDF field, the information MUST appear in that XCCDF field. It MAY also appear in the *<xccdf:metadata>* element.
- Metadata SHALL NOT change the processing model as outlined in Section 7. Benchmark consumer products SHALL perform XCCDF assessments in the same way regardless of the presence of metadata. Metadata MAY still be used to support assessment features that are outside the purview of XCCDF—for example, to hold post-processing instructions to be performed on the completed XCCDF results or to display additional information to the user before or during the assessment.
- Metadata SHOULD NOT alter the character content of the XCCDF properties used to generate output during document generation. Contents of the *<xccdf:metadata>* element MAY be added to the output above and beyond the XCCDF properties. Metadata MAY contain instructions that cause different stylistic conventions to be adopted in the conversion of an XCCDF document to prose.

6.2.5 Platform Names

The Common Platform Enumeration (CPE) specification allows a specific hardware or software platform to be identified by a unique name. CPE names can express only single platforms (e.g., "cpe:2.3:o:microsoft:windows_xp:*:gold:professional:*:*:*:*:*" for Microsoft Windows XP Professional Edition). CPE applicability language statements can express complex logical constructions of CPE names.

<xccdf:Benchmark>, *<xccdf:Profile>*, *<xccdf:Rule>*, and *<xccdf:Group>* elements may be qualified by applicable platform using the *<xccdf:platform>* element. *<xccdf:TestResult>* elements may also include *<xccdf:platform>* elements. The *<xccdf:platform>* element's *@idref* attribute MAY hold a reference to either a CPE name or the *@id* attribute of a CPE applicability language expression that SHALL be defined as a child of the

<xccdf:Benchmark> using a *<cpe2:platform-specification>* element (see Table 4). (The syntax for referring to a local CPE applicability language identifier SHOULD be a "#" character before the identifier, such as "#cpeid1".)

Within XCCDF documents, all CPE names SHALL comply with the CPE 2.3 Naming specification [IR7695], and all CPE applicability language expressions SHALL comply with the CPE 2.3 Applicability Language specification [IR7698]. CPE 2.0 names MAY be used for backwards compatibility, but their use has been deprecated for XCCDF 1.2. All CPE 2.3 names and applicability language expressions in XCCDF documents SHOULD use formatted string bindings but MAY use URI bindings instead, both as defined in [IR7695].

Here is an example of a *<cpe2:platform-specification>* element:

```
<cpe2:platform-specification>
  <cpe2:platform id="xp_and_acrobat_9.0">
    <cpe2:logical-test operator="AND" negate="false">
      <cpe2:fact-ref
        name="cpe:2.3:o:microsoft:windows_xp:*:*:*:*:*:*:*:*"/>
      <cpe2:fact-ref
        name="cpe:2.3:a:adobe:acrobat:9.0:*:*:*:*:*:*:*"/>
    </cpe2:logical-test>
  </cpe2:platform>
</cpe2:platform-specification>
<xccdf:platform idref="#xp_and_acrobat_9.0"/>
```

If an *<xccdf:Profile>*, *<xccdf:Rule>*, or *<xccdf:Group>* does not possess any *<xccdf:platform>* elements, then it SHALL apply to the same set of platforms as its nearest enclosing ancestor *<xccdf:Group>* or *<xccdf:Benchmark>*. If the enclosing ancestor has no *<xccdf:platform>* element, then consult the next enclosing ancestor, repeating until either an ancestor has an *<xccdf:platform>* element or the *<xccdf:Benchmark>* element is reached. If no ancestor, including the *<xccdf:Benchmark>* element, possesses any *<xccdf:platform>* elements, then the *<xccdf:Profile>*, *<xccdf:Rule>*, or *<xccdf:Group>* SHALL nominally apply to all platforms.

6.2.6 <xccdf:reference> Element

The *<xccdf:reference>* element provides a reference to a document or resource where the user can learn more about the subject of the parent element. It may be included within *<xccdf:Benchmark>*, *<xccdf:Rule>*, *<xccdf:Group>*, *<xccdf:Value>*, and *<xccdf:Profile>* elements. When used, it SHALL have either a simple string value or a value consisting of simple Dublin Core elements as described in [DCXML]. It MAY also have an *@href* attribute that gives a URL for the referenced resource. References SHOULD be given as Dublin Core descriptions; a bare string MAY be used for simplicity. If a bare string appears, then it is taken to be the string content for a *<dc:title>* element. For more information, consult [DCES]. Multiple *<xccdf:reference>* elements MAY appear; a benchmark consumer product MAY concatenate them or put them into a reference list, and MAY choose to number them.

6.2.7 <xccdf:signature> Element

XCCDF supports a digital signature mechanism whereby XCCDF document users can validate the integrity, origin, and authenticity of documents. XCCDF provides a means to hold such signatures and a uniform method for applying and validating them [XMLDSIG].

The *<xccdf:signature>* element MAY hold an enveloped digital signature asserting authorship and allowing verification of the integrity of associated data (e.g., its parent element, other documents, portions of other documents). The signature SHALL apply only after inclusion (i.e., XInclude) processing. The *<xccdf:signature>* element is OPTIONAL, and it MAY appear as a child of the *<xccdf:Benchmark>*, *<xccdf:Rule>*, *<xccdf:Group>*, *<xccdf:Value>*, *<xccdf:Profile>*, *<xccdf:TestResult>*, or *<xccdf:Tailoring>* elements.

Any digital signature format employed for XCCDF MUST be capable of identifying the signer, storing all information needed to verify the signature (usually a certificate or certificate chain), and detecting any change to the signed content. XCCDF products that support signatures MUST support the W3C XML-Signature standard enveloped signatures, as defined in [XMLDSIG].

If multiple signatures are needed in an XCCDF document, at most one of them MAY be enveloped; all others MUST be detached [XMLDSIG Section 2].

The single child element of the *<xccdf:signature>* element SHOULD be *<dsig:Signature>* and that *<dsig:Signature>* SHOULD contain one or more *<dsig:Reference>* elements indicating each XML element to be included in the signature. The *<dsig:Reference>* URI SHOULD be a relative URI to the *@Id* attribute value of the enclosing object (prefixed by "#"). For example, if the Id="abc", then the *<Reference>* URI would be "#abc".

6.2.8 Status Tracking

XCCDF provides two elements for status tracking: *<xccdf:status>* and *<xccdf:dc-status>*. Both elements are available for *<xccdf:Benchmark>*, *<xccdf:Rule>*, *<xccdf:Group>*, *<xccdf:Value>*, *<xccdf:Profile>*, and *<xccdf:Tailoring>* elements.

The intent of the *<xccdf:status>* element is to record the maturity or consensus level for its parent element. Permissible values are "incomplete" (under initial development), "draft" (released in draft state), "interim" (revised and in the process of being finalized), "accepted" (released as final), and "deprecated" (no longer needed). If there is more than one *<xccdf:status>* element for a single parent element, then every instance of the *<xccdf:status>* element MUST have a date attribute, and the *<xccdf:status>* element with the latest date SHALL be considered the latest status.

The *<xccdf:dc-status>* element holds additional status information using the Dublin Core format, expressed as elements of the DCMI Simple DC Element specification, as described in [DCES] and [DCXML].

6.2.9 Text Substitution

XCCDF provides capabilities to perform substitution within certain properties of an *<xccdf:Benchmark>*.

An *<xccdf:plain-text>* element is a reusable text block for reference by the *<xccdf:sub>* element. This allows text to be defined once and then reused multiple times. Each *<xccdf:plain-text>* element MUST have a unique NCName identifier. The identifiers for *<xccdf:plain-text>* MUST NOT collide with item identifiers within the *<xccdf:Benchmark>*.

The *<xccdf:sub>* element represents a reference to a parameter value that may be set during tailoring. The *<xccdf:sub>* element is supported by several elements, which are noted as they are defined throughout the rest of this section. The *<xccdf:sub>* element SHALL NOT have any content. It MUST have a single *@idref* attribute, which MUST equal the *@id* attribute of an *<xccdf:Value>* element or an *<xccdf:plain-text>* element. During document generation, each instance of the *<xccdf:sub>* element will be replaced by the text value of the referenced element's contents. If an *<xccdf:Value>* element is referenced, the *<xccdf:sub>* element MAY have a *@use* attribute that indicates whether the *<xccdf:Value>* element's title or value should replace the *<xccdf:sub>* element. If an *<xccdf:plain-text>* element is referenced, the *@use* attribute is ignored and the body of the *<xccdf:plain-text>* element is always used in the substitution.

See Section 7.2.3.6.3 for additional requirements related to substitution processing.

6.2.10 @xml:lang Attribute

Some textual elements of XCCDF, such as titles and descriptions, support the *@xml:lang* attribute to denote the language they use.[1] Elements that have an *@xml:lang* attribute may appear multiple times within a single parent element to support multiple languages. If there are multiple instances of such a textual element within a single parent element, each instance SHOULD have a value for its *@xml:lang* attribute. Each value SHOULD specify the natural language locale for which the instance is written (e.g., "en" for English, "fr" for French). Values SHOULD be valid language tags as defined by [RFC5646]. Although any valid language tag MAY be used, only tags containing language codes (with or without region codes) SHOULD be used. All language and region codes used SHOULD be in the *Internet Assigned Numbers Authority (IANA) Language Subtag Registry* [ILSR]. An example of using the *@xml:lang* attribute is shown below.

```
<xccdf:Value id="xccdf_org.example_value_web-server-port" type="number">
   <xccdf:title xml:lang="en">Web Server Port</xccdf:title>
   <xccdf:title xml:lang="fr">Le Port Du Web Serveur</xccdf:title>
   <xccdf:question xml:lang="en">
      What is the web server's TCP port?
   </xccdf:question>
   <xccdf:question xml:lang="fr">
      Quel est le port du TCP du web serveur?
   </xccdf:question>
   <xccdf:value>80</xccdf:value>
</xccdf:Value>
```

If an element other than *<xccdf:Benchmark>* that supports the *@xml:lang* attribute omits it, the *@xml:lang* attribute of the nearest ancestor element that has the *@xml:lang* attribute SHALL be consulted.

[1] See http://www.w3.org/TR/xml/#sec-lang-tag for additional information on the `@xml:lang` attribute.

6.3 <xccdf:Benchmark>

6.3.1 Basics

Each XCCDF benchmark document SHALL consist of a single root *<xccdf:Benchmark>* element, which encloses the entire benchmark. The example below illustrates the outermost structure of an XCCDF XML document.

```
<?xml version="1.0" ?>
<xccdf:Benchmark id="xccdf_org.example_benchmark_example1" xml:lang="en"
Id="toSign"
      xmlns:htm="http://www.w3.org/1999/xhtml"
      xmlns:xccdf="http://checklists.nist.gov/xccdf/1.2"
      xmlns:cpe2-dict="http://cpe.mitre.org/dictionary/2.0"/>
  <xccdf:status date="2010-06-01">draft</xccdf:status>
  <xccdf:title>Example Benchmark File</xccdf:title>
  <xccdf:description>
     A <htm:b>Small</htm:b> Example
  </xccdf:description>
  <xccdf:platform idref="cpe:2.3:o:cisco:ios:12.0:*:*:*:*:*:*:*"/>
  <xccdf:version>0.2</xccdf:version>
  <xccdf:reference href="http://www.ietf.org/rfc/rfc822.txt">
      Standard for the Format of ARPA Internet Text Messages
  </xccdf:reference>
</xccdf:Benchmark>
```

Figure 1 illustrates a typical internal structure for a Benchmark.

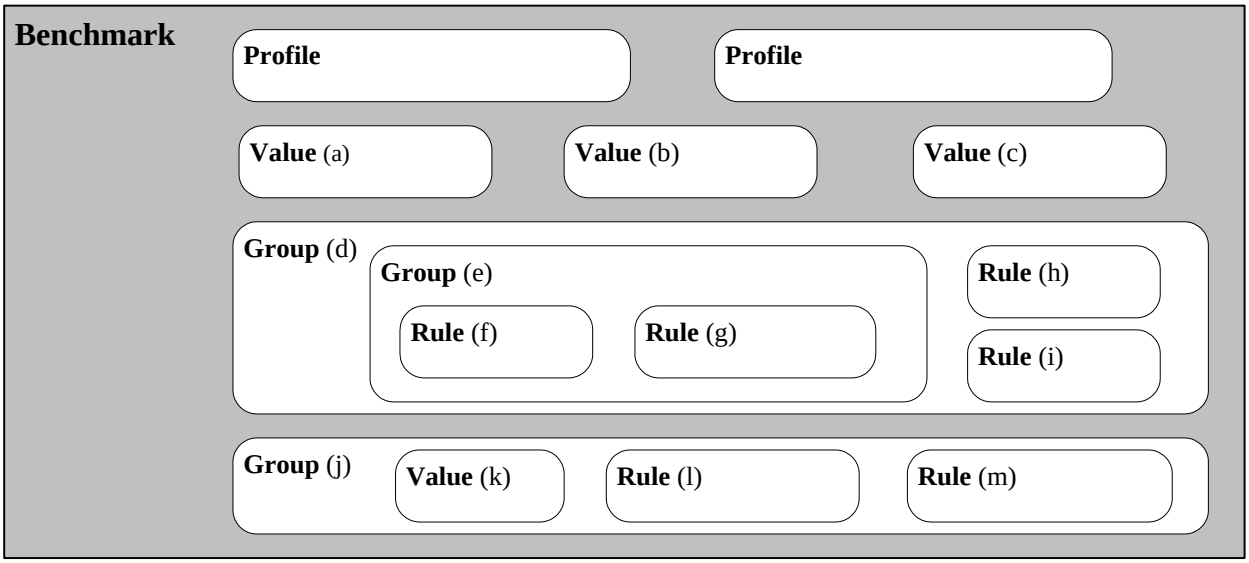


Figure 1: Typical Structure of a Benchmark

The possible inheritance relations between Rule and Value instances are constrained by the tree structure of the Benchmark. All extension relationships MUST be resolved before the Benchmark can be applied. An item MAY only extend another item of the same type that is visible from its scope. In other words, an

item Y MAY extend a base item X, as long as they are the same type and one of the following visibility conditions holds:

- X is a direct child of the Benchmark.
- X is a direct child of a Group which is also an ancestor[2] of Y.
- X is a direct child of a Group which is extended by any ancestor of Y.

For example, in the Benchmark structure shown in Figure 1, it would be legal for Rule (g) to extend Rule (f) or extend Rule (h). It would not be legal for Rule (i) to extend Rule (m), because (m) is not visible from the scope of (i). It would not be legal for Rule (l) to extend Value (k), because they are not of the same type.

The ability for an item to be extended gives benchmark authors the ability to create variations or specialized versions of items without making copies. An extending item can inherit properties from the extended (base item) and sometimes can override properties with new values if needed; however, there are several restrictions on these inheritance capabilities. For example, some properties have different inheritance behaviors depending on how their `@override` attribute is set. Also, group extension has been deprecated as of XCCDF 1.2 (see Table 42) and authors are strongly discouraged from using it due to the risks it poses to interoperability. See Section 7.2.2 for more information on inheritance.

Circular dependencies caused by extension MUST NOT be defined.

6.3.2 Properties

Table 4 describes the `<xccdf:Benchmark>` element's properties.

In this table and in similar tables throughout the rest of the section, the Count column indicates how many times each property SHALL be permitted within a single instance of the parent element. Each Count is expressed as either a single value or a range of values. If the Count entry for a property includes the value 0, the property is OPTIONAL, otherwise the property is REQUIRED. If the Count entry for a property includes the value n, the property MAY be used more than one time, with no upper bound. Here are some examples:

- 1: the property shall be used once
- 1-n: the property shall be used at least once and may be used more than once
- 0-1: the property is optional and may be used at most once
- 0-n: the property is optional and may be used more than once

Table 4: <xccdf:Benchmark> Element Properties

Property	Type	Count	Description
status (element)	*special*	1-n	Status of the benchmark (REQUIRED) and date at which it attained that status (OPTIONAL). The status SHOULD indicate the level of maturity or consensus for the benchmark. See Section 6.2.8.
dc-status (element)	*special*	0-n	Holds additional status information using the Dublin Core format. See Section 6.2.8.

[2] See http://www.w3.org/TR/xpath/#axes for the XPath definition of the term "ancestor".

Property	Type	Count	Description
title (element)	string	0-n	Title of the benchmark; a benchmark SHOULD have a title. It MAY have an *@xml:lang* attribute (see Section 6.2.10) and/or an *@override* attribute (see Section 6.3.1).
description (element)	HTML-enabled text	0-n	Text that describes the benchmark; a benchmark SHOULD have a description. It MAY have one or more *<xccdf:sub>* elements (see Section 6.2.9), an *@override* attribute (see Section 6.3.1), and/or an *@xml:lang* attribute (see Section 6.2.10).
notice (element)	HTML-enabled text	0-n	Legal notices (licensing information, terms of use, etc.), copyright statements, warnings, and other advisory notices about this benchmark and its use. Each element SHALL have a unique NCName identifier as its *@id* attribute. Each element MAY have an *@xml:lang* attribute (see Section 6.2.10) and/or an *@xml:base* attribute (which defines the context for all relative URIs within the *<xccdf:Benchmark>* element).
front-matter (element)	HTML-enabled text	0-n	Introductory matter for the beginning of the benchmark document; intended for use during Document Generation processing only (see Section 7.1). It MAY have one or more *<xccdf:sub>* elements (see Section 6.2.9), an *@override* attribute (see Section 6.3.1), and/or an *@xml:lang* attribute (see Section 6.2.10).
rear-matter (element)	HTML-enabled text	0-n	Concluding material for the end of the benchmark document; intended for use during Document Generation processing only (see Section 7.1). It MAY have one or more *<xccdf:sub>* elements (see Section 6.2.9), an *@override* attribute (see Section 6.3.1), and/or an *@xml:lang* attribute (see Section 6.2.10).
reference (element)	*special*	0-n	Supporting references for the benchmark document. See Section 6.2.6.
plain-text (element)	string	0-n	Definitions for reusable text blocks, each with a unique identifier. See Section 6.2.9.
cpe2: platform-specification (element)	*special*	0-1	A list of identifiers for complex platform definitions, written in CPE applicability language format. Authors MAY define complex platforms within this element, and then use their locally unique identifiers anywhere in the *<xccdf:Benchmark>* element in place of a CPE name. See Section 6.2.5.
platform (element)	string	0-n	Applicable platforms for this benchmark. Authors SHOULD use the element to identify the systems or products to which the benchmark applies. See Section 6.2.5.
version (element)	string	1	Version number of the benchmark, with an OPTIONAL *@time* timestamp attribute (when the version was completed) and an OPTIONAL *@update* URI attribute (where updates may be obtained).
metadata (element)	*special*	0-n	XML metadata for the benchmark. Benchmark metadata allows authorship, publisher, support, and other information to be embedded in a benchmark. See Section 6.2.4.
model (element)	URI	0-n	URIs of suggested scoring models to be used when computing a score for this benchmark. See Section 7.3.2 for more information on models. Parameters MAY be provided as needed for particular models through the *@param* attribute. *@param* attributes with equal name values SHALL NOT appear as children of the same element.
Profile (element)	*special*	0-n	Profiles that reference and customize sets of items in the benchmark; see Section 6.5.
Value (element)	*special*	0-n	Parameter values that support Rules and descriptions in the benchmark; see Section 6.4.5.
Group (element)	*special*	0-n	Groups that comprise the benchmark; each MAY contain additional Values, Rules, and other Groups. See Section 6.4.3.
Rule (element)	*special*		Rules that comprise the benchmark; see Section 6.4.4.

Property	Type	Count	Description
TestResult (element)	*special*	0-n	Benchmark test result records (one per benchmark run); see Section 6.6.
signature (element)	*special*	0-1	A digital signature asserting authorship and allowing verification of the integrity of the benchmark. See Section 6.2.7.
id (attribute)	*special*	1	Unique benchmark identifier. See Section 6.2.3.
Id (attribute)	*special*	0-1	An identifier used for referencing elements included in an XML signature. See Section 6.2.7.
resolved (attribute)	boolean	0-1	True if benchmark has already undergone the resolution process (default: false). See Section 7.2.2.
style (attribute)	string	0-1	Name of a benchmark authoring style or set of conventions or constraints to which this benchmark conforms (e.g., "SCAP 1.2").
style-href (attribute)	URI	0-1	URL of a supplementary stylesheet or schema extension that can be used to verify conformance to the named style.
xml:lang (attribute)	*special*	0-1	The language for the benchmark; see Section 6.2.10.

The properties that comprise an *<xccdf:Benchmark>* or items are an ordered sequence of property values. The order of *<xccdf:Group>* and *<xccdf:Rule>* child elements matters for the appearance of a generated document, so *<xccdf:Group>* and *<xccdf:Rule>* child elements MAY be freely intermingled. All other child elements MUST appear in the order shown in Table 4, and multiple instances of a child element MUST be adjacent.

