

NISTIR 7275

Specification for the Extensible Configuration Checklist Description Format (XCCDF)

Version 1.1

Neal Ziring, Author,
National Security Agency

Timothy Grance, NIST Editor

Specification for the Extensible Configuration Checklist Description Format (XCCDF)

Neal Ziring, NSA Author
Information Assurance Directorate
National Security Agency
Fort Meade, MD 20755-6704

Timothy Grance, NIST Editor
Computer Security Division
Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20988-8930

Version 1.1
January 2006



U.S. DEPARTMENT OF COMMERCE

Carlos M. Gutierrez, Secretary

TECHNOLOGY ADMINISTRATION

Michelle O'Neill, Acting Under Secretary of Commerce for Technology

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY

William H. Jeffrey, 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
114 pages (January 2006)**

<p>Certain commercial entities, equipment, or materials may be identified in this document in order to describe an experimental procedure or concept adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the entities, materials, or equipment are necessarily the best available for the purpose.</p>

Abstract

This document specifies the data model and XML representation for the Extensible Configuration Checklist Description Format (XCCDF). 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 compliance scoring. The specification also defines a data model and format for storing results of benchmark compliance testing. The intent of XCCDF is to provide a uniform foundation for expression of security checklists, benchmarks, and other configuration guidance, and thereby foster more widespread application of good security practices.

Purpose and Scope

The Cyber Security Research and Development Act of 2002 tasks the National Institute of Standards and Technology (NIST) to “develop, and revise as necessary, a checklist setting forth settings and option selections that minimize the security risks associated with each computer hardware or software system that is, or is likely to become widely used within the Federal Government.” Such checklists, when combined with well-developed guidance, leveraged with high-quality security expertise, vendor product knowledge, operational experience, and accompanied with tools, can markedly reduce the vulnerability exposure of an organization.

To promote the use, standardization, and sharing of effective security checklists, NIST and NSA have collaborated with representatives of private industry to developed the XCCDF specification. The specification is vendor-neutral, flexible, and suited for a wide variety of checklist applications.

Audience

The primary audience of the XCCDF specification is government and industry security analysts, and industry security management product developers. NIST and NSA welcome feedback from these groups in improving the XCCDF specification.

Table of Contents

1.	Introduction.....	1
1.1.	Background.....	1
1.2.	Vision for Use.....	2
1.3.	Summary of Changes Since Version 1.0	3
2.	Requirements	4
2.1.	Structure and Tailoring Requirements.....	5
2.2.	Inheritance and Inclusion Requirements.....	6
2.3.	Document and Report Formatting Requirements	7
2.4.	Rule Checking Requirements	7
2.5.	Test Results Requirements.....	8
2.6.	Metadata and Security Requirements	8
3.	Data Model.....	9
3.1.	Benchmark Structure	10
3.2.	Object Detailed Contents	11
3.3.	Processing Models	27
4.	XML Representation.....	37
4.1.	XML Document General Considerations	37
4.2.	XML Element Dictionary	38
4.3.	Handling Text and String Content	65
5.	Conclusions.....	68
6.	Appendix A – XCCDF Schema.....	69
7.	Appendix B – Sample Benchmark File	98
8.	Appendix C: Pre-defined URIs.....	105
9.	References.....	107

Acknowledgements

The editor would like to acknowledge the following individuals who contributed to the initial definition of XCCDF and its initial development: David Proulx, Mike Michinikov, Andrew Buttner, Todd Wittbold, Adam Compton, George Jones, Chris Calabrese, John Banghart, Murugiah Souppaya, John Wack, Trent Pitsenbarger, and Robert Stafford. David Waltermire of the Center for Internet Security was instrumental in supporting the development of XCCDF; he contributed many important concepts and constructs, performed a great deal of proofreading on this specification document, and provided critical input based on implementation experience. Ryan Wilson of Georgia Institute of Technology also made substantial contributions. Thanks also go to the DISA FSO VMS/Gold Disk team for extensive review and many suggestions.