6.4 Item Elements

6.4.1 Properties

Table 5 describes the properties common to all three classes of items: *<xccdf:Group>*, *<xccdf:Rule>*, and *<xccdf:Value>* elements.

Table 5: Item Element Properties

Property	Type	Count	Description
status (element)	*special*	0-n	Status of the item and date at which it attained that status. Benchmark authors MAY use this element to record the maturity or consensus level for item elements in the benchmark. If an item does not have an explicit status given, then its status SHALL be that of its parent. See Section 6.2.8.
dc-status (element)	*special*	0-n	Holds additional status information using the Dublin Core format. See Section 6.2.8.
version (element)	string	0-1	Version number of the item, with an OPTIONAL *@time* timestamp attribute (when the version was completed) and an OPTIONAL *@update* URI attribute (where updates may be obtained).
title (element)	string	0-n	Title of the item. Every item SHOULD have a title, because this helps people understand the purpose of the item. It MAY have one or more *<xccdf:sub>* elements (see Section 6.2.9), an *@xml:lang* attribute (see Section 6.2.10), and/or an *@override* attribute (see Section 6.3.1).
description (element)	HTML-enabled text	0-n	Text that describes the item. It MAY have one or more *<xccdf:sub>* elements (see Section 6.2.9), an *@override* attribute (see Section 6.3.1), and/or an *@xml:lang* attribute (see Section 6.2.10).

Property	Type	Count	Description
warning (element)	HTML-enabled text	0-n	A note or caveat about the item intended to convey important cautionary information for the benchmark user (e.g., "Complying with this rule will cause the system to reject all IP packets"). If multiple warning elements appear, benchmark consumers SHOULD concatenate them for generating reports or documents. Benchmark consumers MAY present this information in a special manner in generated documents. It MAY have one or more `<xccdf:sub>` elements (see Section 6.2.9), an `@override` attribute (see Section 6.3.1), and/or an `@xml:lang` attribute (see Section 6.2.10). Also, see Section 6.4.2 for the possible values of the warning element's OPTIONAL `@category` attribute, which provides a hint as to the nature of the warning.
question (element)	string	0-n	Interrogative text to present to the user during tailoring. It MAY also be included into a generated document. For `<xccdf:Rule>` and `<xccdf:Group>` elements, the question text SHOULD be a simple binary (yes/no) question because it is supporting the selection aspect of tailoring. For `<xccdf:Value>` elements, the question SHOULD solicit the user to provide a specific value. Tools MAY also display constraints on values and any defaults as specified by the other <xccdf:Value> properties (see Section 6.4.5). It MAY have an `@override` attribute (see Section 6.3.1) and/or an `@xml:lang` attribute (see Section 6.2.10).
reference (element)	*special*	0-n	References where the user can learn more about the subject of this item. See Section 6.2.6.
metadata (element)	*special*	0-n	XML metadata associated with this item, such as sources, special information, or other details. See Section 6.2.4.
abstract (attribute)	boolean	0-1	If true, then this item is abstract and exists only to be extended (default: false). ***The use of this attribute for `<xccdf:Group>` elements is deprecated and should be avoided (see* Table 42*).***
cluster-id (attribute)	identifier	0-1	An identifier to be used as a means to identify (refer to) related `<xccdf:Group>`, `<xccdf:Rule>`, and `<xccdf:Value>` elements throughout the `<xccdf:Benchmark>`. It designates membership in a cluster of items, which are used for controlling items via profiles. All the items with the same cluster identifier belong to the same cluster. A selector in an `<xccdf:Profile>` MAY refer to a cluster, thus making it easier for authors to create and maintain profiles in a complex benchmark.
extends (attribute)	identifier	0-1	The identifier of an item on which to base this item. If present, it MUST have a value equal to the `@id` attribute of another item. See Section 6.3.1 and Section 7.2.2. ***The use of this attribute for `<xccdf:Group>` elements is deprecated and should be avoided (see* Table 42*).***
hidden (attribute)	boolean	0-1	If this item should be excluded from any generated documents (default: false). For example, an author might set the `@hidden` attribute on an incomplete item in a draft benchmark. The item may still be used during assessments, but it SHALL NOT appear in generated documents.
prohibitChanges (attribute)	boolean	0-1	If benchmark producers should prohibit changes to this item during tailoring (default: false). An author SHOULD use this when they do not want to allow end users to change the item. For example, benchmark users may use this to define items that are integral to compliance, or enterprise security officers may use it to constrain a benchmark so it reflects organizational policies.
xml:lang (attribute)	*special*	0-1	The language for the item; see Section 6.2.10.
xml:base (attribute)	*special*	0-1	The context for all relative URIs within the item.

Property	Type	Count	Description
Id (attribute)	*special*	0-1	An identifier used for referencing elements included in an XML signature. See Section 6.2.7.

In addition to the properties listed in Table 5, *<xccdf:Group>* and *<xccdf:Rule>* elements also support the properties listed in Table 6. See Sections 6.4.3, 6.4.4, and 6.4.5 for information on additional properties specific to *<xccdf:Group>*, *<xccdf:Rule>*, and *<xccdf:Value>* elements, respectively.

Table 6: Properties Specific to <xccdf:Group> and <xccdf:Rule> Elements

Property	Type	Count	Description
rationale (element)	HTML-enabled text	0-n	Descriptive text giving rationale or motivations for abiding by this group/rule (i.e., why it is important to the security of the target platform). It MAY have one or more *<xccdf:sub>* elements (see Section 6.2.9), an *@override* attribute (see Section 6.3.1), and/or an *@xml:lang* attribute (see Section 6.2.10).
platform (element)	*special*	0-n	Platforms to which this group/rule applies. See Section 6.2.5.
requires (element)	*special*	0-n	The identifiers of other *<xccdf:Group>* or *<xccdf:Rule>* elements in the *<xccdf:Benchmark>* that MUST be selected for this group/rule to be evaluated and scored properly. Each *<xccdf:requires>* element specifies a list of one or more required items by their identifiers, using the *@idref* attribute. If at least one of the specified *<xccdf:Group>* or *<xccdf:Rule>* elements is selected, the requirement is met. An *<xccdf:requires>* element that looked like *<xccdf:requires idref="A B C">* could be read as "requires that item A or item B or item C be selected".
conflicts (element)	identifier	0-n	The identifier of another *<xccdf:Group>* or *<xccdf:Rule>* in the *<xccdf:Benchmark>* that MUST be unselected for this group/rule to be evaluated and scored properly. Each *<xccdf:conflicts>* element specifies a single conflicting item using its *@idref* attribute. If the specified *<xccdf:Group>* or *<xccdf:Rule>* element is not selected, the requirement is met.
selected (attribute)	boolean	0-1	If true, this group/rule SHALL be selected to be processed as part of the *<xccdf:Benchmark>* when it is applied to a target system (default: true). An unselected group SHALL NOT be processed, and its contents SHALL NOT be processed either (i.e., all descendants of an unselected group are implicitly unselected). An unselected rule SHALL NOT be checked and SHALL NOT contribute to scoring. MAY be overridden by a profile; see Section 6.5.3.
weight (attribute)	decimal	0-1	The relative scoring weight of this group/rule, for computing a score, expressed as a non-negative real number (0.0 or greater, maximum 3 digits, default 1.0). It denotes the importance of a group/rule. Under some scoring models, scoring is computed independently for each collection of sibling groups and rules, then normalized as part of the overall scoring process. See Section 7.3.2 for more information on scoring. MAY be overridden by a profile; see Section 6.5.3.

6.4.2 <xccdf:warning> Element

If the *<xccdf:warning>* element's *@category* attribute is used, it MUST have one of the values specified in Table 7.

Table 7: <xccdf:warning> Element @category Attribute Values

Value	Meaning
general	Broad or general-purpose warning (default)
functionality	Warning about possible impacts to functionality or operational features
performance	Warning about changes to target system performance or throughput
hardware	Warning about hardware restrictions or possible impacts to hardware
legal	Warning about legal implications
regulatory	Warning about regulatory obligations or compliance implications
management	Warning about impacts to the management or administration of the target system
audit	Warning about impacts to audit or logging
dependency	Warning about dependencies between this element and other parts of the target system, or version dependencies

6.4.3 <xccdf:Group> Element

An *<xccdf:Group>* element contains descriptive information about a portion of a benchmark, as well as *<xccdf:Rule>*, *<xccdf:Value>*, and/or other *<xccdf:Group>* elements. Table 8 describes the *<xccdf:Group>* element's properties (in addition to the properties in Table 5 and Table 6).

Table 8: <xccdf:Group> Element Properties

Property	Type	Count	Description
Value (element)	Value	0-n	Values that belong to this group; see Section 6.4.5.
Group (element)	Group	0-n	Sub-groups under this group; see Section 6.4.3.
Rule (element)	Rule		Rules that belong to this group; see Section 6.4.4.
signature (element)	*special*	0-1	A digital signature asserting authorship and allowing verification of the integrity of the group. See Section 6.2.7.
id (attribute)	*special*	1	Unique element identifier; SHALL be used by other elements to refer to this element. See Section 6.2.3.

6.4.4 <xccdf:Rule> Element

6.4.4.1 Basics

An *<xccdf:Rule>* element defines a single item to be checked as part of a benchmark, or an extendable base definition for such items. The *<xccdf:check>* child element of an *<xccdf:Rule>* specifies how to verify compliance with a security practice or guideline. See Table 9 and Section 6.4.4.4 for more information on the *<xccdf:check>* element.

The example below shows a very simple *<xccdf:Rule>* element.

```
 <xccdf:Rule id="xccdf_org.example_rule_pwd-perm" selected="1" weight="6.5"
severity="high">
   <xccdf:title>Password File Permission</xccdf:title>
   <xccdf:description>Check the access control on the password file.
     Normal users should not be able to write to it.
   </xccdf:description>
   <xccdf:requires idref="xccdf_org.example_rule_passwd-exists"/>
```

```
  <xccdf:fixtext>
    Set permissions on the passwd file to owner-write, world-read
  </xccdf:fixtext>
  <xccdf:fix strategy="restrict" reboot="0" disruption="low">
        chmod 644 /etc/passwd
  </xccdf:fix>
  <xccdf:check system="http://oval.mitre.org/XMLSchema/oval-definitions-5">
    <xccdf:check-content-ref href="ovaldefs.xml" name="oval:org.example:def:123"/>
  </xccdf:check>
</xccdf:Rule>
```

One of XCCDF's main features is the organization and selection of target-applicable groups and rules for performing security and operational checks on systems. XCCDF can access granular and expressive mechanisms for assessing the state of a system according to the rule criteria. Examples of these mechanisms are definitions expressed in the Open Vulnerability and Assessment Language (OVAL) and questionnaires expressed in the Open Checklist Interactive Language (OCIL). These checking mechanisms follow the conceptual model of collecting or acquiring the state of a target system, and then assessing the state for conformance to conditions and criteria expressed as rules.

XCCDF MAY use rule checking systems other than (or in addition to) OVAL and OCIL. To facilitate this, XCCDF supports referencing by including the appropriate check content location and check reference in the *<xccdf:check>* element. Rule checking systems are defined separately from XCCDF itself so that both XCCDF and the rule checking system can evolve and be used independently. It is helpful for rule checking mechanisms used by XCCDF to comply with the following:

- **The mechanism can express both positive and negative criteria.** A positive criterion means that if certain conditions are met, then the system satisfies the check, while a negative criterion means that if the conditions are met, the system fails the check. Experience has shown that both kinds are necessary for checks.
- **The mechanism can express Boolean combinations of criteria.** It is often impossible to express a high-level security property as a single quantitative or qualitative statement about a system's state. Therefore, the ability to combine statements with 'and' and 'or' is critical.
- **The mechanism can incorporate tailoring values set by the user.** Value modification is important for XCCDF document tailoring, including substitution of tailored values as well as tailoring of a selected set of rules.

6.4.4.2 Properties

Table 9 describes the *<xccdf:Rule>* element's properties (in addition to the properties in Table 5 and Table 6).

Table 9: <xccdf:Rule> Element Properties

Property	Type	Count	Description
ident (element)	string	0-n	A globally meaningful identifier for this rule. MAY be the name or identifier of a security configuration issue or vulnerability that the rule remediates. Has an associated URI that SHALL denote the organization or naming scheme that assigned the name (see Table 10 for assigned URIs). By setting an *<xccdf:ident>* element on a rule, the benchmark author effectively declares that the rule instantiates, implements, or remediates the issue for which the name was assigned. For example, the *<xccdf:ident>* value might be a CVE identifier; the rule could be a check that the target platform was not subject to the vulnerability named by the CVE identifier, and the URI would be that of the CVE website. An *<xccdf:ident>* element MAY also have other attributes from other namespaces in order to provide additional metadata for the given identifier.
impact-metric (element)	string	0-1	The potential impact of failure to conform to the rule, expressed as a CVSS 2.0 base vector (defined by NIST IR 7435). ***The property is deprecated and its use should be avoided (see* Table 42*).*** A suggested alternate location for impact metric data is the *<xccdf:metadata>* element within the *<xccdf:Rule>*.
profile-note (element)	HTML-enabled text	0-n	Text that describes special aspects of the rule related to one or more profiles. This allows an author to document things within rules that are specific to a given profile, and then select the appropriate text based on the selected profile and display it to the reader. The element MUST have a *@tag* attribute that holds an identifier; an *<xccdf:Profile>* MAY refer to this tag through the *<xccdf:Profile>* *@note-tag* attribute. The element also MAY have one or more *<xccdf:sub>* elements (see Section 6.2.9) and/or an *@xml:lang* attribute (see Section 6.2.10).
fixtext (element)	*special*	0-n	Data that describes how to bring a target system into compliance with this rule. Each *<xccdf:fixtext>* element MAY be associated with one or more *<xccdf:fix>* element values. See Section 6.4.4.5.
fix (element)	*special*	0-n	A command string, script, or other system modification statement that, if executed on the target system, can bring it into full, or at least better, compliance with this rule. See Section 6.4.4.5.
check (element)	*special*	(1-n instances of check) XOR (1 instance of complex-check)	The definition of, or a reference to, the target system check needed to test compliance with this rule. Sibling *<xccdf:check>* elements MUST have different values for the combination of their *@selector* and *@system* attributes, and different values for their *@id* attribute (if any). See Section 6.4.4.4.
complex-check (element)	*special*		A boolean expression composed of operators (and, or, not) and individual checks. See Section 6.4.4.4.
signature (element)	*special*	0-1	A digital signature asserting authorship and allowing verification of the integrity of the rule. See Section 6.2.7.
role (attribute)	string	0-1	The rule's role in scoring and reporting. It MAY be one of the following: • "full" (if the rule is selected, then check it and let the result contribute to the score and appear in reports) (default) • "unscored" (if the rule is selected, then check it and include the results in any report, but do not include the result in score computations) • "unchecked" (if the rule is selected, then do not check it; just force the result status to "notchecked") MAY be overridden by a profile.
id (attribute)	*special*	1	Unique element identifier; SHALL be used by other elements to refer to this element. See Section 6.2.3.

Property	Type	Count	Description
severity (attribute)	string	0-1	Severity level code to be used for metrics and tracking. When used, it SHALL have one of the following values: • "unknown" (severity not defined) (default) • "info" (rule is informational only; failing the rule does not imply failure to conform to the security guidance of the benchmark) • "low" (not a serious problem) • "medium" (fairly serious problem) • "high" (grave or critical problem) MAY be overridden by a profile; see Section 6.5.3.
multiple (attribute)	boolean	0-1	Applicable in cases where there are multiple instances of a target. For example, a rule may provide a recommendation about the configuration of application user accounts, but an application may have many user accounts. Each account would be considered an instance of the broader assessment target of user accounts. If the `@multiple` attribute is set to true, each instance of the target to which the rule can apply SHOULD be tested separately and the results SHOULD be recorded separately. If `@multiple` is set to false, the test results of such instances SHOULD be combined. If the checking system does not combine these results automatically, the results of each instance SHOULD be ANDed together to produce a single result using the AND truth table in Section 6.4.4.4. (default:false) If the benchmark consumer cannot perform multiple instantiation, or if multiple instantiation of the rule is not applicable for the target system, then the benchmark consumer MAY ignore this attribute. For example, OVAL checks cannot produce separate results for individual instances of a target. See Section 7.2.3.5.2.

6.4.4.3 <xccdf:ident> Elements

Table 10 lists assigned URIs that MAY appear as the value of the `@system` attribute of an `<xccdf:ident>` element. If an identification system included in Table 10 is being specified, the `@system` attribute's value SHALL correspond to the appropriate URI in the table. An author MAY create a new URI for an identification system not listed in the table; this URI SHALL NOT duplicate any of the Table 10 URI values.

Table 10: Assigned Values for the @system Attribute of an <xccdf:ident> Element

URI	Identifier Value Description
http://cce.mitre.org	Common Configuration Enumeration (CCE) – the identifier value MUST be a CCE version 5 number
http://cpe.mitre.org	CPE – the identifier value MUST be a CPE version 2.0 or 2.3 name
http://cve.mitre.org	CVE – the identifier value MUST be a CVE number
http://www.cert.org	CERT Coordination Center – the identifier value SHOULD be a CERT advisory identifier (e.g., "CA-2004-02")
http://www.kb.cert.org	US-CERT vulnerability notes database – the identifier value SHOULD be a vulnerability note number (e.g., "709220")
http://www.us-cert.gov/cas/techalerts	US-CERT technical cyber security alerts – the identifier value SHOULD be a technical cyber security alert ID (e.g., "TA05-189A")

An `<xccdf:ident>` element MAY also have additional attributes from schemas other than the XCCDF schema. Individual organizations and standards MAY associate specific interpretations of rules based on the value of an `<xccdf:ident>` element and these additional attributes are allowed in order

to refine those interpretations. These additional attributes MUST NOT alter the processing of a benchmark document as described in Section 7, although they MAY be added to the output of Document Generation or displayed to users during processing.

6.4.4.4 <xccdf:check> and <xccdf:complex-check> Elements

Table 11 shows the possible properties of the *<xccdf:check>* element.

Table 11: <xccdf:check> Element Properties

Property	Type	Count	Description
check-import (element)	*special*	0-n	Identifies a value to be retrieved from the checking system during testing of a target system. This element MUST be empty. After the associated check results have been collected, the result structure returned by the checking engine is processed to collect the named information. This information is then recorded in the *<xccdf:check-import>* element in the corresponding *<xccdf:rule-result>*. This could be a single value or structured XML. In the latter case, the *@import-xpath* attribute can be used to traverse the structured XML and identify specific information of interest to record in the result.
check-export (element)	*special*	0-n	A mapping from an *<xccdf:Value>* element to a checking system variable (i.e., external name or id for use by the checking system). This supports export of tailoring values from the XCCDF processing environment to the checking system. The *@value-id* attribute of the *<xccdf:check-export>* element MUST match the *@id* attribute of an *<xccdf:Value>* element in the benchmark. The interface between the XCCDF benchmark consumer and the checking system SHOULD support, at a minimum, passing the following properties of the *<xccdf:Value>* element: *<xccdf:value>*, *@type*, and *@operator*.
check-content-ref (element)	*special*	0-n	Points to code for a detached check in another location that uses the language or system specified by the *<xccdf:check>* element's *@system* attribute. The *@href* attribute identifies the code location, and the OPTIONAL *@name* attribute MUST refer to a particular part, element, or component of the code. The default behavior of an *<xccdf:check-content-ref>* element that does not have a *@name* attribute SHALL be to execute all checks in the referenced code and AND their results together into a single *<xccdf:rule-result>*. However, if the *@multi-check* attribute (below) is set to true and a nameless *<xccdf:check-content-ref>* is used, each check in the targeted code is reported as a separate *<xccdf:rule-result>*. If multiple *<xccdf:check-content-ref>* elements appear, they represent alternative locations from which a benchmark consumer may obtain the check content. Benchmark consumers SHOULD process the alternatives in the order in which they appear in the XML. The first *<xccdf:check-content-ref>* from which content can be successfully retrieved SHOULD be used. (Note that ensuring the validity of this content is not the responsibility of a benchmark consumer.) If both *<xccdf:check-content-ref>* and *<xccdf:check-content>* elements appear in a single *<xccdf:check>* element, benchmark consumers SHOULD use the *<xccdf:check-content>* element only if none of the references can be resolved to provide content.

Property	Type	Count	Description
check-content (element)	*special*	0-1	Holds the actual code of a check, in the language or system specified by the *<xccdf:check>* element's *@system* attribute. The body of this element MAY be any XML, but SHALL NOT contain any XCCDF elements. It is OPTIONAL for benchmark consumers to process this element; typically it will be passed to a checking system or engine. If both *<xccdf:check-content-ref>* and *<xccdf:check-content>* elements appear in a single *<xccdf:check>* element, benchmark consumers SHOULD use the *<xccdf:check-content>* element only if none of the references can be resolved to provide content.
system (attribute)	URI	1	The URI for a checking system. The URI SHALL be compliant with the appropriate checking system specification (e.g., OVAL, OCIL). If the checking system uses XML namespaces, then the *@system* attribute for the system SHOULD be its namespace.
negate (attribute)	boolean	0-1	If set to true, the final result of the check is negated according to the truth table in Table 14. (default: false)
id (attribute)	identifier	0-1	Unique identifier for this element.
selector (attribute)	string	0-1	This MAY be referenced from a profile or used during manual tailoring to refine the application of the rule. If no selectors are specified for a given *<xccdf:rule>*, all *<xccdf:check>* elements with non-empty *@selector* attributes SHALL be ignored. If a rule has multiple *<xccdf:check>* elements with the same *@selector* attribute, each MUST employ a different checking system, as identified by the *@system* attribute of the *<xccdf:check>* element.
multi-check (attribute)	boolean	0-1	Applicable in cases where multiple checks are executed to determine compliance with a single rule. This situation can arise when a check includes an *<xccdf:check-content-ref>* element that does not include a *@name* attribute. The default behavior of a nameless *<xccdf:check-content-ref>* SHALL be to execute all checks in the referenced check content location and AND their results together into a single *<xccdf:rule-result>* using the AND truth table below (Table 12). This corresponds to a *@multi-check* attribute value of "false". If, however, the *@multi-check* attribute is set to "true" and a nameless *<xccdf:check-content-ref>* is used, the rule produces a separate *<xccdf:rule-result>* for each check in the check content location. (default: false) See Section 7.2.3.5.2.
xml:base (attribute)	*special*	0-1	The context for all relative URIs within the check.

In place of an *<xccdf:check>* element, XCCDF allows an *<xccdf:complex-check>* element. An *<xccdf:complex-check>* is a boolean logical expression whose individual terms are *<xccdf:check>* and/or *<xccdf:complex-check>* elements. This allows benchmark authors to create more sophisticated checks and to mix checks written with different checking systems. An *<xccdf:Rule>* MAY have at most one *<xccdf:complex-check>* element; on inheritance, the extending rule's *<xccdf:complex-check>* SHALL replace the extended rule's *<xccdf:complex-check>*. A benchmark consumer processing a benchmark picks at most one *<xccdf:check>* or *<xccdf:complex-check>* element to process for each rule. If both *<xccdf:check>* elements and an *<xccdf:complex-check>* element appear in a rule, then the *<xccdf:check>* elements will be ignored. See Section 7.2.3.5 for additional information on check processing.

When an *<xccdf:check>* element is used with an *<xccdf:complex-check>* element, the *<xccdf:check>* element's *@multi-check* attribute MUST be ignored.

Operators that MAY be used to combine the constituents of an *<xccdf:complex-check>* are AND and OR. Truth tables that MUST be used when evaluating these operators appear in Table 12 (AND) and Table 13 (OR). Each *<xccdf:complex-check>* MAY also specify that the expression should be negated (NOT); see Table 14 for the corresponding truth table. All of the abbreviations in the truth tables come from the description of the *<xccdf:rule-result>* element (see Table 26 in Section 6.6.4.2 for definitions of each abbreviation).

With an "AND" operator, the *<xccdf:complex-check>* evaluates to Pass only if all of its enclosed terms (*<xccdf:check>* and *<xccdf:complex-check>* elements) evaluate to Pass. Table 12 shows the truth table for "AND".

Table 12: Truth Table for AND

AND	P	F	U	E	N	K	S	I
P	P	F	U	E	P	P	P	P
F	F	F	F	F	F	F	F	F
U	U	F	U	U	U	U	U	U
E	E	F	U	E	E	E	E	E
N	P	F	U	E	N	N	N	N
K	P	F	U	E	N	K	K	K
S	P	F	U	E	N	K	S	S
I	P	F	U	E	N	K	S	I

The "OR" operator evaluates to Pass if any of its enclosed terms evaluate to Pass. The truth table for "OR" is shown in Table 13.

Table 13: Truth Table for OR

OR	P	F	U	E	N	K	S	I
P	P	P	P	P	P	P	P	P
F	P	F	U	E	F	F	F	F
U	P	U	U	U	U	U	U	U
E	P	E	U	E	E	E	E	E
N	P	F	U	E	N	N	N	N
K	P	F	U	E	N	K	K	K
S	P	F	U	E	N	K	S	S
I	P	F	U	E	N	K	S	I

If the *@negate* attribute is set to true, then the result of the *<xccdf:complex-check>* is complemented (inverted). The full truth table for negation is listed in Table 14.

Table 14: Truth Table for Negation

	P	F	U	E	N	K	S	I
not	F	P	U	E	N	K	S	I

The example below shows an *<xccdf:complex-check>* with several components.

```
<xccdf:complex-check operator="OR">
  <xccdf:check system="http://oval.mitre.org/XMLSchema/oval-definitions-5">
     <xccdf:check-content-ref href="xpDefs.xml"
         name="oval:org.example:def:456"/>
  </xccdf:check>
  <xccdf:complex-check operator="AND" negate="1">
     <xccdf:check system="http://oval.mitre.org/XMLSchema/oval-definitions-5">
         <xccdf:check-content-ref href="xpDefs.xml"
             name="oval:org.example:def:789"/>
     </xccdf:check>
     <xccdf:check system="http://scap.nist.gov/schema/ocil/2.0">
         <xccdf:check-content-ref href="xpInter.xml"
             name="ocil:org.example:questionnaire:6"/>
     </xccdf:check>
   </xccdf:complex-check>
</xccdf:complex-check>
```

6.4.4.5 <xccdf:fixtext> and <xccdf:fix> Elements

The *<xccdf:fixtext>* and *<xccdf:fix>* elements help tools support sophisticated facilities for automated and interactive remediation of benchmark findings. Table 15 lists the possible properties of an *<xccdf:fixtext>* element.

Table 15: Possible Properties for <xccdf:fixtext> Element

Property	Type	Count	Description
sub (element)	identifier	0-n	Specifies an *<xccdf:Value>* or *<xccdf:plain-text>* substitution. See Section 6.2.9.
xml:lang (attribute)	*special*	0-1	The language for the element; see Section 6.2.10.
override (attribute)	boolean	0-1	Specifies inheritance behavior (see Section 6.3.1). (default: false)
fixref (attribute)	identifier	0-1	A reference to a specific *<xccdf:fix>* *@id* attribute. This allows pairing explanatory text with specific fix procedures.
reboot (attribute)	boolean	0-1	Whether or not remediation will require a reboot or hard reset of the target. When specified, it SHALL have one of two values: true (1) or false (0) (default: 0).
strategy (attribute)	string	0-1	The method or approach for fixing the problem. When specified, the attribute SHALL have one of the following values: • unknown (strategy not defined) (default) • configure (adjust target configuration/settings) • combination (combination of two or more approaches) • disable (turn off or uninstall a target component) • enable (turn on or install a target component) • patch (apply a patch, hotfix, update, etc.) • policy (remediation requires out-of-band adjustments to policies or procedures) • restrict (adjust permissions, access rights, filters, or other access restrictions) • update (install upgrade or update the system)

Property	Type	Count	Description
disruption (attribute)	string	0-1	An estimate of the potential for disruption or operational degradation that the application of this fix will impose on the target. When specified, the attribute SHALL have one of the following values: • unknown (disruption not defined) (default) • low (little or no disruption expected) • medium (potential for minor or short-lived disruption) • high (potential for serious disruption)
complexity (attribute)	string	0-1	The estimated complexity or difficulty of applying the fix to the target. When specified, the attribute SHALL have one of the following values: • unknown (complexity not defined) (default) • low (fix is very simple to apply) • medium (fix is moderately difficult or complex) • high (fix is very complex to apply)

Table 16 lists the possible properties of an *<xccdf:fix>* element.

Table 16: Possible Properties for <xccdf:fix> Element

Property	Type	Count	Description
sub (element)	identifier	0-n	Specifies an *<xccdf:Value>* or *<xccdf:plain-text>* substitution. See Section 6.2.9.
instance (element)	string		Designates a spot where the name of the instance SHOULD be substituted into the fix template to generate the final fix data. If the *@context* attribute is omitted, the value of the context SHALL default to "undefined".
id (attribute)	identifier	0-1	A local identifier for the element. It is OPTIONAL for the id to be unique; multiple *<xccdf:fix>* elements MAY have the same id but different values for their other attributes.
reboot (attribute)	boolean	0-1	Whether or not remediation will require a reboot or hard reset of the target. Permitted values: true (1) and false (0) (default: 0).
strategy (attribute)	string	0-1	The method or approach for fixing the problem. See Table 15 for the list of possible values.
disruption (attribute)	string	0-1	An estimate of the potential for disruption or operational degradation that the application of this fix will impose on the target. See Table 15 for the list of possible values.
complexity (attribute)	string	0-1	The estimated complexity or difficulty of applying the fix to the target. See Table 15 for the list of possible values.
system (attribute)	URI	0-1	A URI that identifies the scheme, language, engine, or process for which the fix contents are written. Table 17 defines several general-purpose URNs that MAY be used for this, and tool vendors and system providers MAY define and use target-specific URNs.
platform (attribute)	URI	0-1	In case different fix scripts or procedures are required for different target platform types (e.g., different patches for Windows Vista and Windows 7), this attribute allows a CPE name or CPE applicability language expression to be associated with an *<xccdf:fix>* element. This SHOULD appear on an *<xccdf:fix>* when the content applies to only one platform out of several to which the rule could apply.

Table 17 lists predefined values for the *@system* attribute of an *<xccdf:fix>* element.

Table 17: Predefined Values for @system Attribute of <xccdf:fix> Element

URI	Content of the <xccdf:fix> Element
urn:xccdf:fix:commands	A list of target system commands; executed in order, the commands should bring the target system into compliance with the rule.
urn:xccdf:fix:urls	A list of one or more URLs. The resources identified by the URLs should be applied to bring the system into compliance with the rule.
urn:xccdf:fix:script:*language*	A script written in the given *language*. Executing the script should bring the target system into compliance with the rule. The following languages are pre-defined: sh – Bourne shell csh – C Shell perl – Perl batch – Windows batch script python – Python and all Python-based scripting languages vbscript – Visual Basic Script (VBS) javascript – Javascript (ECMAScript, Jscript) tcl – Tcl and all Tcl-based scripting languages
urn:xccdf:fix:patch:*vendor*	A patch identifier, in proprietary format as defined by the vendor. The vendor string should be the DNS domain name of the vendor. For example, for Microsoft Corporation, the DNS domain is “Microsoft.com”.

6.4.5 <xccdf:Value> Element

6.4.5.1 Basics

An *<xccdf:Value>* is a named parameter (with a unique *@id* attribute) that can be substituted into properties of other elements within the benchmark, including the interior of structured check specifications and fix scripts. An *<xccdf:Value>* can hold string, boolean, or numeric content, or lists thereof. *<xccdf:Value>* elements can include information about permissible values, display defaults, and restrictions on tailoring for the Value. For example, an *<xccdf:Value>* might be used to hold a benchmark’s lower limit for password length on some OS. In a profile for that OS to be used in a closed lab, the default value might be 8, but in a profile for that OS to be used on the Internet, the default value might be 12.

The example below shows an *<xccdf:Value>* element.

```
<xccdf:Value id="xccdf_org.example_value_web-server-port" type="number"
  operator="equals">
  <xccdf:title>Web Server Port</xccdf:title>
  <xccdf:description>TCP port on which the server listens
  </xccdf:description>
  <xccdf:value>12080</xccdf:value>
  <xccdf:default>80</xccdf:default>
  <xccdf:lower-bound>0</xccdf:lower-bound>
  <xccdf:upper-bound>65535</xccdf:upper-bound>
</xccdf:Value>
```

<xccdf:Value> elements MAY encapsulate values that are lists (possibly zero-length) of simple types. These structures are supported by the *<xccdf:complex-value>* element, the *<xccdf:complex-default>* element, and within the *<xccdf:choices>* element, the

<xccdf:complex-choice> element. If any of these elements has no child elements, it represents an empty list.

A single *<xccdf:Value>* MAY use a mixture of both simple (i.e., a single number, string, or boolean value) and complex types. Apart from its ability to encapsulate different types of information, a complex field is used in the same way as its simple counterpart.

6.4.5.2 Properties

Table 18 describes the *<xccdf:Value>* element's properties (in addition to the core item properties previously described in Table 5).

Table 18: <xccdf:Value> Element Properties

Property	Type	Count	Description
value (element)	string	1-n	The current value of this *<xccdf:Value>*. This element MAY have a *@selector* attribute (see Section 6.4.5.5).
complex-value (element)	*special*		Similar to *<xccdf:value>* except that this element supports values that are lists of simple types. This element MAY have a *@selector* attribute (see Section 6.4.5.5).
default (element)	string	0-n	The default value displayed to the user as a suggestion by benchmark producers during tailoring of this *<xccdf:Value>* element. (This is not the default value of an *<xccdf:Value>*.) This element MAY have a *@selector* attribute (see Section 6.4.5.5).
complex-default (element)	*special*		Similar to *<xccdf:default>* except that this element supports default values that are lists of simple types. This element MAY have a *@selector* attribute (see Section 6.4.5.5).
match (element)	string	0-n	A Perl Compatible Regular Expression (PCRE) (see [PCRE] and [UNICODE]) that a benchmark producer MAY apply during tailoring to validate a user's input for the Value. It SHALL use implicit anchoring. It SHALL apply only when the *@type* is "string" or "number" or a list of strings and/or numbers. For example, if the *@type* was "string", but the value was meant to be a Cisco IOS router interface name, then the *<xccdf:match>* element might be set to "[A-Za-z]+ *[0-9]+(/[0-9.]+)*". This would allow a tailoring tool to reject an invalid user input like "f8xq+" but accept a legal one like "Ethernet1/3". This element MAY have a *@selector* attribute (see Section 6.4.5.5).
lower-bound (element)	decimal	0-n	Minimum legal value for this Value. It is used to constrain value input during tailoring, when the *@type* is "number". Values supplied by the user for tailoring the benchmark MUST be equal to or greater than this number. This element MAY have a *@selector* attribute (see Section 6.4.5.5).
upper-bound (element)	decimal	0-n	Maximum legal value for this Value. It is used to constrain value input during tailoring, when the *@type* is "number". Values supplied by the user for tailoring the benchmark MUST be less than or equal to this number. This element MAY have a *@selector* attribute (see Section 6.4.5.5).
choices (element)	*special*	0-n	A list of legal or suggested choices (values) for an *<xccdf:Value>* element, to be used during tailoring and document generation. See Section 6.4.5.3.

Property	Type	Count	Description
source (element)	URI	0-n	URI indicating where the tool may acquire values, value bounds, or value choices for this *<xccdf:Value>* element. XCCDF does not attach any meaning to the URI; it may be an arbitrary community or tool-specific value, or a pointer directly to a resource. If several values for the *<xccdf:source>* element appear, then they SHALL represent alternative means or locations for obtaining the value in descending order of preference (i.e., most preferred first).
signature (element)	*special*	0-1	A digital signature asserting authorship and allowing verification of the integrity of the Value. See Section 6.2.7.
id (attribute)	*special*	1	Unique element identifier. See Section 6.2.3.
type (attribute)	string	0-1	The data type of the Value: "string", "number", or "boolean" (default: "string"). A tool may choose any convenient form to store a Value's *<xccdf:value>* element, but the *@type* attribute conveys how the Value SHOULD be treated for user input validation purposes during tailoring processing. The *@type* attribute MAY also be used to give additional guidance to the user or to validate the user's input. For example, if an *<xccdf:value>* element's *@type* attribute is "number", then a tool might choose to reject user tailoring input that is not composed of digits. In the case of a list of values, the *@type* attribute, if present, SHALL be applied to all elements of the list individually.
operator (attribute)	string	0-1	The operator to be used for comparing this Value to some part of the test system's configuration during rule checking (default: "equals"). See Section 6.4.5.4.
interactive (attribute)	boolean	0-1	Whether tailoring for this Value should also be performed during benchmark application (default: false). The benchmark consumer MAY ignore the attribute if asking the user is not feasible or not supported.
interfaceHint (attribute)	string	0-1	A hint or recommendation to a benchmark consumer or producer about how the user might select or adjust the Value. If used, this element SHALL have one of the following interface pattern values: • "choice" (multiple choice) • "textline" (multiple lines of text) • "text" (single line of text) • "date" (date) • "datetime" (date and time)

6.4.5.3 <xccdf:choices> Element

The *<xccdf:choices>* element SHOULD be used when there are a moderate number of known values that are most appropriate. For example, if the Value were the authentication mode for a server, the choices might be "password" and "pki". Table 19 lists the possible properties for an *<xccdf:choices>* element. If a product presents the choice values from an *<xccdf:choices>* element to a user, they SHOULD be presented in the order in which they appear.

Table 19: Possible Properties for <xccdf:choices> Element

Property	Type	Count	Description
choice (element)	string	1-n	The text of a possible choice.
complex-choice (element)	*special*		A possible choice consisting of a list of values. If this element has no *<xccdf:item>* children, it SHALL represent an empty list.
mustMatch (attribute)	boolean	0-1	If this is true, the list SHALL represent all the legal values. If this is false or absent, the list SHALL represent suggested values, and other values MAY also be legal (subject to the parent *<xccdf:Value>* element's *@upper-bound*, *@lower-bound*, and *@match* attributes).

Property	Type	Count	Description
selector (attribute)	string	0-1	Used for tailoring via a profile. See Section 6.4.5.5.

6.4.5.4 @operator Attribute

When defining an *<xccdf:Value>* element, the benchmark author MAY specify the operator to be used for comparing the Value to a configuration during rule checking. For example, one part of an OS benchmark might be verifying that the configuration included a minimum password length; the *<xccdf:Value>* element that holds the tailorable minimum could have *@type* "number" and *@operator* "greater than". Exactly how *<xccdf:Value>* elements are used in rules depends on the capabilities of the checking system. Benchmark consumers MAY ignore the *@operator* attribute; therefore, authors SHOULD include sufficient information in the *<xccdf:description>* and *<xccdf:question>* elements to make the role of the Value clear. Table 20 describes the *@operator* values permitted for each *@type* value.

Table 20: Permitted Operators by Value Type

Value Type	Available Operators
number	equals, not equal, less than, greater than, less than or equal, greater than or equal (default: equals)
boolean	equals, not equal (default: equals)
string	equals, not equal, pattern match (pattern match means regular expression match; SHOULD comply with [UNICODE]) (default: equals)
lists	as component data type

6.4.5.5 @selector Attribute

As detailed in Table 18, an *<xccdf:Value>* MAY include various elements that constrain or limit the values that the Value may be given. Authors MAY use these elements to assist users in tailoring the benchmark. These elements all support the *@selector* attribute. If there are multiple instances of one of these elements within a single *<xccdf:Value>* element, no more than one of the instances MAY omit the *@selector* attribute, and every other instance MUST have a different value specified for its *@selector* attribute. For elements that may be substituted for each other—specifically, *<xccdf:value>* and *<xccdf:complex-value>* elements, and *<xccdf:default>* and *<xccdf:complex-default>* elements—no more than one instance of either element MAY omit the *@selector* attribute, and every other instance of both elements MUST have a different value specified for its *@selector* attribute. For more information about selector types, see Section 6.5.3.

In the absence of any profile or tailoring actions, the default *<xccdf:value>* or *<xccdf:complex-value>* element in an *<xccdf:Value>* SHALL be the one with an empty or absent *@selector*. If there is no *<xccdf:value>* or *<xccdf:complex-value>* element with an empty or absent *@selector*, the first *<xccdf:value>* or *<xccdf:complex-value>* element in top-down processing of the XML SHALL be the default element. For all other selectable *<xccdf:Value>* elements (i.e., *<xccdf:default>*, *<xccdf:complex-default>*, *<xccdf:match>*, *<xccdf:upper-bound>*, *<xccdf:lower-bound>*, and *<xccdf:choices>*), the default activity SHALL be to ignore all elements without an empty or absent *@selector*.