Trademark Information

Cisco and IOS are registered trademarks of Cisco Systems, Inc. in the USA and other countries. Windows is a registered trademark of Microsoft Corporation in the USA and other countries. Solaris is a registered trademark of Sun Microsystems, Inc. OVAL is a trademark of The MITRE Corporation.

Warnings

SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE EXPRESSLY DISCLAIMED. IN NO EVENT SHALL THE CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

1. Introduction

The security of an IT system may be measured in a variety of ways, but one way that has worked well in practice is conformance of the system configuration to a security benchmark. A typical benchmark includes criteria and rules for hardening a system against the most common forms of compromise and exploitation, and for reducing the exposed ‘attack surface’ of a system. Many different companies, government agencies, and community groups create and disseminate security benchmarks. While these various organizations often cooperate on the definition of the rules embodied in these consensus benchmarks, the underlying specification, test, and report formats used for these endeavors have been specialized and unique.

Configuring a system into conformance with a benchmark or other security specification is a highly technical task. To aid system administrators, commercial and community developers have created automated tools that can score a system’s conformance and recommend corrective measures. Many of these tools are data-driven: they accept a benchmark specification in some program-readable form, and use it to perform the checks and tests necessary to measure conformance and generate reports. However, with rare exceptions, none of these tools employ the same data formats, thus requiring duplication of effort and precluding interoperability.

This note describes a data model and processing discipline for supporting secure configuration and assessment. The requirements and goals are explained in detail below, but may be summarized briefly as document generation, expression of policy-aware configuration rules, support for complex and compound rules, support for compliance scoring, and support for customization and tailoring. The model and its XML representation are intended to be platform-independent and portable, to foster broad adoption and sharing of rules. The processing discipline of the format requires, for some uses, a service layer that can collect and store system information and perform simple policy-neutral tests against the system information. These conditions are described in detail below. The XML representation is expressed as an XML Schema in Appendix A.

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

1.1. Background

Today, groups promoting good security practices and system owners wishing to adopt them face an overload in the size and complexity of their tasks. As systems get larger, automated tools become a necessity for uniform application of security rules and visibility into system status. These conditions have created a need for mechanisms that:

- 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,

- permit scoring, reporting, and tracking of security status and checklist conformance, both over distributed systems and over the same systems across their operational lifetimes, and
- foster development of interoperable community and commercial tools for creating and employing security benchmarks.

Today, such mechanisms exist only in some isolated niche areas (e.g. MS Windows patch validation) and they support only narrow slices of security benchmark compliance functionality. This note proposes a data model and format specification for an extensible, interoperable benchmark ‘language’.

1.2. Vision for Use

XCCDF is designed to enable easier, more uniform creation of security benchmarks, and allow benchmarks to be used with a variety of commercial and open tools. The motivation for this is improvement of security for IT systems, including the Internet, by better application of known security practices and configuration settings.

The scenarios below illustrate some uses of security benchmarks and tools that XCCDF will foster.

- Scenario 1 –
An academic group produces a benchmark for secure configuration of a particular server operating system version. A government organization issues a set of rules extending the academic benchmark to meet more stringent user authorization criteria imposed by statute. A medical enterprise downloads both the academic benchmark and the government extension, tailors the combination to fit their internal security policy, and applies an enterprise-wide audit using a commercial security audit tool. Reports output by the tool include remediative measures which the medical enterprise IT staff can use to bring their systems into full internal policy compliance.
- Scenario 2 –
A federally-funded lab issues a security advisory about a new Internet worm. In addition to a prose description of the worm’s attack vector, they include a set of short benchmarks in a standard format that assess vulnerability to the worm for various operating system platforms. Organizations all over the world pick up the advisory, and use installed tools that support the standard format to check their status and fix vulnerable systems.
- Scenario 3 –
An industry consortium wants to produce a security checklist for a popular commercial 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 can craft one checklist in a standard format for the core settings, and then write several OS-specific ones that incorporate the core settings by reference. Users download the core checklist and the OS-specific checklists that apply to their installations, and run a checking tool to score their compliance with the checklist.