6.5 <xccdf:Profile> Element

6.5.1 Basics

An *<xccdf:Profile>* element is a named tailoring for a benchmark. While a benchmark can be tailored in place by setting properties of various elements, *<xccdf:Profile>* elements allow one benchmark document to hold several independent tailorings. The *<xccdf:select>* element children of the *<xccdf:Profile>* affect which *<xccdf:Group>* and *<xccdf:Rule>* elements are selected for processing when the *<xccdf:Profile>* is in effect. The *<xccdf:refine-rule>* element allows modification of properties in *<xccdf:Group>* and *<xccdf:Rule>* elements, while the *<xccdf:refine-value>* element allows modification of *<xccdf:Value>* properties, including selection of the effective value. Finally, *<xccdf:set-value>* and *<xccdf:set-complex-value>* allow an *<xccdf:Value>* element's value to be set directly to a simple or complex setting, respectively. The example below shows a simple *<xccdf:Profile>*.

```
 <xccdf:Profile id="xccdf_org.example_profile_strict" prohibitChanges="1"
          extends="xccdf_org.example_profile_lenient" note-tag="strict-tag">
   <xccdf:title>Strict Security Settings</xccdf:title>
   <xccdf:description>
     Strict lockdown rules and values, for hosts deployed to high-risk environments.
   </xccdf:description>
   <xccdf:set-value idref="xccdf_org.example_value_password-len">10</xccdf:set-value>
   <xccdf:refine-value idref="xccdf_org.example_value_session-timeout"
          selector="quick"/>
   <xccdf:refine-rule idref="xccdf_org.example_value_session-auth-rule"
          selector="harsh"/>
   <xccdf:select idref="xccdf_org.example_value_password-len-rule" selected="1"/>
   <xccdf:select idref="xccdf_org.example_value_audit-cluster" selected="1"/>
   <xccdf:select idref="xccdf_org.example_value_telnet-disabled-rule" selected="1"/>
   <xccdf:select idref="xccdf_org.example_value_telnet-settings-cluster"
          selected="0"/>
</xccdf:Profile>
```

An *<xccdf:Profile>* MAY extend another *<xccdf:Profile>* in the same benchmark by using the *@extends* attribute. An *<xccdf:Profile>* in an *<xccdf:Tailoring>* element MAY extend any *<xccdf:Profile>* in a benchmark, but *<xccdf:Profile>* elements in an *<xccdf:Tailoring>* element SHALL NOT be extended.

Certain properties of the extended *<xccdf:Profile>* appear before the corresponding properties in the extending *<xccdf:Profile>*. Since *<xccdf:Profile>* properties are processed in the order in which they appear in the XML, this means that properties of the extending *<xccdf:Profile>* can override corresponding properties inherited from the extended *<xccdf:Profile>*. See Sections 7.2.2 and 7.2.3.4 for additional information on extension and inheritance.

6.5.2 Properties

Table 21 describes the *<xccdf:Profile>* element's properties.

Table 21: <xccdf:Profile> Element Properties

Property	Type	Count	Description
status (element)	*special*	0-n	Status of this *<xccdf:Profile>*and date at which it attained that status. Authors MAY use this element to record the maturity or consensus level of a profile. If the status is not given explicitly, then the *<xccdf:Profile>* is taken to have the same status as its parent *<xccdf:Benchmark>*. See Section 6.2.8.
dc-status (element)	*special*	0-n	Holds additional status information using the Dublin Core format. See Section 6.2.8.
version (element)	string	0-1	Version number of this *<xccdf:Profile>*, with an OPTIONAL *@time* timestamp attribute (when the version was completed) and an OPTIONAL *@update* URI attribute (where updates may be obtained).
title (element)	string	1-n	Title of this *<xccdf:Profile>*. It MAY have one or more *<xccdf:sub>* elements (see Section 6.2.9), an *@xml:lang* attribute (see Section 6.2.10), and/or an *@override* attribute (see Section 6.3.1).
description (element)	HTML-enabled text	0-n	Text that describes this *<xccdf:Profile>*. It MAY have one or more *<xccdf:sub>* elements (see Section 6.2.9), an *@override* attribute (see Section 6.3.1), and/or an *@xml:lang* attribute (see Section 6.2.10).
reference (element)	*special*	0-n	A reference where the user can learn more about the subject of this profile. See Section 6.2.6.
platform (element)	string	0-n	A target platform for this profile. See Section 6.2.5.
select, set-complex-value, set-value, refine-value, refine-rule (element)	*special*	0-n	References to Groups, Rules, and Values for customization and tailoring. See Section 6.5.3.
metadata (element)	*special*	0-n	Metadata associated with this *<xccdf:Profile>*. See Section 6.2.4.
signature (element)	*special*	0-1	A digital signature asserting authorship and allowing verification of the integrity of the *<xccdf:Profile>*. See Section 6.2.7.
id (attribute)	*special*	1	Unique identifier for this *<xccdf:Profile>*. See Section 6.2.3.
prohibitChanges (attribute)	boolean	0-1	Whether or not products should prohibit changes to this *<xccdf:Profile>* (default: false).
abstract (attribute)	boolean	0-1	If true, then this *<xccdf:Profile>* exists solely to be extended by other *<xccdf:Profile>* elements (default: false). See Sections 6.3.1 and 7.2.2.
note-tag (attribute)	identifier	0-1	Tag identifier to specify which *<xccdf:profile-note>* element from an *<xccdf:Rule>* should be associated with this *<xccdf:Profile>.*
extends (attribute)	identifier	0-1	The id of an *<xccdf:Profile>* on which to base this *<xccdf:Profile>.*
xml:base (attribute)	*special*	0-1	The context for all relative URIs within the profile.
Id (attribute)	*special*	0-1	An identifier used for referencing elements included in an XML signature. See Section 6.2.7.

6.5.3 Selectors

Each *<xccdf:Profile>* can contain one or more selectors to express a particular customization or tailoring of the *<xccdf:Benchmark>*. Table 22 defines the five kinds of selectors.

Table 22: Selectors

Property	Type	Count	Description
select (element)	*special*	0-n	Designates a Rule, Group, or cluster of Rules and Groups. It overrides the *@selected* attribute on the designated items, providing a means for including or excluding rules from an assessment. The *<xccdf:select>* element MAY have one or more *<xccdf:remark>* elements (strings) containing explanatory material or other prose. Each instance of *<xccdf:remark>* MAY have an *@xml:lang* attribute (see Section 6.2.10).The *<xccdf:select>* element's *@idref* attribute (identifier) SHALL specify the designated Group, Rule, or cluster of Rules and Groups; it MUST match the *@id* attribute of a Group or Rule in the Benchmark, or the cluster id assigned to one or more Rules or Groups. If the *<xccdf:select>* element's REQUIRED *@selected* attribute (boolean) is set to true, the Rule, Group, or Rules and Groups in the cluster SHALL have their *@selected* attribute set to true, otherwise they SHALL have their *@selected* attribute set to false. Subsequent tailoring actions may further modify these properties.
set-value (element)	string	0-n	Points to an *<xccdf:Value>* element and overrides its *<xccdf:value>* and *<xccdf:complex-value>* element(s) without changing any other properties. It provides a means for directly specifying the value of a variable to be used in benchmark processing. This selector MAY also be applied to the *<xccdf:Value>* elements in a cluster by specifying the cluster id in the *@idref* attribute, in which case it SHALL override the *<xccdf:value>* elements of all of them.
set-complex-value (element)	*special*	0-n	Supports the direct specification of complex value types such as lists. Zero or more item elements MAY appear as children of this element; if no child elements are present, this element SHALL represent an empty list. This overrides the *<xccdf:value>* and *<xccdf:complex-value>* element(s) of an *<xccdf:Value>* element. Like *<xccdf:set-value>*, *<xccdf:set-complex-value>* MAY also be applied to *<xccdf:Value>* elements in a cluster.
refine-value (element)	*special*	0-n	Designates the Value constraints to be applied during tailoring, for an *<xccdf:Value>* element or the *<xccdf:Value>* members of a cluster. The element MAY have one or more *<xccdf:remark>* elements containing explanatory material or other prose. Each instance of *<xccdf:remark>* MAY have an *@xml:lang* attribute (see Section 6.2.10). Possible attributes for the *<xccdf:refine-value>* element are the id (REQUIRED) of a Value or item cluster, the id of a Value selector, and a new setting for the Value operator. The element provides a means for authors to impose different constraints on tailoring for different profiles. (Constraints MUST be designated with a selector id. For example, a particular numeric Value might have several different sets of *<xccdf:value>*, *<xccdf:upper-bound>*, and *<xccdf:lower-bound>* elements, designated with different selector ids. The *<xccdf:refine-value>* selector tells benchmark consumers which value to employ and bounds to enforce when that particular profile is in effect.

Property	Type	Count	Description
refine-rule (element)	*special*	0-n	Allows the author to select check statements and override the *@weight*, *@severity*, and *@role* of a Rule, Group, or cluster of Rules and Groups. The element MAY have one or more *<xccdf:remark>* elements containing explanatory material or other prose. Each instance of *<xccdf:remark>* MAY have an *@xml:lang* attribute (see Section 6.2.10). The *<xccdf:refine-rule>* element has an *@idref* attribute (NCName) that SHALL refer to a Rule, Group, or cluster. Despite the name, this selector does apply for Groups and for clusters that include Groups, but it SHALL only affect their *@weight* attribute. If the selector specified in an *<xccdf:refine-rule>* element does not match any of the selectors specified on any of the check children of a Rule, then the *<xccdf:check>* child element without a *@selector* attribute MUST be used.

The example below illustrates how selectors work with the *<xccdf:refine-value>* element. See Section 6.4.5.5 for more information on the *@selector* attribute. In the example, before the *<xccdf:refine-value>* is applied, the effective value is 8 and the effective lower bound is 8. (These are the two properties that have no selectors and are therefore the default.) After the *<xccdf:refine-value>* is applied, the effective value is 14 and the effective lower bound is 12.

```
<xccdf:Value id="xccdf_org.example_value_pw-length" type="number"
   operator="equals">
   <xccdf:title>Minimum password length policy</xccdf:title>
   <xccdf:value>8</xccdf:value>
   <xccdf:value selector="high">14</xccdf:value>
   <xccdf:lower-bound>8</xccdf:lower-bound>
   <xccdf:lower-bound selector="high">12</xccdf:lower-bound>
</xccdf:Value>
<xccdf:Profile id="xccdf_org.example_profile_enterprise-internet">
   <xccdf:title>Enterprise internet server profile</xccdf:title>
   <xccdf:refine-value idref="xccdf_org.example_value_pw-length" selector="high"/>
</xccdf:Profile>
```

6.6 <xccdf:TestResult> Element

6.6.1 Basics

An *<xccdf:TestResult>* element encapsulates the results of a single application of a benchmark to a single target platform. The *<xccdf:TestResult>* element normally appears as the child of the *<xccdf:Benchmark>* element, although it may also appear as the top-level element of an XCCDF results document. XCCDF is not intended to be a database format for detailed results; the *<xccdf:TestResult>* element offers a way to store the results of individual tests in modest detail, with the ability to reference lower-level testing data.

Benchmark consumers SHOULD include mechanisms so that *<xccdf:TestResult>* elements can retain their context even if separated from their source benchmark since the *<xccdf:TestResult>* only has meaning relative to the Rules that produced it. There are several ways to preserve this context, including making sure that the *<xccdf:benchmark>* element in the *<xccdf:TestResult>* is a persistent link to the specific document and version of the benchmark that produced the result, filling in the relevant version, organization, and similar fields, and/or using descriptive metadata. This document

does not mandate any specific approach, but benchmark consumers SHOULD ensure some mechanism is in place to avoid context loss.

The example below shows an *<xccdf:TestResult>* element with a few *<xccdf:rule-result>* children.

```
<xccdf:TestResult id="xccdf_org.example_testresult_ios-test5"
        end-time="2007-09-25T7:45:02-04:00"
        xmlns:xccdf="http://checklists.nist.gov/xccdf/1.2" version="1.0">
   <xccdf:benchmark href="ios-sample-12.4.xccdf.xml"
        id="xccdf_org.example_benchmark_ios-test-benchmark"/>
   <xccdf:title>Sample Results Block</xccdf:title>
   <xccdf:remark>Test run by Bob on Sept 25, 2007</xccdf:remark>
   <xccdf:organization>Department of Commerce</xccdf:organization>
   <xccdf:organization>National Institute of Standards and Technology
   </xccdf:organization>
   <xccdf:identity authenticated="1" privileged="1">admin_bob</xccdf:identity>
   <xccdf:target>lower.test.net</xccdf:target>
   <xccdf:target-address>192.168.248.1</xccdf:target-address>
   <xccdf:target-address>2001:8::1</xccdf:target-address>
   <xccdf:target-facts>
        <xccdf:fact type="string" name="urn:xccdf:fact:ethernet:MAC">
            02:50:e6:c0:14:39
        </xccdf:fact>
        <xccdf:fact type="string" name="urn:xccdf:fact:ethernet:MAC">
            02:50:e6:1f:33:b0
        </xccdf:fact>
   </xccdf:target-facts>
   <xccdf:set-value
        idref="xccdf_org.example_value_exec-timeout-time">10
   </xccdf:set-value>
   <xccdf:rule-result idref="xccdf_org.example_rule_ios12-no-finger-service"
        time="2007-09-25T13:45:00-04:00">
        <xccdf:result>pass</xccdf:result>
   </xccdf:rule-result>
   <xccdf:rule-result idref="xccdf_org.example_rule_req-exec-timeout"
        time="2007-09-25T13:45:06-04:00">
        <xccdf:result>fail</xccdf:result>
        <xccdf:instance>console</xccdf:instance>
        <xccdf:fix system="urn:xccdf:fix:commands" reboot="0" disruption="low">
             line console
             exec-timeout 10 0
        </xccdf:fix>
   </xccdf:rule-result>
   <xccdf:score>67.5</xccdf:score>
   <xccdf:score system="urn:xccdf:scoring:absolute">0</xccdf:score>
</xccdf:TestResult>
```

6.6.2 Properties

Table 23 describes the *<xccdf:TestResult>* element's properties. Although several of its child elements technically support the *@override* attribute (discussed in Section 6.3.1), the *<xccdf:TestResult>* element cannot be extended. Therefore, *@override* has no meaning within an *<xccdf:TestResult>* element and its children, and it SHOULD NOT be used for them.

Table 23: <xccdf:TestResult> Element Properties

Property	Type	Count	Description
benchmark (element)	*special*	0-1	Reference to the *<xccdf:Benchmark>* for which the *<xccdf:TestResult>* records results. REQUIRED if this *<xccdf:TestResult>* element is the top-level element, OPTIONAL otherwise. The REQUIRED *@href* attribute gives the URI of the benchmark document, and the OPTIONAL *@id* attribute holds the value of that *<xccdf:Benchmark>* element's *@id* attribute.
tailoring-file (element)	*special*	0-1	Holds identifying information about an *<xccdf:Tailoring>* element. See Section 6.6.5.
title (element)	string	0-n	Title of the test. It MAY have an *@xml:lang* attribute (see Section 6.2.10).
remark (element)	string	0-n	A remark about the test, possibly supplied by the person administering the benchmark assessment. It MAY have an *@xml:lang* attribute (see Section 6.2.10).
organization (element)	string	0-n	The name of the organization or other entity responsible for applying this benchmark and generating this result. When multiple organization elements are used to indicate multiple organization names in a hierarchical organization, the highest-level organization SHOULD appear first (e.g., "U.S. Government") followed by subordinate organizations (e.g., "Department of Defense").
identity (element)	string	0-1	Information about the system identity or user employed during application of the benchmark. If used, SHALL specify the name of the authenticated identity. Has REQUIRED boolean attributes that specify whether the identity was authenticated with the target system during the application of the benchmark *(@authenticated)*, and whether the identity was granted administrative or other special privileges beyond those of a normal user *(@privileged)*.
profile (element)	identifier	0-1	The identifier of the *<xccdf:Profile>* used for the test. If the *<xccdf:TestResult>* element contains an *<xccdf:tailoring-file>* element, then this SHALL be the identifier of the associated tailoring profile. Otherwise, if there is no *<xccdf:tailoring-file>* element present, this SHALL be the identifier of the associated benchmark profile if a profile was applied, and it SHALL be absent if no profile was applied. See Section 6.6.5.
target (element)	string	1-n	Name or description of the target system whose test results are recorded in the *<xccdf:TestResult>* element (the system to which a benchmark test was applied). Each appearance of the element supplies a name by which the target host or device was identified at the time the test was run. The name MAY be any string, but applications SHOULD include the fully qualified DNS name whenever possible.
target-address (element)	string	0-n	Network address of the target system to which a benchmark test was applied. Typical forms for the address include IP version 4 (IPv4), IP version 6 (IPv6), and Ethernet media access control (MAC).
target-facts (element)	string	0-1	A list of named facts about the target system or platform. Each *<xccdf:fact>* in the list SHALL specify the value of the fact itself. Each *<xccdf:fact>* SHALL have a *@name* attribute (URI) and MAY include a *@type* attribute ("string", "number", or "boolean") (default: "boolean"). Pre-defined *@name* attribute values are listed in Table 24; product developers MAY define additional platform-specific and product-specific facts.

Property	Type	Count	Description
target-id-ref (element)	*special*	0-n	Used to reference target identification information located in an external document. The assessment target MAY be identified using other identification schemas such as Asset Identification (see [IR7693] for additional information). Each instance of this element identifies the same target of this report (multiple identifiers for a single target). Inclusion of an *<xccdf:target-id-ref>* element does not remove the requirement to identify the target using the *<xccdf:target>* element.
any			Arbitrary contents, namespace="##other". This is for other target identification structures, such as those defined in the Asset Identification schema (see [IR7693]). Inclusion of an identifying element does not remove the requirement to identify the target using the *<xccdf:target>* element.
platform (element)	string	0-n	The platforms that the target system was found to meet. See Section 6.2.5.
set-value (element)	string	0-n	Specific setting for a single *<xccdf:Value>* element used during the test.
set-complex-value (element)	*special*		Specific setting for a single *<xccdf:Value>* element used during the test when the given value is set to a complex type, such as a list.
rule-result (element)	*special*	0-n	The result of a single instance of a rule application against the target. The *<xccdf:TestResult>* MUST include one *<xccdf:rule-result>* record for each *<xccdf:Rule>* that was selected in the resolved benchmark; it MAY also include *<xccdf:rule-result>* records for *<xccdf:Rule>* elements that were unselected in the benchmark. See Section 6.6.4.
score (element)	decimal	1-n	An overall score for this benchmark test. The OPTIONAL *@system* attribute identifies the URI for a scoring model (see Section 7.3.2 for more information on pre-defined models). The OPTIONAL *@maximum* attribute specifies the maximum possible value of the score.
metadata (element)	*special*	0-n	XML metadata associated with this *<xccdf:TestResult>*. For example, this element can hold a copy of the *<xccdf:metadata>* element of the *<xccdf:Benchmark>* which served as the source for these results. This is especially useful if the *<xccdf:TestResult>* will be separated from its source benchmark because the publication and support information in the benchmark can travel with the *<xccdf:TestResult>*. Benchmark consumers MAY also add their own metadata to *<xccdf:TestResult>* elements they produce. See Section 6.2.3.
signature (element)	*special*	0-1	A digital signature asserting authorship and allowing verification of the integrity of the *<xccdf:TestResult>*. See Section 6.2.7.
id (attribute)	*special*	1	Unique identifier for this element; see Section 6.2.3.
start-time (attribute)	dateTime	0-1	Time when test began.
end-time (attribute)	dateTime	1	Time when test was completed and the results recorded.
test-system (attribute)	string	0-1	Name of the benchmark consumer program that generated this *<xccdf:TestResult>* element; SHOULD be either a CPE name or a CPE applicability language expression.
version (attribute)	string	0-1	The version number string copied from the *<xccdf:Benchmark>*.
Id (attribute)	*special*	0-1	An identifier used for referencing elements included in an XML signature. See Section 6.2.7.

6.6.3 <xccdf:fact> Element

The URIs listed in Table 24 SHOULD be used, when applicable to the target, to record facts about the IT asset to which an *<xccdf:TestResult>* applies.

Table 24: Predefined @name Attribute Values for <xccdf:fact> Elements

URI	Description
urn:xccdf:fact:asset:identifier:mac	Ethernet media access control address (SHOULD be sent as a pair with the IPv4 or IPv6 address to ensure uniqueness)
urn:xccdf:fact:asset:identifier:ipv4	IPv4 address
urn:xccdf:fact:asset:identifier:ipv6	IPv6 address
urn:xccdf:fact:asset:identifier:host_name	Host name of the asset, if assigned
urn:xccdf:fact:asset:identifier:fqdn	Fully qualified domain name
urn:xccdf:fact:asset:identifier:ein	Equipment identification number or other inventory tag number
urn:xccdf:fact:asset:identifier:pki:	X.509 PKI certificate for the asset (encoded in Base-64)
urn:xccdf:fact:asset:identifier:pki:thumbprint	SHA-1 hash of the PKI certification for the asset (encoded in Base-64)
urn:xccdf:fact:asset:identifier:guid	Globally unique identifier for the asset
urn:xccdf:fact:asset:identifier:ldap	LDAP directory string (distinguished name) of the asset, if assigned
urn:xccdf:fact:asset:identifier:active_directory	Active Directory realm to which the asset belongs, if assigned
urn:xccdf:fact:asset:identifier:nis_domain	NIS domain of the asset, if assigned
urn:xccdf:fact:asset:environmental_information:owning_organization	Organization that tracks the asset on its inventory
urn:xccdf:fact:asset:environmental_information:current_region	Geographic region where the asset is located
urn:xccdf:fact:asset:environmental_information:administration_unit	Name of the organization that does system administration for the asset
urn:xccdf:fact:asset:environmental_information:administration_poc:title	Title (e.g., Mr, Ms, Col) of the system administrator for an asset
urn:xccdf:fact:asset:environmental_information:administration_poc:e-mail	E-mail address of the system administrator for the asset
urn:xccdf:fact:asset:environmental_information:administration_poc:first_name	First name of the system administrator for the asset
urn:xccdf:fact:asset:environmental_information:administration_poc:last_name	Last name of the system administrator for the asset

6.6.4 <xccdf:rule-result> Element

6.6.4.1 Properties

The *<xccdf:rule-result>* element within the *<xccdf:TestResult>* element holds the result of applying an *<xccdf:Rule>* from the benchmark to a target system or component of a target system. Table 25 describes the *<xccdf:rule-result>* element's properties.

Table 25: <xccdf:rule-result> Element Properties

Property	Type	Count	Description
result (element)	string	1	Result of applying this *<xccdf:Rule>* to a target or target component: MUST be set to one of the values listed in Table 26.
override (element)	*special*	0-n	An XML block explaining how and why an auditor chose to override the result. See Section 6.6.4.3.
ident (element)	string	0-n	A long-term globally meaningful identifier for the issue, vulnerability, platform, etc. copied from the *<xccdf:Rule>*. SHALL have a *@system* attribute designating the URI of the organization or scheme that assigned the identifier. An *<xccdf:ident>* element MAY also have other attributes pulled from other namespaces in order to provide additional metadata for the given identifier. See Section 6.4.4.3 for more on this element.
metadata (element)	*special*	0-n	XML metadata associated with this *<xccdf:rule-result>*. For example, this element could hold a copy of the metadata of the *<xccdf:Rule>* that served as the source for these results. Benchmark consumers MAY also add their own metadata to the *<xccdf:rule-result>* elements they produce. See Section 6.2.4 for additional information.
message (element)	string	0-n	Diagnostic messages from the checking engine, with a REQUIRED *@severity* attribute that denotes the seriousness or conditions of the message. There are three message severity values: “error”, “warning”, and “info”. These elements SHALL NOT affect scoring; they are present merely to convey diagnostic information from the checking engine. Benchmark consumers MAY choose to log these messages or display them to the user.
instance (element)	string	0-n	Name of the target subsystem or component to which this result applies, for a multiply instantiated *<xccdf:Rule>*. The element is important for an *<xccdf:Rule>* that applies to components of the target system, especially when a target might have several such components, and where the *@multiple* attribute of the *<xccdf:Rule>* is set to true. For example, an *<xccdf:Rule>* might specify a particular setting that needs to be applied on every interface of a firewall; for benchmark results, a firewall target with three interfaces could have three *<xccdf:rule-result>* elements with the same rule id, each with an independent value for the *<xccdf:result>* element. For more discussion of multiply instantiated *<xccdf:Rule>* elements, see Section 7.2.3.4. The OPTIONAL *@context* and *@parentContext* attributes provide context and hierarchy information for nested contexts. If the *@context* attribute is omitted, the value of the context SHALL default to “undefined”. At most one *<xccdf:instance>* child of an *<xccdf:rule-result>* MAY have a context of “undefined”.
fix (element)	string	0-n	Fix script for this target platform, if available (would normally appear only for result values of “fail”). See Table 9. It is assumed to have been ‘instantiated’ by the testing tool and any substitutions or platform selection already made.
check (element)	*special*	(1-n instances of check) XOR (1 instance of complex-check)	Encapsulated or referenced results to detailed testing output from the checking engine (if any). Consists of the URI that designates the checking system, and detailed output data from the checking engine. The detailed output data MAY take the form of encapsulated XML or text data, or it MAY be a reference to an external URI. (Note: this is analogous to the form of the *<xccdf:Rule>* element’s *<xccdf:check>* element, used for referring to checking engine input.) See Section 6.4.4.4.
complex-check (element)	*special*		A copy of the *<xccdf:Rule>* element’s *<xccdf:complex-check>* element where each component *<xccdf:check>* element of the *<xccdf:complex-check>* element is an encapsulated or referenced results to detailed testing output from the checking engine (if any) as described above. See Section 6.4.4.4.

Property	Type	Count	Description
idref (attribute)	identifier	1	Identifier of an *<xccdf:Rule>* (from the benchmark designated in the *<xccdf:TestResult>*).
role (attribute)	string	0-1	The role string copied from the *@role* attribute of the *<xccdf:Rule>*.
severity (attribute)	string	0-1	The *@severity* attribute value copied from the *<xccdf:Rule>* (default: "unknown").
time (attribute)	dateTime	0-1	Time when application of this instance of this *<xccdf:Rule>* was completed.
version (attribute)	string	0-1	The version number string copied from the *<xccdf:version>* element of the *<xccdf:Rule>*.
weight (attribute)	decimal	0-1	The weight number copied from the *@weight* attribute of the *<xccdf:Rule>*. Expressed as a non-negative real number (0.0 or greater, maximum 3 digits, default 1.0).

6.6.4.2 <xccdf:result> Element

Table 26 lists the possible results of a single test (assigned to the *<xccdf:result>* element).

Table 26: Possible Results for a Single Test

Result and Abbreviation	Description
pass [P]	The target system or system component satisfied all the conditions of the *<xccdf:Rule>*.
fail [F]	The target system or system component did not satisfy all the conditions of the *<xccdf:Rule>*.
error [E]	The checking engine could not complete the evaluation, therefore the status of the target's compliance with the *<xccdf:Rule>* is not certain. This could happen, for example, if a testing tool was run with insufficient privileges and could not gather all of the necessary information.
unknown [U]	The testing tool encountered some problem and the result is unknown. For example, a result of 'unknown' might be given if the testing tool was unable to interpret the output of the checking engine (the output has no meaning to the testing tool).
notapplicable [N]	The *<xccdf:Rule>* was not applicable to the target of the test. For example, the *<xccdf:Rule>* might have been specific to a different version of the target OS, or it might have been a test against a platform feature that was not installed.
notchecked [K]	The *<xccdf:Rule>* was not evaluated by the checking engine. This status is designed for *<xccdf:Rule>* elements that have no *<xccdf:check>* elements or that correspond to an unsupported checking system. It may also correspond to a status returned by a checking engine if the checking engine does not support the indicated check code.
notselected [S]	The *<xccdf:Rule>* was not selected in the benchmark.
informational [I]	The *<xccdf:Rule>* was checked, but the output from the checking engine is simply information for auditors or administrators; it is not a compliance category. This status value is designed for *<xccdf:Rule>* elements whose main purpose is to extract information from the target rather than test the target.
fixed [X]	The *<xccdf:Rule>* had failed, but was then fixed (possibly by a tool that can automatically apply remediation, or possibly by the human auditor).

6.6.4.3 <xccdf:override> Element

The *<xccdf:override>* element provides a mechanism for an auditor to change the result assigned by the checking tool or to document the reason for deviation from the rule requirement. This is necessary

when checking a rule requires reviewing manual procedures or other non-IT conditions, and/or when a check gives an inaccurate result on a particular target system. The *<xccdf:override>* element SHALL contain the properties listed in Table 27. Note that the *<xccdf:override>* element is unrelated to the *@override* attribute (Section 6.3.1). If the *<xccdf:override>* element is present, the *<xccdf:result>* element SHALL be changed to match the *<xccdf:override>* element's *<xccdf:new-result>* property.

Table 27: <xccdf:override> Element Properties

Property	Type	Count	Description
old-result (element)	string	1	The rule result status before this override
new-result (element)	string	1	The new, override rule result status
remark (element)	string	1	Rationale or explanation text for why or how the override was applied. It MAY have an *@xml:lang* attribute (see Section 6.2.10).
time (attribute)	dateTime	1	When the override was applied
authority (attribute)	string	1	Name or other identification for the human principal authorizing the override

The example below shows how an *<xccdf:override>* element would appear in an *<xccdf:rule-result>*. Note: if an *<xccdf:override>* element is added to an *<xccdf:rule-result>* that was previously signed, it will break any XML digital signature applied to the enclosing *<xccdf:TestResult>* element.

```
<xccdf:rule-result idref="xccdf_org.example_rule_rule76" time="2005-04-26T14:38:19Z"
severity="low">
  <xccdf:result>pass</xccdf:result>
  <xccdf:override time="2005-04-26T15:03:20Z" authority="Bob Smith">
    <xccdf:old-result>fail</xccdf:old-result>
    <xccdf:new-result>pass</xccdf:new-result>
    <xccdf:remark>
      Manual inspection showed this rule is satisfied. The relevant registry
      key was protected per policy, but with a more restrictive ACL than the
      benchmark was designed to check. The rule result has been overridden to 'pass'.
    </xccdf:remark>
  </xccdf:override>
  <xccdf:instance context="registry">
    HKLM\SOFTWARE\Policies
  </xccdf:instance>
  <xccdf:check system="http://www.mitre.org/XMLSchema/oval-definitions-5">
    <xccdf:check-content-ref href="oval-out.xml" name="oval:org.example:def:123"/>
  </xccdf:check>
</xccdf:rule-result>
```

6.6.5 <xccdf:tailoring-file> Element

The *<xccdf:tailoring-file>* element in the *<xccdf:TestResult>* element is used to provide information about an *<xccdf:Tailoring>* element. Table 28 shows the properties of an *<xccdf:tailoring-file>* element. The *<xccdf:tailoring-file>* element MUST be present if and only if one of the *<xccdf:Profile>* elements of the *<xccdf:Tailoring>* element was applied or created during the activities that created the given *<xccdf:TestResult>* element. "Applied" refers to applying a profile during benchmark profile processing, while "created" refers to a

user's manual tailoring actions resulting in the generation of a new *<xccdf:Tailoring>* element. (See Section 6.7 for information on *<xccdf:Tailoring>*.)

Table 28: <xccdf:tailoring-file> Element Properties

Property	Type	Count	Description
href (attribute)	URI	1	Persistent location of the *<xccdf:Tailoring>* element
id (attribute)	identifier	1	Identifier for the *<xccdf:Tailoring>* element
version (attribute)	string	1	Version of the *<xccdf:Tailoring>* element
time (attribute)	dateTime	1	Timestamp for the *<xccdf:Tailoring>* element

6.7 <xccdf:Tailoring> Element

6.7.1 Basics

An *<xccdf:Tailoring>* element allows named tailorings (i.e., profiles) of a benchmark to be defined separately from the benchmark itself. There are several reasons why this might be desirable:

- The benchmark might not be controlled by the organization wishing to add the profile to it.
- The benchmark might have digital signatures that would be corrupted by benchmark modification.
- The benchmark might undergo revision by its author, so modifications by a different party would represent a development fork that complicates maintenance.
- It enables the capturing of manual tailoring actions in a well-defined format (see Section 5.2).

The profiles in an *<xccdf:Tailoring>* element can be used in two ways. First, an organization might wish to pre-define a set of tailoring actions to be applied on top of or instead of the tailoring performed by a benchmark's profiles. This may be necessary to adjust the benchmark to the local needs of an enterprise. By creating new *<xccdf:Profile>* elements with these tailorings and saving them in the *<xccdf:Tailoring>* element, these can be applied to future assessments. The *<xccdf:Tailoring>* elements also serve as records of these additional tailoring actions, providing the context needed to interpret *<xccdf:TestResult>* elements.

In addition, an *<xccdf:Tailoring>* element can be used to to record manual tailoring actions performed during the course of an assessment. Manual tailoring actions SHOULD be limited to actions that could be performed using selectors in an *<xccdf:Profile>* element. If this is done, any manual tailoring action can be recorded in a series of selectors in a newly defined *<xccdf:Profile>* element. By storing this *<xccdf:Profile>* in an *<xccdf:Tailoring>* element, the processes that led to a particular set of *<xccdf:TestResult>* elements can be saved for future reference. Of course, such an *<xccdf:Tailoring>* element could then be used as input to subsequent assessments.

6.7.2 Properties

An *<xccdf:Tailoring>* element holds one or more *<xccdf:Profile>* elements. These *<xccdf:Profile>* elements record additional tailoring activities that apply to a given *<xccdf:Benchmark>*. *<xccdf:Tailoring>* elements are separate from benchmark documents, but each *<xccdf:Tailoring>* element is associated with a specific benchmark document. By

defining these tailoring actions separately from the benchmark document to which they apply, these actions can be recorded without affecting the integrity of the source itself.

Table 29 lists the possible properties for the *<xccdf:Tailoring>* element.

Table 29: <xccdf:Tailoring> Element Properties

Property	Type	Count	Description
benchmark (element)	*special*	0-1	Identifies the Benchmark to which this tailoring applies. A tailoring document is only applicable to a single Benchmark. The *<xccdf:benchmark>* element holds the associated Benchmark's location URI, version string, and id value. Note, however, that this is a purely informative field. Benchmark consumers are not required to perform any lookup or even verify that a tailoring document is being applied to the correct Benchmark.
status (element)	*special*	0-n	Status of the tailoring and date at which it attained that status. Authors MAY use this element to record the maturity or consensus level of a tailoring. See Section 6.2.8.
dc-status (element)	*special*	0-n	Holds additional status information using the Dublin Core format. See Section 6.2.8.
version (element)	*special*	1	The version of this tailoring, with a REQUIRED *@time* attribute (when it was created). This is necessary because, under some circumstances, a copy of a tailoring document might be automatically generated. Without the version and timestamp, tracking of these automatically created tailoring documents could become problematic.
metadata (element)	*special*	0-n	XML metadata for the tailoring. See Section 6.2.4.
Profile (element)	Profile	1-n	Profiles that reference and customize sets of items in a Benchmark; see Section 6.5. These elements are identical to Profiles that might be found in a normal Benchmark. Moreover, they can extend the Profiles in the tailoring document's associated Benchmark using normal Profile extension mechanics. However, Profiles in the tailoring document cannot themselves be extended. For this reason, no Profile in the tailoring document should have its abstract property set to "true".
signature (element)	*special*	0-1	A digital signature asserting authorship and allowing verification of the integrity of the tailoring. See Section 6.2.7.
id (attribute)	*special*	1	Unique identifier for this element. See Section 6.2.3.
Id (attribute)	*special*	0-1	An identifier used for referencing elements included in an XML signature. See Section 6.2.7.

6.7.3 Profile Shadowing

Profiles in an *<xccdf:Tailoring>* element MAY "shadow" profiles in the associated benchmark document. When a tailoring profile shadows a benchmark profile, it assumes the identifier of that profile in all subsequent processing, and the shadowed benchmark profile effectively becomes invisible. Shadowing occurs when the tailoring profile's *@id* AND *@extends* attributes are both identical to the *@id* attribute of the benchmark profile. Note that a tailoring profile MAY extend a benchmark profile without shadowing it; the tailoring profile would simply use an identifier different from the identifier of the benchmark profile. However, a tailoring profile's *@id* attribute SHALL NOT duplicate a benchmark profile's id unless the tailoring profile is extending the benchmark profile.

Table 30 summarizes this behavior. In this table, consider an *<xccdf:Tailoring>* element with a single *<xccdf:Profile>* whose *@id* and *@extends* attributes have the given values. This

<xccdf:Tailoring> element is associated with an *<xccdf:Benchmark>* that likewise has an *<xccdf:Profile>* whose *@id* attribute has the given value.

Table 30: Profile Shadowing Behavior

Tailoring Profile		Benchmark Profile	Behavior
id	**extends**	**id**	
A	A	A	The tailoring profile shadows the benchmark profile. If profile "A" is selected and applied, it will be the profile of that name as defined in the *<xccdf:Tailoring>* element.
B	A	A	The tailoring profile extends but does not shadow the benchmark profile. If profile "A" is applied, it is the benchmark profile of that name that is applied. If profile "B" is applied, it is the tailoring profile of that name that is applied.
A	B	A	An error is thrown during processing: the tailoring profile identifier duplicates the identifier of a benchmark profile without extending that profile.

6.7.4 Tailoring Actions and Profile Selectors

As mentioned earlier, a tailoring profile can be used to record manual tailoring actions and to serve as a record of these actions when evaluating a given *<xccdf:TestResult>*. In such a case, the following user tailoring actions are represented by the profile selectors designated in Table 31.

Table 31: Tailoring Actions and Profile Selectors

Action	Selector
User selects or de-selects a Rule	*<xccdf:select>*
User provides a value for use with a given Value	*<xccdf:set-value>*
User provides a list of values for use with a given Value	*<xccdf:set-complex-value>*
User switches the check that a Rule is to perform	*<xccdf:refine-rule>*
User changes the weight of a Rule or Group	*<xccdf:refine-rule>*

7. XCCDF Processing

7.1 Introduction

The XCCDF specification is designed to support automated XCCDF document processing by a variety of products. There are three basic types of processing that a benchmark consumer might apply to an XCCDF document:

- **Tailoring** involves loading an XCCDF benchmark document, applying customizations through the application of profiles or through user actions, and then generating an XCCDF benchmark document that incorporates the tailoring.
- **Document Generation** involves loading an XCCDF benchmark document and generating textual or formatted output, usually in a form suitable for printing or human perusal.
- **Assessing** involves loading an XCCDF benchmark document, checking target systems or data sets that represent the target systems, computing one or more scores, and generating one or more *<xccdf:TestResult>* elements. Some products also generate other outputs or store compliance information in some kind of database.

XML schema validation SHOULD be performed by benchmark consumers prior to processing XCCDF benchmark documents.

Digital signature generation and validation MAY be performed by products as part of processing. However, because signatures are only valid on a source document, not valid after the document has been processed, products that perform signature validation on source documents MUST do so after XInclude processing and before performing any other processing on those documents.

7.2 Loading and Traversal

7.2.1 Introduction

Each type of processing includes two common steps: loading the XCCDF document, then traversing its contents to generate output. Loading and traversal are discussed below. Note that loading MUST be complete before traversal begins.

7.2.2 Loading

Table 32 explains the basics of the loading processing sequence.

Table 32: Loading Processing Sequence Sub-Steps

Sub-Step	Description
Loading.Import	Import the XCCDF document into the program and build an initial internal representation of its elements and attributes. If the document cannot be read or parsed, then Loading fails. (At the beginning of this step, any inclusion processing specified with XInclude elements MUST be performed in compliance with [XINCLUDE]. The resulting XML information set SHOULD be validated against the XCCDF schema; if validation fails, then Loading fails. XML Inclusion processing is independent of all XCCDF processing and MUST happen before any XCCDF validation or other processing.) Go to the next sub-step.

Sub-Step	Description
Loading.Noticing	For each *<xccdf:notice>* element of the *<xccdf:Benchmark>* element, add the notice to the product's set of legal notices. If a notice with an identical *@id* value is already a member of the set, then an error SHOULD be raised. Go to the next sub-step.
Loading.Resolve	If the *@resolved* attribute of the *<xccdf:Benchmark>* element is set to true, which asserts that the other Loading.Resolve sub-steps are unnecessary, then Loading succeeds, otherwise go to the next sub-step.
Loading.Resolve.Items	For each item in the *<xccdf:Benchmark>* that has an *@extends* attribute, resolve it by using the following steps: (1) resolve the extended item (i.e., perform Loading.Resolve.Items on the extended item) (2) insert the necessary property sequences from the extended item into the appropriate locations in the extending item (see Table 33 below) (3) remove the *@extends* attribute If any item's *@extends* identifier does not match the identifier of a visible item of the same type, then Loading fails. If the directed graph formed by the *@extends* attributes includes a loop, then indicate a processing error and Loading fails, otherwise go to the next sub-step.
Loading.Resolve.Profiles	For each *<xccdf:Profile>* in the *<xccdf:Benchmark>* that has an *@extends* attribute, resolve the set of properties in the extending *<xccdf:Profile>* by applying the following steps: (1) resolve the extended *<xccdf:Profile>* (i.e., perform Loading.Resolve.Profiles on the extended profile) (2) insert the necessary property sequences from the extended *<xccdf:Profile>* into the appropriate locations in the extending *<xccdf:Profile>* (see Table 33 below) If any *@extends* identifier does not match the identifier of another *<xccdf:Profile>* in the *<xccdf:Benchmark>*, then Loading fails. If the directed graph formed by the *@extends* attributes includes a loop, then indicate a processing error and Loading fails. Otherwise, go to the next sub-step.
Loading.Resolve.Tailoring	If no *<xccdf:Tailoring>* element is being applied to this *<xccdf:Benchmark>*, go to the next sub-step. Otherwise, for each *<xccdf:Profile>* in the *<xccdf:Tailoring>* element that has an *@extends* attribute, resolve the set of properties in the extending *<xccdf:Profile>* by applying the following steps: (1) prepend the property sequence from the extended *<xccdf:Profile>* to that of the extending *<xccdf:Profile>* (see Table 33 below) (2) if the *<xccdf:Profile>* element's *@id* and *@extends* attributes are both identical to the *@id* of an *<xccdf:Profile>* element in the source *<xccdf:Benchmark>*, then set the *@abstract* attribute of the extended *<xccdf:Profile>* in the source *<xccdf:Benchmark>* to true. If any *<xccdf:Profile>* *@extends* attribute identifier does not match the identifier of another *<xccdf:Profile>* in the source *<xccdf:Benchmark>*, then Loading fails. If any tailoring profile's identifier duplicates the identifier of a benchmark profile in the source benchmark without also extending that profile, then Loading fails. Otherwise, go to the next sub-step.
Loading.Resolve.Abstract	For each item in the *<xccdf:Benchmark>* for which the *@abstract* attribute is true, remove the item. Any item for which the *@abstract* attribute is true SHALL NOT be included in any generated document and SHALL NOT be exported to any checking engine or used in any check. For each *<xccdf:Profile>* in the *<xccdf:Benchmark>* for which the *@abstract* attribute is true, remove the *<xccdf:Profile>*. Go to the next sub-step.
Loading.Resolve.Finalize	Set the *<xccdf:Benchmark>* *@resolved* attribute to true; Loading succeeds.