1.3. Summary of Changes since Version 1.0

XCCDF 1.0 received some review and critique after its release in January 2005. Most of the additions and changes in 1.1 come directly from suggestions by users and potential users. The list below gives the major changes; other differences are noted in the text.

- Persistent/standard identifiers - To foster standardization and re-use of XCCDF rules, community members suggested that Rule objects bear long-term, globally unique identifiers. Support for identifiers, along with the scheme or organizations which assigns them, is now part of the Rule object.
- Versioning - To foster re-use of XCCDF rules, and to allow more precise tracking of benchmark results over time, Benchmarks, Rules, and Profiles all support a version number. The version number now supports a timestamp, too.
- Severity - Rules can not support a severity level: info, low, medium, and high. Severity levels can be adjusted via Profiles.
- Signatures - Each object that can be a standalone XCCDF documents can have an XML digital signature: Benchmark, Group, Rule, Value, Profile, and TestResult.
- Rule result enhancements - Added the override property for rule-result members of the TestResult object. Clarified the use of different rule result status values in scoring, and added several new rule result status values. Added better instance details, for multiply-instantiated rules.
- Enhancements for remediation - Added several minor enhancements to the Rule's properties for automated and interactive remediation (fix and fixtext elements).
- Interactive Value tailoring - Added the 'interactive' property to Value objects; it gives a benchmark checking tool a hint that it should solicit a new value prior to each application of the benchmark. Added the 'interfaceHint' property to allow the author to suggest a UI model to the tool.
- Scoring models - Added the notion of multiple scoring models and described two new models. Added the model and param elements, expanded the score element.
- Re-usable plain text blocks - Added named, re-usable text blocks for benchmarks.
- Richer XHTML references - Defined mechanisms for using XHTML "object" and "a" tags to reference other XCCDF objects within a generated document.
- Target facts - Added the target facts list to the TestResult object, to allow an XCCDF document to store arbitrary facts about target systems.
- Complex checks – Added the ability to compose boolean expressions from multiple individual checks in Rules.
- Extension control – added the 'override' attribute to most property element children that can appear more than once in a Rule, Group, Value, or Profile.
- Added the 'source' property to the Value object, to allow a benchmark author to indicate (as URIs) possible ways to obtain correct or candidate values.
- Added a descriptive note facility for relating Rules and Profiles.

2. Requirements

The general objective for XCCDF is to allow security analysts and IT experts to create effective and inter-operable benchmarks, and to support use of benchmarks with a wide variety of tools. Figure 1 shows some purposes for which a benchmark might be used.