If Loading succeeds for an XCCDF document, the internal data model SHOULD be complete and every item SHOULD contain all of its own content. An XCCDF document that has no `@extends` or `@abstract` attributes is called a resolved document.

During the Loading.Resolve.Items and Loading.Resolve.Profiles steps, the processor MUST flatten inheritance relationships. The conceptual model for XCCDF object properties is a list of name-value pairs; property values defined in an extending object are added to the list inherited from the extending object. Where they are added to this list depends on the inheritance processing model for the given property. There are five such models:

- **None** – the property value or values are not inherited.
- **Prepend** – the property values are inherited from the extended object, but values on the extending object come first, and inherited values follow.
- **Append** – the property values are inherited from the extended object; additional values may be defined on the extending object and appear after the inherited values.
- **Replace** – the property value is inherited; a property value explicitly defined on the extending object replaces an inherited value.
- **Override** – if explicitly tagged as 'override' (by setting the `@override` attribute to "true"), the property is processed as if it uses the Replace model. Otherwise, the property is processed as if it uses the Append model.

Table 33 shows the inheritance processing model for each of the properties supported on `<xccdf:Rule>`, `<xccdf:Group>`, `<xccdf:Value>`, and `<xccdf:Profile>` elements.

Table 33: Inheritance Processing Model

Processing Model	Properties	Remarks
None	abstract, cluster-id, extends, id, signature, status, dc-status	These properties cannot be inherited at all; they MUST be given explicitly
Prepend	source, choices	
Append	requires, conflicts, ident, fix, value, complex-value, default, complex-default, lower-bound, upper-bound, match, select, refine-value, refine-rule, set-value, set-complex-value, profile-note	
Replace	hidden, prohibitChanges, selected, version, weight, operator, interfaceHint, check, complex-check, role, severity, type, interactive, multiple, note-tag, impact-metric	For the check property, checks with different systems or different selectors SHALL be considered different properties
Override	title, description, platform, question, rationale, warning, reference, fixtext	For properties that have a locale specified (xml:lang), values with different locales SHALL be considered different properties

Group extension is deprecated in XCCDF 1.2; however, if it is used, the Loading.Resolve.Items step MUST generate a fresh unique id for any Group, Rule, or Value object that gets created through extension of its enclosing Group. This could be accomplished by generating and assigning a random unique id during Loading.Resolve.Items. It should be emphasized, however, that use of this feature is strongly discouraged because the lack of any standardized procedure for id generation means that tools from different vendors are unlikely to handle group extension the same way, leading to problems with interoperability.

7.2.3 Traversal

7.2.3.1 Introduction

The second processing step is Traversal. The concept behind Traversal is basically an in-order, depth-first walk through all the items that make up a benchmark. The following subsections explain how Traversal works for Benchmark, item, Profile, and check elements.

7.2.3.2 Benchmark Processing Algorithm

Table 34 explains the basics of the benchmark processing algorithm.

Table 34: Benchmark Processing Algorithm Sub-Steps

Sub-Step	Description
Benchmark.Front	Process the properties of the *<xccdf:Benchmark>* element that are not processed in other sub-steps.
Benchmark.Profile	If the identifier of an *<xccdf:Profile>* was specified, then apply the settings in the *<xccdf:Profile>* to the *<xccdf:Benchmark>*. At most one *<xccdf:Profile>* id MAY be specified in a single instance of document generation or assessment.
Benchmark.ManualTailoring	Benchmark consumer products that are also benchmark producers MAY allow users to apply manual tailoring actions at this time. If that happens, the product SHOULD generate a new *<xccdf:Tailoring>* element to record these actions. The nature of this element depends on prior actions: • If no *<xccdf:Profile>* was applied in the Benchmark.Profile step, then the new *<xccdf:Tailoring>* element contains a single *<xccdf:Profile>* consisting of selectors documenting user tailoring. This new *<xccdf:Profile>* does not extend any other *<xccdf:Profile>* and must have its own unique identifier. • If an *<xccdf:Profile>* from the source *<xccdf:Benchmark>* was applied in the Benchmark.Profile step, then the new *<xccdf:Tailoring>* element contains a single *<xccdf:Profile>* consisting of selectors documenting user tailoring. This *<xccdf:Profile>* extends the applied source benchmark profile and duplicates its identifier (i.e., it shadows that source benchmark profile). • If an *<xccdf:Profile>* from some other *<xccdf:Tailoring>* element was applied in the Benchmark.Profile step, then the new *<xccdf:Tailoring>* element is created as a copy of the utilized *<xccdf:Tailoring>* element with the same id, an iterated version, and an updated timestamp. Selectors documenting user tailoring actions are then appended to the copy of the applied tailoring profile. In all cases, *<xccdf:TestResult>* elements will record the new/modified *<xccdf:Profile>* in the new *<xccdf:Tailoring>* element in order to provide traceability of user tailoring actions.
Benchmark.Content	If the processing type is Tailoring, skip to the next sub-step. Otherwise, for each item in the *<xccdf:Benchmark>*, initiate Item.Process (see Table 35).
Benchmark.Back	Finalize the processing of the *<xccdf:Benchmark>*.

The sub-steps Front and Back will be different for each kind of processing, and each product MAY perform specialized handling of the benchmark properties that are processed during the Front and Back sub-steps. For document generation, profiles MAY be processed separately as part of Benchmark.Back to generate part of the output document.

7.2.3.3 Item Processing Algorithm

7.2.3.3.1 Basics

Table 35 explains the basics of the item processing algorithm. Note that when the processing type is Tailoring, Item processing is not performed.

Table 35: Item Processing Algorithm Sub-Steps

Sub-Step	Description
Item.Process	Examine the contents of the *<xccdf:requires>* and *<xccdf:conflicts>* elements; if any instances of *<xccdf:requires>* have all their items unselected, or any *<xccdf:conflicts>* instances have any items selected, then set the *@selected* attribute to false. See Section 7.2.3.3.2.
Item.Select	If any of the following conditions holds, cease processing of this item. • The processing type is Document Generation, and the item's *@hidden* attribute is true. • The processing type is Assessing, and the item's *@selected* attribute is false. If this item is a rule, its result becomes notselected (see Table 26). • The processing type is Assessing, and the item is a rule with a *@role* attribute whose value is "unchecked". The result of this rule becomes notchecked (see Table 26). • The processing type is Assessing, and the current platform (if known by the product) is not a member of the set of platforms for this item. If this item is a rule, its result becomes notapplicable (see Table 26). At the beginning of Document Generation, a user may have specified a platform to constrain document generation. If the user-defined platform used for document generation is not a member of the set of platforms for this item, then the product MAY stop processing of this item.
Group.Front	If the item is an *<xccdf:Group>*, then process its properties.
Group.Content	If the item is an *<xccdf:Group>*, then for each item in the *<xccdf:Group>*, initiate Item.Process.
Rule.Content	If the item is an *<xccdf:Rule>*, then process its properties (see Section 7.2.3.5).
Value.Content	If the item is an *<xccdf:Value>*, then process its properties.

The list below describes some of the processing in more detail.

- For Document Generation, the key to processing is to generate an output stream that can be formatted as a readable or printable document. The exact formatting discipline depends on the tool and the target output format. In general, the *@selected* attribute is not germane to Document Generation.
- For Assessing, the key to processing is applying the *<xccdf:Rule>* checks to the target system or collecting data about the target system. It is also possible that some *<xccdf:Rule>* checks will need to be applied to multiple contexts or features of the target system or trigger multiple blocks of code in the checking language, generating multiple pass or fail results for a single *<xccdf:Rule>* element. For more information, see the *<xccdf:multi-check>* discussion in Section 6.4.4.4 and the *<xccdf:multiple>* discussion in Section 6.4.4.2.

7.2.3.3.2 <xccdf:requires> and <xccdf:conflicts> Elements

To prevent ambiguity, benchmark consumers MUST process the items of the *<xccdf:Benchmark>* in order, and MUST NOT change the selected property of any *<xccdf:Rule>* or *<xccdf:Group>* more than once during a processing session. It should be emphasized that *<xccdf:Group>* and *<xccdf:Rule>* elements SHALL NOT change from deselected to selected based on their

<xccdf:requires> and *<xccdf:conflicts>* elements. Also, *<xccdf:requires>* and *<xccdf:conflicts>* elements SHALL only change their parent item; they SHALL NOT modify other items in the *<xccdf:Benchmark>*. Finally, note also that *<xccdf:requires>* and *<xccdf:conflicts>* elements SHALL NOT be evaluated more than once. Later changes to a Benchmark's state might result in deselections that would cause a previous evaluation of requires/conflicts properties to come to a different conclusion. However, prior evaluations SHALL NOT change, even if their results become "incorrect" after subsequent items are processed.

Rules and Groups may contain any number of *<xccdf:requires>* and *<xccdf:conflicts>* elements and if any of these elements do not evaluate to true, then that item SHALL become deselected. Essentially, the results of the individual *<xccdf:requires>* and *<xccdf:conflicts>* elements are ANDed together to determine whether a given item's *<xccdf:requires>* and *<xccdf:conflicts>* elements are met.

Here are a few examples of the processing of *<xccdf:requires>* and *<xccdf:conflicts>* elements. In all examples, it is assumed that application of profiles and/or manual tailoring has already occurred.

Example #1 – Simple requires/conflicts example

Below is a simple example of an *<xccdf:Rule>* that uses *<xccdf:requires>* and *<xccdf:conflicts>*:

```
<xccdf:Rule id="xccdf_org.example_rule_Rule1" selected="true">
    ..
    <xccdf:requires idref="xccdf_org.example_rule_Rule2
                           xccdf_org.example_rule_Rule3"/>
    <xccdf:requires idref="xccdf_org.example_group_Group1" />
    <xccdf:conflicts idref="xccdf_org.example_rule_Rule4" />
    ...
</xccdf:Rule>
```

The above *<xccdf:Rule>* would only be selected if at least one of Rule2 or Rule3 was selected, if Group1 was selected, and if Rule4 was not selected. Expressed in boolean logic format, Rule1's requires/conflicts parameters would be met if and only if:

```
((Rule2 OR Rule3) AND Group1 AND ~Rule4)
```

In the above algebra, a name evaluates to "true" if and only if the named item is selected. Note that if Rule1 were already de-selected, the requires/conflicts evaluation becomes moot – Rule1 would never change to selected even if all its requires and conflicts properties were met. Likewise, Rule1's requires and conflicts statements never affect Rule2, Rule3, Rule4, or Group1.

Example #2 – Ordering of requires/conflicts

As mentioned earlier, an item's compliance with its *<xccdf:requires>* and *<xccdf:conflicts>* elements is only evaluated at one point in time and subsequent changes to the benchmark's state, even if they would make those evaluations "incorrect" if re-run, do not change the prior results. Consider the following *<xccdf:Rule>* elements:

```
<xccdf:Rule id="xccdf_org.example_rule_Rule1" selected="true">
    ...
    <xccdf:requires idref="xccdf_org.example_rule_Rule2">
    ...
```

```
</xccdf:Rule>
<xccdf:Rule id="xccdf_org.example_rule_Rule2" selected="true">
    ...
    <xccdf:requires idref="xccdf_org.example_rule_Rule3">
    ...
</xccdf:Rule>
<xccdf:Rule id="xccdf_org.example_rule_Rule3" selected="false">
    ...
    <xccdf:requires idref="xccdf_org.example_rule_Rule4">
    ...
</xccdf:Rule>
<xccdf:Rule id="xccdf_org.example_rule_Rule4" selected="true">
    ...
</xccdf:Rule>
```

In the above example, Rule1 requires Rule2, Rule2 requires Rule3, and Rule3 requires Rule4, although since Rule3 is already deselected, its requires statements are irrelevant. Looking at the above scenario, one might be tempted to believe that Rule1, Rule2, and Rule3 will all end up deselected, but this is not the case. The following steps show how item processing of these Rules would proceed. For this example, we will assume we are doing assessment.

1. **Item.Process(Rule1)** – Because Rule2 is required, we see if Rule2 is selected. It is, so we make no change to Rule1's selection status.
2. **Item**.**Select(Rule1)** – Rule1 is selected. Continue processing Rule1.
3. **Rule.Content(Rule1)** – Process Rule1's content.
4. **Item.Process(Rule2)** – Because Rule3 is required, we see if Rule3 is selected. It is not, so we set selected on Rule2 to false.
5. **Item**.**Select(Rule2)** – Rule2 is not selected. Terminate processing of Rule2.
6. **Item.Process(Rule3)** – Because Rule4 is required, we see if Rule4 is selected. It is, but Rule 3 is already de-selected and remains so.
7. **Item**.**Select(Rule3)** – Rule3 is not selected. Terminate processing of Rule3.
8. **Item.Process(Rule4)** – Rule4 has no requires/conflicts properties so this step is skipped.
9. **Item**.**Select(Rule4)** – Rule4 is selected. Continue processing Rule4.
10. **Rule.Content(Rule4)** – Process Rule4's content.

The final result was that Rule1 and Rule4 were selected and processed while Rule2 and Rule3 were de-selected and not processed. This happens even though Rule1 requires Rule2. Because we have completed processing of Rule1's content before we start processing of Rule2, by the time we realize that Rule2's requires statement cannot be met and Rule2 becomes deselected, the effect this change would have on Rule1 is moot because Rule1 has already been run.

This example demonstrates the importance of processing items in the order in which they appear in the benchmark XML. If a benchmark consumer processed these items in a different order (for example, from the bottom up), this would result in a different set of Rule contents being processed, which would violate the XCCDF specification.

Example #3 – Requires, conflicts, and Groups

Example #2 shows how a Rule's content might be processed even though a Rule that it requires is not (eventually) selected. Another way this can happen is if a required Rule is contained in a deselected Group. Consider the following example:

```
<xccdf:Rule id="xccdf_org.example_rule_Rule1" selected="false">
    ...
</xccdf:Rule>
<xccdf:Group id="xccdf_org.example_group_Group1" selected="false">
    ...
    <xccdf:Rule id="xccdf_org.example_rule_Rule2" selected="true">
        ...
        <xccdf:requires idref="xccdf_org.example_rule_Rule1" />
        ...
    </xccdf:Rule>
    ...
</xccdf:Group>
<xccdf:Rule id="xccdf_org.example_rule_Rule3" selected="true">
    ...
    <xccdf:requires idref="xccdf_org.example_rule_Rule2" />
    ...
</xccdf:Rule>
```

Because Group1 is deselected, none of its contents will ever be processed. Thus, even though Rule2 would never be run and even though its requires property would not be met, it remains selected and, as such, allows the requires statement of Rule3 to evaluate to true. The following steps show how Item Processing of these Rules would proceed. For this example, we will assume we are doing assessment.

1. **Item.Process(Rule1)** – Rule1 has no requires/conflicts properties so this step is skipped.
2. **Item**.**Select(Rule1)** – Rule1 is not selected. Terminate processing of Rule1.
3. **Item.Process(Group1)** – Group1 has no requires/conflicts properties so this step is skipped.
4. **Item**.**Select(Group1)** – Group1 is not selected. Terminate processing of Group1. *Note that we never get to the Group.Content step so Rule2 never undergoes any form of processing.*
5. **Item.Process(Rule3)** – Because Rule2 is required, we see if Rule2 is selected. It is, so we make no change to Rule3's selection status.
6. **Item**.**Select(Rule3)** – Rule3 is selected. Continue processing Rule3.
7. **Rule.Content(Rule3)** – Process Rule3's content.

Rule3 is run even though it requires a Rule that is not run.

7.2.3.4 Profile Selector Processing

Profile selectors (*<xccdf:select>*, *<xccdf:refine-value>*, *<xccdf:set-value>*, *<xccdf:set-complex-value>*, and *<xccdf:refine-rule>* elements) SHALL be processed in the order in which they appear in the XML. Since these selectors are processed under the "append" extension processing model, an extending *<xccdf:Profile>* MAY override the inherited selectors of the *<xccdf:Profile>* it extends.

<xccdf:Profile> selector processing can be understood more easily by looking at an example. Assume the existence and initial configuration of an *<xccdf:Benchmark>* with the *<xccdf:Rule>*, *<xccdf:Group>*, and *<xccdf:Value>* elements listed in Table 36:

Table 36: Profile Selector Example: Initial Configuration

Item	id	cluster-id	selected	Defined Selectors
Rule	Rule1	Cluster2	false	(empty)
Rule	Rule2		true	(empty), sel3
Rule	Rule3	Cluster1	true	(empty)
Rule	Rule4	Cluster1	true	sel1, sel5
Rule	Rule5	Cluster3	true	sel1
Group	Group1	Cluster1	true	
Group	Group2	Cluster1	true	
Value	Value1			(empty), sel1, sel2
Value	Value2	Cluster2		(empty), sel1, sel2, sel5
Value	Value3	Cluster2		(empty), sel1, sel5, sel6
Value	Value4	Cluster3		(empty), sel4

Based on this configuration, the initial state of the *<xccdf:Benchmark>* is as listed in Table 37:

Table 37: Profile Selector Example: Initial Benchmark State

Item id	Selected?	Applicable Selector
Rule1	not selected	(empty)
Rule2	selected	(empty)
Rule3	selected	(empty)
Rule4	selected	-none-
Rule5	selected	-none-
Group1	selected	-none-
Group2	selected	-none-
Value1		(empty)
Value2		(empty)
Value3		(empty)
Value4		(empty)

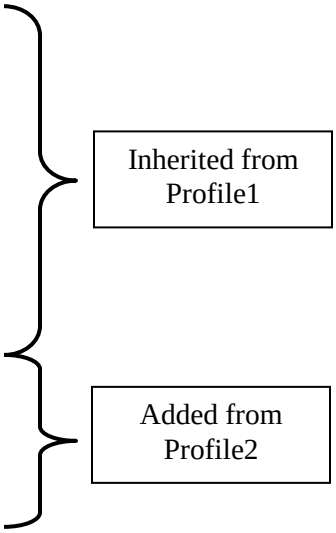
Now consider the following *<xccdf:Profile>* definitions:

```
<xccdf:Profile id="xccdf_org.example_profile_Profile1" abstract="true">
    ...
    <xccdf:select idref="xccdf_org.example_rule_Rule1" selected="true" />
    <xccdf:select idref="Cluster1" selected="false" />
    <xccdf:select idref="xccdf_org.example_group_Group1" selected="true" />
    <xccdf:refine-value idref="xccdf_org.example_value_Value1" selector="sel1" />
    <xccdf:refine-value idref="Cluster2" selector="sel2" />
    <xccdf:set-value idref="xccdf_org.example_value_Value4">NEWVALUE</set-value>
    <xccdf:refine-rule idref="xccdf_org.example_rule_Rule2" selector="sel3" />
    <xccdf:refine-rule idref="Cluster3" selector="sel1" />
</xccdf:Profile>
<xccdf:Profile id="xccdf_org.example_profile_Profile2" extends="Profile1">
    ...
    <xccdf:select idref="xccdf_org.example_rule_Rule1" selected="false" />
    <xccdf:select idref="xccdf_org.example_rule_Rule3" selected="true" />
```

```
    <xccdf:refine-value idref="Cluster2" selector="sel5" />
    <xccdf:refine-rule idref="xccdf_org.example_rule_Rule5" selector="sel6" />
</xccdf:Profile>
```

Because Profile selectors are appended under extension, after Loading steps have completed, Profile2's effective list of selectors would look like the following (with numbers added for reference and identifier names condensed for brevity):

1. <select idref="Rule1" selected="true" />
2. <select idref="Cluster1" selected="false" />
3. <select idref="Group1" selected="true" />
4. <refine-value idref="Value1" selector="sel1" />
5. <refine-value idref="Cluster2" selector="sel2" />
6. <set-value idref="Value4">NEWVALUE</set-value>
7. <refine-rule idref="Rule2" selector="sel3" />
8. <refine-rule idref="Cluster3" selector="sel1" />
9. <select idref="Rule1" selected="false" />
10. <select idref="Rule3" selected="true" />
11. <refine-value idref="Cluster2" selector="sel5" />
12. <refine-rule idref="Rule5" selector="sel6" />



If Profile2 is selected, then the Benchmark.Profile processing step will cause the following changes to the resolved Benchmark as each selector is processed:

1. Rule1 becomes "selected"
2. Rule3, Rule4, Group1, and Group2 become "not selected" due to their associations with Cluster1
3. Group1 becomes "selected", overriding the change to this setting from line 2
4. Value1 changes to using the "sel1" selector
5. Value2 and Value3 change to using the "sel2" selector due to their associations with Cluster2. However, since Value3 does not utilize any selector named "sel2", the effectively associates Value3 with the empty selector. Note that Rule1 is not affected even though it is a member of Cluster2. This is because a refine-value selector only affects Values.
6. The effective value of Value4 changes to "NEWVALUE"
7. Rule2 changes to using the "sel3" selector
8. Rule5 changes to using the "sel1" selector due to its association with Cluster3
9. Rule1 becomes "not selected", overriding the change to this setting from line 1
10. Rule3 becomes "selected", overriding the change to this setting from line 2
11. Value2 and Value3 change to using the "sel5" selector due to their associations with Cluster2. This overrides the change to these settings from line 5.
12. Rule5 changes to using the "sel6" selector, but since it does not utilize any selector named "sel6" this would lead to the use of the empty selector. Since Rule5 does not define any empty selector either, this Rule would effectively not utilize any selectable field. Since the check field is the only selectable field in a Rule, this means that Rule5 would have no check associated with it (i.e.,

under evaluation, it would return a result of "notchecked"). This overrides the change to this setting from line 8.

The final configuration of the *<xccdf:Benchmark>* due to application of Profile2 would be as shown in Table 38:

Table 38: Profile Selector Example: Final Benchmark State

Item id	Selected?	Applicable Selector
Rule1	not selected	(empty)
Rule2	selected	sel3
Rule3	selected	(empty)
Rule4	not selected	-none-
Rule5	selected	sel6 (Not defined so this becomes -none-)
Group1	selected	-none-
Group2	not selected	-none-
Value1		sel1
Value2		sel5
Value3		sel5
Value4		="NEWVALUE"

This example demonstrates how a single item could be tailored multiple times due to the influence of multiple selectors.

It should also be noted that selectors do not need to be of the same type to override each other's behaviors. All three of the value selectors, *<xccdf:refine-value>*, *<xccdf:set-value>*, and *<xccdf:set-complex-value>*, can affect the *<xccdf:value>* or *<xccdf:complex-value>* element of a named *<xccdf:Value>*. However, an *<xccdf:Value>* may have only one *<xccdf:value>* or one *<xccdf:complex-value>* element selected at any time. As a result, the use of any of the aforementioned selectors to change an *<xccdf:value>* or *<xccdf:complex-value>* will replace any prior tailoring of that *<xccdf:value>* or *<xccdf:complex-value>*.

7.2.3.5 Check Processing

7.2.3.5.1 Basics

During the Rule.Content sub-step of the item processing algorithm, the properties of a given *<xccdf:Rule>* are processed, including its check structures. If an *<xccdf:Rule>* contains an *<xccdf:complex-check>*, then the benchmark consumer MUST process it and MUST ignore any *<xccdf:check>* elements that are also contained by the *<xccdf:Rule>*. However, within a given *<xccdf:complex-check>*, processing of component checks SHALL follow the same procedures described below.

Check processing involves selecting one supported *<xccdf:check>* element and then executing its check code. Table 39 explains the basics of the check processing algorithm.

Table 39: Check Processing Algorithm Sub-Steps

Sub-Step	Description
Check.Initialize	Create an empty list of candidate checks.
Check.Selector	Identify the *<xccdf:check>* elements that are candidates for use. • If a selector has been identified for this Rule and there is at least one *<xccdf:check>* element with a matching *@selector* attribute, then add all such *<xccdf:check>*.elements to the candidate list in the order in which they appear and proceed to the next sub-step. • Otherwise, if there is at least one *<xccdf:check>* element with an absent or empty *@selector* attribute, then add all such *<xccdf:check>*.elements to the candidate list in the order in which they appear and proceed to the next sub-step. • Otherwise, there are no valid candidate checks. Stop check processing and give this *<xccdf:Rule>* a result of "notchecked".
Check.System	Iterate through each *<xccdf:check>* element that appears in the candidate list (in order). If the system given in the element's *@system* attribute is supported by an available checking engine: • If the parent element is an *<xccdf:Rule>*, terminate this sub-step and proceed to the next sub-step using that *<xccdf:check>*. The next sub-step will only be applied to a single *<xccdf:check>* element. • If the parent element is an *<xccdf:complex-check>*, add this *<xccdf:check>* to the list of applicable checks and continue iterating through the list of checks. After all *<xccdf:check>* elements have been processed, the remaining check processing sub-steps must be applied to each *<xccdf:check>* in the list of applicable checks. This may result in the following sub-steps being applied to multiple *<xccdf:check>* elements. • If the list of candidate *<xccdf:check>* elements is exhausted without finding one that uses a supported system, stop check processing and give this *<xccdf:Rule>* a result of "notchecked".
Check.Content	Iterate through each *<xccdf:check-content-ref>* element in the *<xccdf:check>* element (in order). If the reference can be resolved (i.e., if the checking-language code can be made available to the checking engine) then terminate this sub-step and proceed to the next sub-step using the referenced content. If the list of *<xccdf:check-content-ref>* elements is exhausted without any reference being resolvable, then if there is an *<xccdf:check-content>* element, proceed to the next step using the included content. Otherwise, stop check processing and give this *<xccdf:Rule>* a result of "notchecked".
Check.Export	Make the referenced checking-language code, as well as any exported values as indicated by *<xccdf:check-export>* elements, available to the checking engine. (This can be done immediately or it can be done in a batch after all *<xccdf:Rule>* elements in the benchmark have been processed.) If this *<xccdf:Rule>* has a *@role* attribute whose value is “unscored”, give this *<xccdf:Rule>* a result of “informational”. Otherwise, give this *<xccdf:Rule>* a result appropriate to the result returned by the checking engine.

A benchmark consumer SHALL NOT “backtrack” in the processing of these steps. For example, once a check with a preferred system is selected by the benchmark consumer (Check.System), the benchmark consumer SHALL NOT attempt to use a different check that uses a different system, even if none of the originally selected check's content can be resolved.

Figure 2 provides a flowchart that illustrates check processing.

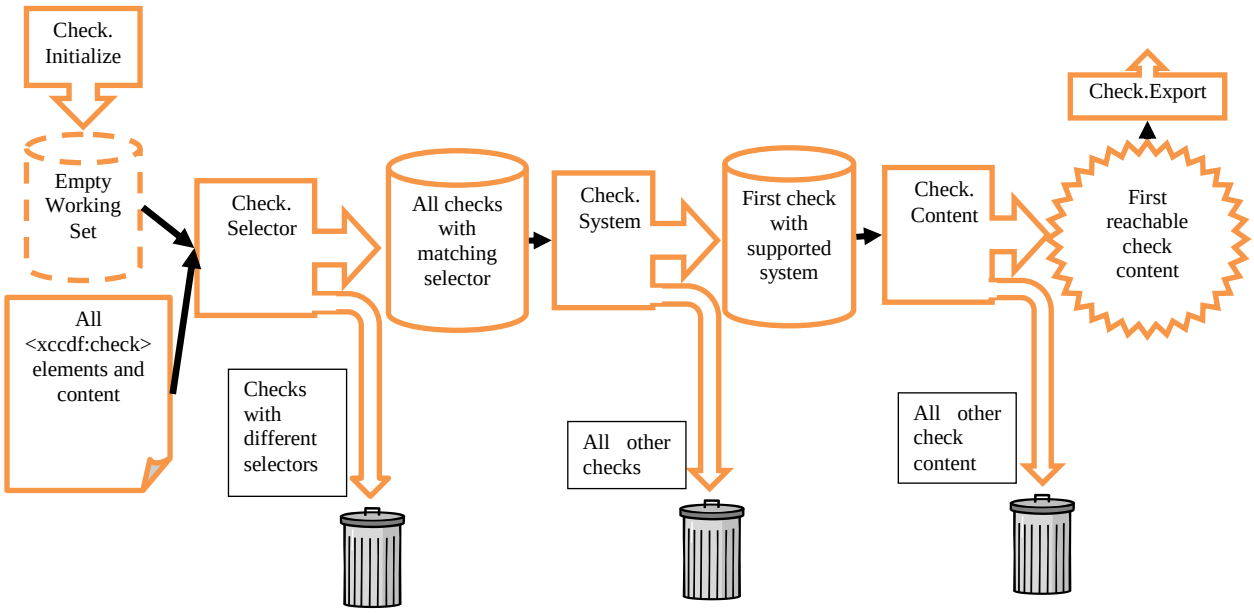


Figure 2: Check Processing Flowchart (when the check's parent is an <xccdf:Rule>)

7.2.3.5.2 Rules with Multiple Results

Many XCCDF documents include *<xccdf:Rule>* elements that apply to system components. For example, a host OS benchmark could contain *<xccdf:Rule>* elements that apply to all users, and a router benchmark could contain *<xccdf:Rule>* elements that apply to all network interfaces. When the system holds many such components, it may not be adequate for a benchmark consumer to report that an *<xccdf:Rule>* failed; it MAY report exactly which components failed the *<xccdf:Rule>*.

A processing engine that performs a checking system test MAY deliver one or more results in response to a check. In the most common case, each *<xccdf:Rule>* will yield one *<xccdf:rule-result>* element. In a case where an *<xccdf:Rule>* was applied multiple times to multiple components of the system under test, a single *<xccdf:Rule>* could yield multiple *<xccdf:rule-result>* elements. If the *<xccdf:Rule>* *@multiple* attribute is set to true, then each instance of the assessment target SHOULD be reported separately. Similarly, if an *<xccdf:check>* element leads to the execution of multiple checks (i.e., an *<xccdf:check-content-ref>* that lacks a *@name* attribute is used) and the *@multi-check* attribute is set to true, each check executed MUST be reported separately. Otherwise, an *<xccdf:Rule>* contributes to the positive score only if ANDing the results of all instances of that *<xccdf:Rule>* produces a test result of 'pass' according to the truth table that appears in the description of the *<xccdf:complex-check>* element in Section 6.4.4.4. If any component of the target system fails the checking system test, then the entire *<xccdf:Rule>* SHALL be considered to have failed. This is sometimes called "strict scoring". See Section 6.4.4.2 for more information on the *@multiple* attribute and Section 6.4.4.4 for the *@multi-check* attribute.

When creating multiple *<xccdf:rule-result>* elements that stem from a single *<xccdf:Rule>*, each of these *<xccdf:rule-result>* elements MUST identify the same *<xccdf:Rule>* in its *@idref* attribute.

- When multiple *<xccdf:rule-result>* elements are caused by multiple target instances with *@multiple* set to true, the *<xccdf:instance>* element of the *<xccdf:rule-result>* element SHOULD include, at minimum, the instance name. It MAY include additional information to provide additional context for that instance.
- When multiple *<xccdf:rule-result>* elements are caused by multiple executed checks with *@multi-check* set to true, the *<xccdf:check>* element of the *<xccdf:rule-result>* element MUST identify the executed check. This SHOULD be done by including an *<xccdf:check-content-ref>* element that explicitly points to the corresponding result content for each of the checks executed to produce this particular result. Alternatively, it MAY be done by including the checking result structure directly in an *<xccdf:check-content>* element.

It is possible for a single *<xccdf:Rule>* to reference multiple checks, some of which test multiple target instances. This would lead to both the *<xccdf:instance>* and *<xccdf:check>* elements being utilized in the manner described above.

7.2.3.6 Other Processing

7.2.3.6.1 XHTML Formatting

Some text-valued XCCDF elements may contain formatting specified with elements from [XHTML] (see Section 6.2.2). How a benchmark consumer handles embedded XHTML content in XCCDF text properties is implementation-dependent, but every benchmark consumer MUST be able to process XCCDF content even when embedded XHTML elements are present. Products that perform document generation processing SHOULD attempt to preserve the formatting semantics implied by the Text and List modules, support the link semantics implied by the Hypertext module, and incorporate the images referenced via the Image module. Such products MAY also wish to establish conventions for each of the or class attribute values (see Table 2).

7.2.3.6.2 Locale

XCCDF textual content may use the *@xml:lang* attribute to specify natural language locales. Benchmark producers and consumers SHOULD employ *@xml:lang* attributes whenever possible to create localized output. If a product has an effective language selected, it SHOULD use textual content corresponding to that language and SHOULD NOT use textual content corresponding to other languages. If a product does not have an effective language selected or ignores *@xml:lang* attributes, it MUST display all textual content in order.

7.2.3.6.3 Text Substitution

Text substitution when the *<xccdf:sub>* element's *@idref* attribute holds the id of an *<xccdf:plain-text>* element always behaves the same way: any *<xccdf:sub>* element reference to an *<xccdf:plain-text>* element SHOULD be replaced by the string content of that element.

When the *<xccdf:sub>* element's *@idref* attribute holds the id of an *<xccdf:Value>* element, the *<xccdf:sub>* element's *@use* attribute MUST be consulted.

- If the value of the *@use* attribute is "value", then the *<xccdf:sub>* element SHOULD be replaced by a string representation of the content of the currently-selected *<xccdf:value>* or *<xccdf:complex-value>* element of the referenced *<xccdf:Value>* element.

- If the value of the *@use* attribute is "title", then the *<xccdf:sub>* element SHOULD be replaced by the body of the *<xccdf:title>* element of the referenced *<xccdf:Value>* element. The *<xccdf:title>* element's *@xml:lang* attribute may be used to select the appropriate title to use if multiple titles are present. At the tool author's discretion, the title MAY be followed by the *<xccdf:Value>* element's currently-selected *<xccdf:value>* element, suitably demarcated.

- If the value of the *@use* attribute is "legacy", then during Tailoring, process the *<xccdf:sub>* element as if *@use* was set to "title", but during Document Generation or Assessment, process the *<xccdf:sub>* element as if *@use* was set to "value".

Any appearance of the *<xccdf:instance>* element in the content of an *<xccdf:fix>* element SHOULD be replaced by a locale-appropriate string to represent a target system instance name.

During creation of *<xccdf:TestResult>* elements, any *<xccdf:fix>* elements present in applied Rules and matching the platform to which the test was applied SHOULD be subjected to substitution and the resulting string used as the value of the *<xccdf:fix>* element for the *<xccdf:rule-result>* element. Each *<xccdf:sub>* element SHALL be replaced by what the *@idref* attribute references, which is either the appropriate string from the referenced *<xccdf:Value>* element, as described above, or the *<xccdf:plain-text>* definition used during the test. Formatting for this replacement is implementation-dependent for a referenced *<xccdf:Value>* element, but for an *<xccdf:plain-text>* definition it is a simple string replacement. Also, each *<xccdf:instance>* element SHOULD be replaced by the value of the *<xccdf:rule-result>* element's *<xccdf:instance>* element.

Benchmark consumers MUST support resolution of XHTML *<object>* elements, regardless of whether XHTML rendering is supported. The XHTML *<object>* element supports substitutions of a variety of information from an item or profile, or the string content of an *<xccdf:plain-text>* definition. To avoid possible conflicts with uses of an XHTML *<object>* that should not be processed specially, each XCCDF *<object>* reference MUST be a relative URI beginning with "`#xccdf:`". The following URI values can be used to refer to things from an XHTML *<object>* element, using the *@data* attribute:

- **`#xccdf:value:`*id*.** Insert the value of the *<xccdf:plain-text>* block, *<xccdf:Value>*, or *<xccdf:fact>* with id *id*. The value of the reference SHOULD be substituted for the entire *<object>* element and its content (if any). If the *id* cannot be resolved, then the textual content of the *<object>* element SHOULD be retained.

- **`#xccdf:title:`*id*.** Insert the string content of the *<xccdf:title>* element of the item with id *id*. Use the current language value locale setting, if any. The *<xccdf:title>* string SHOULD be substituted for the entire *<object>* element and its content (if any). If the *id* cannot be resolved, then the textual content of the *<object>* element SHOULD be retained.

7.2.3.6.4 Reference Processing

XCCDF benchmark consumers MUST support reference processing that uses the XHTML anchor ("a") element. The anchor element can be used to create an intra-document link to an XCCDF item or profile.

To avoid possible conflicts with uses of the XHTML anchor element that should not be processed specially, each XCCDF anchor reference MUST be a relative URI beginning with "`#xccdf:`". The URI value **`#xccdf:link:`*`id`*** can be used to refer to things from an anchor element, using the *`@href`* attribute. This creates an intra-document link to the point in the document where the item *id* is described. The content of the element SHOULD be the text of the link.

7.3 Assessment Outputs

7.3.1 Overview

When a benchmark consumer performs an assessment against a system, it accepts as inputs the state of the system and a benchmark, and MAY produce any of the following output (also shown in Figure 3):

- **Benchmark report** – A human-readable report about testing, including the score, and a listing of which rules passed and which failed on the system. If a given rule applies to multiple parts or components of the system, then multiple pass/fail entries MAY appear on this list. The report MAY also include recommended steps for improving compliance. The format of the report is not specified here, but might be some form of formatted or rich text (e.g., HTML).
- **Benchmark results** – Machine-readable testing results, meant for storage, long-term tracking, or incorporation into other reports (e.g., a site-wide report). This SHOULD be in XCCDF, using the *`<xccdf:TestResult>`* element.
- **Fix scripts** – Machine-readable content, usually text, the application of which will remediate some or all of the non-compliance issues found by the benchmark consumer. These scripts MAY be included in *`<xccdf:TestResult>`* elements (see Section 6.6).

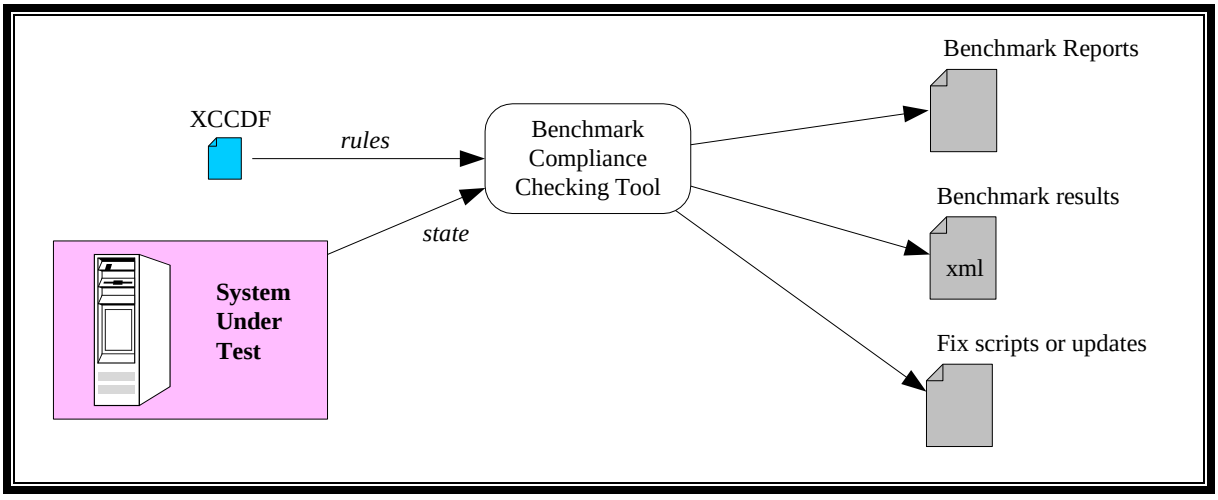


Figure 3: Workflow for Assessing Benchmark Compliance

7.3.2 Scoring Models

7.3.2.1 Overview

XCCDF has four scoring models, which are defined below. Tools MAY support additional proprietary or community models. A benchmark MAY recommend one or more scoring models to be used when computing a benchmark score by indicating them in the *`<xccdf:Benchmark>`* element's *`<xccdf:model>`* element. A tool MAY use any score computation model designated by the user. In the

four models defined below, a result of "fixed" SHALL be treated as a "pass" result for scoring purposes; other scoring models MAY handle this differently.

7.3.2.2 The Default Model

This model is identified by the URI "urn:xccdf:scoring:default". Tools MUST support this model.

In the default model, computation of the XCCDF score proceeds independently for each collection of siblings in each Group, and then for the siblings within the *<xccdf:Benchmark>*. This relative-to-siblings weighted scoring model is designed for flexibility and to foster independent authorship of collections of Rules. Benchmark authors should keep the model in mind when assigning weights to Groups and Rules.

The elements of an *<xccdf:Benchmark>* form the nodes of a tree. The default model score computation algorithm simply computes a normalized weighted sum at each tree node, omitting Rules and Groups that are not selected and Groups that have no selected Rules under them. The algorithm that SHALL be used at each selected node is listed in Table 40.

Table 40: Default Model Algorithm Sub-Steps

Sub-Step	Description
Score.Default.Rule	If the node is a Rule, initialize rule_count and rule_score to 0. Then for each associated rule-result: - if the rule-result's result is not a member of the set {notapplicable, notchecked, informational, notselected}, then add 1 to rule_count. - if the rule-result's result is "pass", add 1 to rule_score. When this has been done for every rule-result associated with this Rule: - if rule_count is 0, set this Rule's score and count to 0. - otherwise, set the Rule's count to 1 and the Rule's score to 100 * rule_score / rule_count.
Score.Default.Group.Init	If the node is a Group or the Benchmark, assign a count of 0, a score *s* of 0.0, and an accumulator *a* of 0.0.
Score.Default.Group.Recurse	For each selected child of this Group or Benchmark, do the following: (1) compute the count and weighted score for the child using this algorithm, (2) if the child's count value is not 0, then add the child's weighted score to this node's score *s*, add 1 to this node's count, and add the child's weight value to the accumulator *a*.
Score.Default.Group.Normalize	Normalize this node's score: compute $s = s / a$.
Score.Default.Weight	Assign the node a weighted score equal to the product of its score and its weight.

The final test score is the normalized score value on the root node of the tree (the *<xccdf:Benchmark>* element).

7.3.2.3 The Flat Model

This model is identified by the URI "urn:xccdf:scoring:flat". The algorithm in Table 41 SHALL be used to compute the score.

Table 41: Flat Model Algorithm Sub-Steps

Sub-Step	Description
Score.Flat.Init	Initialize both the score *s* and the maximum score *m* to 0.0.
Score.Flat.Rules	For each Rule: Initialize rule_count and rule_score to 0. For each rule-result associated with that Rule: - if the rule-result's result is not a member of the set {notapplicable, notchecked, informational, notselected}, then add 1 to rule_count. - if the rule-result's result is "pass", add 1 to rule_score. If rule_count is not 0: - add the Rule's weight to *m.* - add the Rule's weight * rule_score / rule_count to *s.*

Thus, the flat model simply computes the sum of the weights for the Rules that passed as the score, and the sum of the weights of all the applicable Rules as the maximum possible score. This model is simple and easy to compute, but scores between different target systems may not be directly comparable because the maximum score can vary. Tools SHOULD support this model.

7.3.2.4 The Flat Unweighted Model

This model is identified by the URI "urn:xccdf:scoring:flat-unweighted". It is computed exactly the same way as the flat model, except that all weights not set to 0 are taken to be 1.0. Items with weights of 0 remain 0 in this model and, as such, do not contribute to the final score. Essentially, the model computes the number of rules that passed. Tools SHOULD support this model.

7.3.2.5 The Absolute Model

This model is identified by the URI "urn:xccdf:scoring:absolute". It gives a score of 1 only when all applicable Rules in the benchmark pass, and 0 otherwise. It is computed by applying the Flat Model and returning 1 if *s*=*m*, and 0 otherwise. Tools MAY support this model.

Appendix A—Converting XCCDF 1.1.4 Content to XCCDF 1.2

XCCDF 1.2 contains several changes that are not transparently backwards compatible with XCCDF 1.1.4 content. This said, converting content between the two versions can be done easily. This appendix notes changes that prevent transparent backwards compatibility and describes procedures to convert content from the older version to the newer one. See Appendix B for a more comprehensive list of changes from XCCDF 1.1.4 to XCCDF 1.2.

A.1 Changes to the XCCDF XML Namespace

The XML namespace of XCCDF changed from "http://checklists.nist.gov/xccdf/1.1" in XCCDF 1.1.4 to "http://checklists.nist.gov/xccdf/1.2" in XCCDF 1.2. All XCCDF 1.1.4 content that needs to be validated against the XCCDF 1.2 schema must use the XCCDF 1.2 namespace.

A.2 Conversion of Identifiers

XCCDF 1.2 enforces a canonical format for the *@id* attributes (identifiers) of all major XCCDF elements: *<xccdf:Benchmark>*, *<xccdf:Rule>*, *<xccdf:Group>*, *<xccdf:Value>*, *<xccdf:Profile>*, *<xccdf:TestResult>*, and *<xccdf:Tailoring>*. As such, the values of *@id* attributes in XCCDF 1.1.4 are unlikely to be compliant with this new format. Fortunately, conversion from XCCDF 1.1.4 identifiers to XCCDF 1.2 identifiers is simple and mechanical using the following steps:

1. Specify a reverse-DNS style namespace (e.g., com.company or gov.agency), denoted as N
2. For each major XCCDF element (as listed above)
    a. Denote the type T as the name of that type of element expressed in all lower case (i.e., benchmark, rule, group, value, profile, testresult)
    b. Denote the XCCDF 1.1.4 id of that element as I
    c. The new id value of that element becomes: xccdf_N_T_I

This procedure allows any XCCDF 1.1.4 identifier to be replaced with a recognizably similar identifier value (since the old identifier value becomes part of the new identifier value) that complies with the new restrictions imposed by XCCDF 1.2. Note that references to identifiers will also need to be updated to match the changed identifier values.

A.3 Conversion of <xccdf:sub> Elements

XCCDF 1.2 gives authors a greater degree of control of how *<xccdf:sub>* elements get replaced during text substitution. In previous versions, when an *<xccdf:sub>* element referenced an *<xccdf:Value>* element, either the *<xccdf:Value>* element's title or currently-selected value would be substituted for the *<xccdf:sub>* element, depending on the processing model. In XCCDF 1.2, authors can use the *<xccdf:sub>* element's *@use* attribute to control substitution regardless of the processing model.

In XCCDF 1.2 the default value of the *@use* attribute is "value", which causes the referenced *<xccdf:Value>* element's currently-selected value to be inserted during text substitution. In all legacy content, which would not have a *@use* attribute and would therefore use this default, this would represent a change in behavior. To ensure that documents converted from XCCDF 1.1.4 to XCCDF 1.2 continue to have the same text substitution processing as before, every *<xccdf:sub>* element in the resulting

XCCDF 1.2 document should be given a *@use* attribute with a value of "legacy". The "legacy" setting indicates that substitution processing should be performed in the context-dependent manner employed by XCCDF 1.1.4 and before.

A.4 Properties Removed or Deprecated Since XCCDF 1.1.4

Several properties of *<xccdf:Benchmark>*, *<xccdf:Rule>*, and *<xccdf:Group>* elements were removed or deprecated between XCCDF 1.1.4 and XCCDF 1.2. Table 42 identifies these properties, explains the rationale behind their removal or deprecation, and identifies alternative operations that subsume their capabilities. Converting content from XCCDF 1.1.4 to XCCDF 1.2 should make use of these alternative operations.

Table 42: Alternative Operations for Removed and Deprecated XCCDF 1.1.4 Constructs

Parent Element	Property Name	Rationale and Alternatives
Benchmark	platform-definitions	All three of these properties were used to define sets of platforms to which a benchmark might apply. All used schemas that are no longer maintained and these properties have been deprecated in XCCDF since XCCDF 1.1.3 or earlier. All three properties have been removed from XCCDF 1.2. Instead of these properties, use the *<cpe2:platform-specification>* property to define sets of platforms.
Benchmark	Platform-Specification	
Benchmark	cpe-list	
Group	extends, abstract	Group extension has been deprecated in XCCDF 1.2 because it was shown to have multiple issues that make it unlikely for content that employs this feature to be interoperable across XCCDF-compliant tools. When dealing with XCCDF 1.1.4 content that employs group extension, the groups should be fully resolved when converting to XCCDF 1.2. This is to say, the act of creating the extended groups should be performed and completed, and the result then becomes the groups of the XCCDF 1.2 document.
Rule	impact-metric	The impact-metric element was found to have little use in standard XCCDF use cases, so it has been deprecated. Content stored in the impact-metric element in XCCDF 1.1.4 content should be copied to an <impact-metric> element within the corresponding Rule's metadata to preserve it when converting to XCCDF 1.2.

Appendix B—Change Log

Release 0 – 1 February 2008

- The specification for XCCDF 1.1.4 was released as final.

Release 1 – 29 July 2010 (Initial public comment draft)

Functional Additions/Changes:

- The flexibility of the metadata field was greatly expanded and metadata fields were added to all major XCCDF structures.
- The dc-status field was added to several major XCCDF structures to store status information using Dublin Core Elements 1.1 metadata.
- Results of checks can be negated.
- XCCDF 1.2 adds the concept of a complex value capable of holding lists.
- The processing of Profile selectors explicitly permits selectors to have overlapping scopes.
- The XCCDF 1.2 specification defines some sample classes to support stylistic labeling of XHTML content.
- The use of the check-import element was clarified and the import-xpath attribute was added to better support import of XML structures from checking systems.

Deprecations/Removals:

- The impact-metric element in Rules and the role attribute in Rules and rule-results were deprecated.
- Group extension (abstract and extends attributes) was deprecated.

Release 2 – 27 July 2011 (Second public comment draft)

Editorial Changes:

- The document was completely reorganized (see Table 43 below for mappings from the structure of the previous releases to this draft).
- The document was thoroughly edited. RFC 2119 language (SHALL, SHOULD, MAY, etc.) was added to explicitly declare requirements and recommendations for XCCDF documents and products.
- The document has a short glossary of key terms.

Functional Additions:

- There is a new section that explicitly defines high-level XCCDF conformance requirements for products and documents.
- This draft introduces the concept of a tailoring document. The schema now has a top-level Tailoring element, as well as a tailoring-file element within the TestResult element. Also, a Benchmark.ManualTailoring sub-step was added to the benchmark processing algorithm.
- There is a new appendix that explains how to convert XCCDF 1.1.4 content to XCCDF 1.2.
- The multi-check attribute was added to the Rule's check element, to be used to drive result reporting behavior when multiple checks are executed to determine compliance with a single Rule.

Functional Changes:

- The XCCDF namespace has been changed from "http://checklists.nist.gov/xccdf/1.1" to "http://checklists.nist.gov/xccdf/1.2". XCCDF 1.2 is no longer backwards compatible with XCCDF 1.1.4.

- Use of Common Platform Enumeration (CPE) version 2.3 for platform specification is required. Formatted string bindings are required for CPE names and applicability language expressions in XCCDF documents.
- The id attribute of Benchmark, Rule, Group, Value, Profile, TestResult, and Tailoring elements has a mandatory standard format intended to enable global uniqueness of identifiers.
- The TestResult element supports referencing asset identification information located in an external document.
- The pre-defined scoring models have been modified to compute scores per Rule rather than per rule-result.
- Complex-values support zero-length lists.

Deprecations/Removals:

- The following platform-related properties deprecated in earlier versions of XCCDF have been removed from version 1.2: cpe-list, platform-definitions, and Platform-Specification.

Table 43: Mapping Previous Release Sections to This Release

Release 0 (XCCDF 1.1.4 Final) or Release 1 (Initial Public Comment Draft)	Release 2 (Second Public Comment Draft)
1. Introduction	5.1
1.1 Background	5.1
1.2 Vision for Use	5.1
1.3 Summary of Changes since Version 1.0	Appendix B (also, deleted all changes pertaining to previous XCCDF versions)
2. Requirements	Dropped
2.1 Structure and Tailoring Requirements	5.2
2.2 Inheritance and Inclusion Requirements	5.2
2.3 Document and Report Formatting Requirements	6.2.2
2.4 Rule Checking Requirements	6.4.4.1
2.5 Test Results Requirements	5.3
2.6 Metadata and Security Requirements	6.2.7 (signatures), 6.2.4 (metadata), rest deleted
3. Data Model	6.1 (object data types), 6.4.5.1 (Value), 6.3.1 (scoring weight), data model figure deleted
3.1 Benchmark Structure	6.3.1
3.2 Object Content Details	6.2.2 (type conventions), 6.3.2 (table, count conventions)
Benchmark	6.3.2 (Benchmark table), 6.2.8 (status, dc-status), 6.2.2 (HTML markup fields, classifiers), 6.2.5 (CPE names), 6.2.9 (plain-text), 6.2.4 (metadata), 6.3.2 (style, style-href), 6.2.7 (signature)
Item	6.4.1 (Item table), 6.2.8 (status, dc-status), 6.4.1 (hidden, cluster-id)
Group	6.4.1 and 6.4.3 (Group table), 7.2.3.3.2 (requires and conflicts), 6.2.5 (platform), 6.4.1 (weight)
Rule	6.4.1 and 6.4.4.2 (Rule table) 6.3.1 and 7.2.2 (extension), 6.4.4.2 (multiple), 6.4.4.4 (multi-check), 6.2.5 (platform), 6.4.4.2 (ident), 6.4.4.4 (check), 6.4.4.5 (fixtext, fix)

Release 0 (XCCDF 1.1.4 Final) or Release 1 (Initial Public Comment Draft)	Release 2 (Second Public Comment Draft)
Value	6.4.5.2 (Value table), 6.4.5.1 (Value types), 6.4.5.2 and 6.3.1 (type, default, abstract, extends), 6.4.5.4 (operator), 6.4.5.5 (selectors), 6.4.5.2 (upper-bound, lower-bound), 6.4.5.3 (choices), 6.4.5.2 (match), 6.4.1 (prohibitChanges, hidden), 6.4.5.2 (interactive, interfaceHint, source)
Profile	6.5.2 (Profile table), 6.5.1 (definition, extension), 6.5.2 (note-tag), 6.2.8 (status, dc-status), 6.5.3 (selectors)
TestResult	6.6.2 and 6.6.1
rule-result	6.6.4.1 (rule-result Table), 6.6.4.2 (test results), 6.6.4.1 (instance, metadata, check), 6.6.4.3 (override)
3.3 Processing Models	7.1, dropped Transformation and Test Report Generation
Loading Processing Sequence	7.2.2
Benchmark Processing Algorithm	7.2.3.2
Item Processing Algorithm	7.2.3.3.1, 7.2.3.3.2
Profile Selector Processing	7.2.3.4
Substitution Processing	7.2.3.6.3
Rule Application and Compliance Scoring	7.3.1
Rule Check Processing	7.2.3.5.1
Multiply-Instantiated Rules	7.2.3.5.2
Scoring and Results Model	7.3.2, 7.3.3
Score Computation Algorithms	7.3.3
4. XML Representation	N/A
4.1 XML Document General Considerations	6.3.1, 6.2.1, 7.2.3.6.1 (XHTML requirements)
4.2 XML Element Dictionary	N/A
<Benchmark>	6.3.2
<Group>	6.4.1, 6.4.3
<Profile>	6.5.1, 6.5.2
<Rule>	6.4.1, 6.4.4.1, 6.4.4.2
<TestResult>	6.6.2, 6.6.1
<Value>	6.4.1, 6.4.5.1, 6.4.5.2
<benchmark>	6.6.2
<check>	6.4.4.4, 6.4.4.2
<check-import>	6.4.4.4
<check-export>	6.4.4.4
<check-content>	6.4.4.4
<check-content-ref>	6.4.4.4
<choices>	6.4.5.3
<choice>	6.4.5.3
<complex-check>	6.4.4.4
<complex-choice>	6.4.5.3
<complex-default>	6.4.5.3, 6.4.5.5
<complex-value>	6.4.5.3, 6.4.5.5
<conflicts>	6.4.1

Release 0 (XCCDF 1.1.4 Final) or Release 1 (Initial Public Comment Draft)	Release 2 (Second Public Comment Draft)
<cpe-list>	Appendix A
<dc-status>	6.2.8
<default>	6.4.5.2
<description>	Section 6 (several places), 6.2.9 (sub)
<external-type>	Removed from XCCDF
<fact>	6.6.2
<fix>	6.4.4.5, 6.6.4.1 (child of rule-result)
<fixtext>	6.4.4.5
<front-matter>	6.3.2
<ident>	6.4.4.2
<identity>	6.6.2
<impact-metric>	6.4.4.2
<instance>	6.6.4.1 (child of rule-result), 6.4.4.5 (child of fix)
<item>	N/A
<lower-bound>	6.4.5.2
<match>	6.4.5.2
<message>	6.6.4.1
<metadata>	6.2.4
<model>	6.3.2
<new-result>	6.6.4.3
<notice>	6.3.2
<old-result>	6.6.4.3
<organization>	6.6.2
<override>	6.6.4.3
<param>	6.3.2
<plain-text>	6.3.2, 6.2.9
<platform>	6.2.5
<platform-specification>	Appendix A
<platform-definitions>, <Platform-Specification>	Appendix A
<profile>	6.6.2
<profile-note>	6.4.4.2
<question>	6.4.1
<rationale>	6.4.1
<rear-matter>	6.3.2
<reference>	6.2.6
<refine-rule>	6.5.2, 6.5.3
<refine-value>	6.2.3, 6.5.3
<remark>	Section 6 (several places)
<requires>	6.4.1
<result>	6.6.4.1, 6.6.4.2
<rule-result>	6.6.4.1

Release 0 (XCCDF 1.1.4 Final) or Release 1 (Initial Public Comment Draft)	Release 2 (Second Public Comment Draft)
<score>	6.6.2
<select>	6.5.2, 6.5.3
<set-complex-value>	6.5.2, 6.5.3
<set-value>	6.5.2, 6.5.3
<signature>	6.2.7
<source>	6.4.5.2
<status>	6.2.8
<sub>	6.2.9
<target>	6.6.2
<target-address>	6.6.2
<target-facts>	6.6.2
<title>	Section 6 (several places)
<upper-bound>	6.4.5.2
<value>	6.4.5.2
<version>	Section 6 (several places)
<warning>	6.4.1, 6.4.2
4.3 Handling Text and String Content	N/A
XHTML Formatting and Locale	7.2.3.6.1, 6.2.2, 6.2.10
String Substitution and Reference Processing	7.2.3.6.3, 7.2.3.6.4
5. Conclusions	N/A
Appendix A. XCCDF Schema	Removed from specification, posted separately as .xsd file
Appendix B. Sample Benchmark File	Dropped
Appendix C. Pre-Defined URIs	N/A
Long-Term Identification Systems	6.4.4.3
Check Systems	6.4.4.4
Scoring Models	7.3.3
Target Platform Facts	6.6.3
Remediation Systems	6.4.4.5
Appendix D. References	2
Appendix E. Acronym List	3.2

Release 3 – 29 September 2011

Editorial Changes:

- Minor editorial changes were made throughout the publication.
- An example of using the requires and conflicts elements was added.

Functional Additions:

- A use attribute was added to the sub element, to distinguish between replacement with values and titles during substitution processing.

Functional Changes:

- The requirements and recommendations related to limits on the use of metadata were clarified.

- The requirements for CPE version 2.3 names were changed; formatted string bindings are recommended for CPE names and applicability language expressions in XCCDF documents, but URI bindings may be used instead of formatted string bindings.
- Multiple instances of the dc-status element are allowed in all locations, instead of just one instance.
- The ident property can contain attributes from external namespaces.

Deprecations/Removals:

- The role attribute in Rules and rule-results, which had been deprecated in an earlier draft, was restored.

Release - November 2011

Editorial Changes:

- Typos were fixed in several tables correcting incorrect assertions about the presence or requirement of certain attributes and child elements
- Several element types that were previously defined inline were split out into their own global types, bringing them into alignment with prior conventions in the XCCDF schema.
- Annotations in the XCCDF schema were extensively revised and expanded to better reflect the information present in the revised specification

Deprecations/Removals:

- The profileIdKeyRef keyref constraint in the Benchmark element was removed. This keyref constraint required that profile references in a TestResult to match the id of a Profile in a Benchmark and was causing problems when the Profile existed only in a Tailoring file.
- The overridableIdrefType complex type was removed from the schema since it proved to be unreferenced.
- The import statement for the Dublin Core schema was removed as it was no longer necessary in the XCCDF schema.