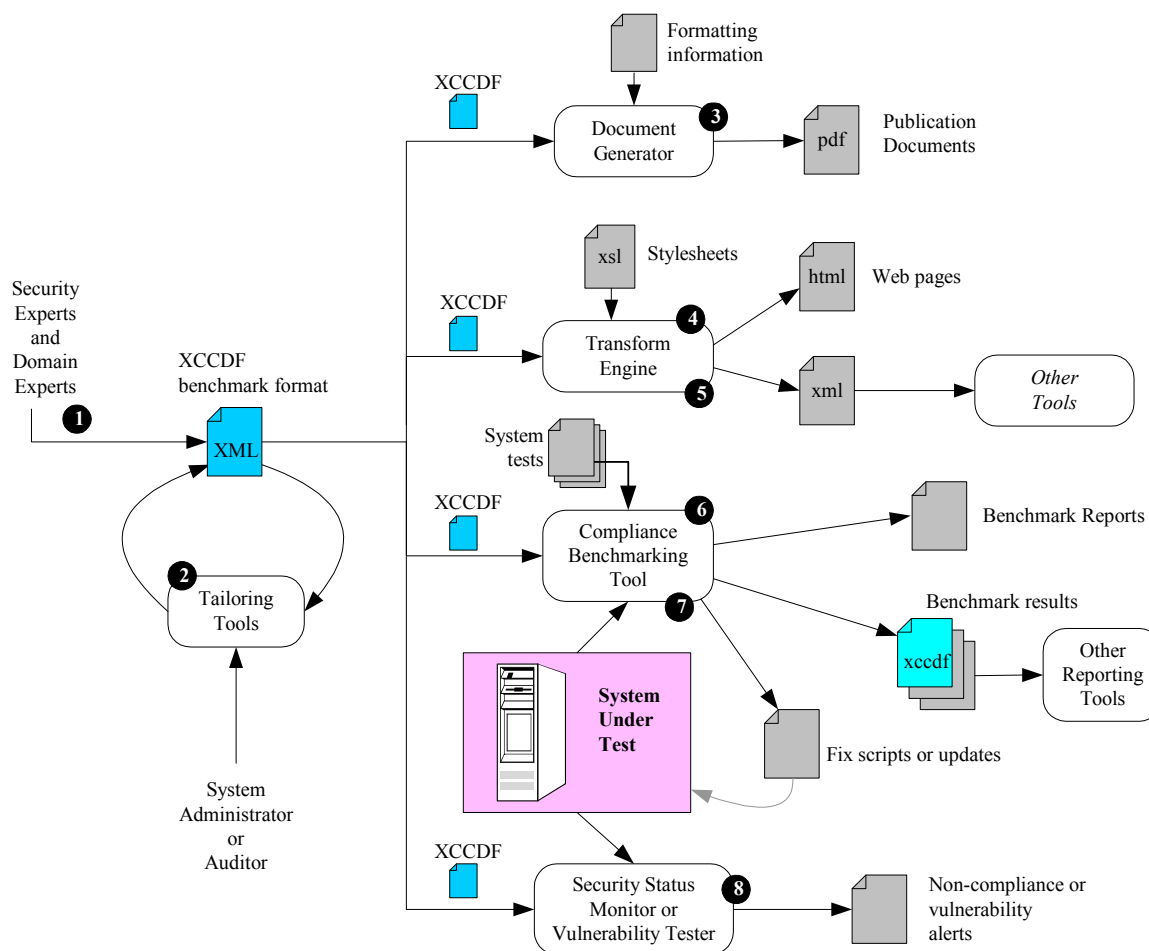


Figure 1 – Use Cases for XCCDF Documents

The list below describes some requirements for each of the uses.

1. Security and domain experts create a benchmark, which is an organized collection of rules about a particular kind of system or platform. To support this use, XCCDF must be an open, standardized format, amenable to generation and editing with a variety of tools. It must be expressive enough to represent complex conditions and relationships about the systems to be benchmarked, and it must also be able to incorporate descriptive material and remediative measures. (XCCDF benchmarks may include specification of the hardware and/or software platforms to which they apply. The specification should be concrete and granular enough for compliance checking tools to detect whether a rule is suited for a target platform.)

2. Auditors and system administrators may employ tailoring tools to customize a benchmark for their local environment or policies. An XCCDF document must include the structure and interrogative text needed to guide the user in tailoring a benchmark, and it must be able to hold or incorporate the user's tailoring responses.
3. In addition to supporting tailoring and security audits, an XCCDF document should be structured to foster generation of hardcopy benchmark guides.
4. The structure of a XCCDF document should support transformation into HTML, for posting the benchmark as a web page.
5. An XCCDF document should be transformable into (other) XML formats, to promote portability and interoperability.
6. The primary use for an XCCDF benchmark is to drive automated security benchmarking tools. Such tools should accept one or more XCCDF documents, and supporting system test definitions, and check whether their rules are satisfied by some particular target system. The XCCDF document should support generation of a compliance report, including a weighted compliance score.
7. In addition to a benchmark report, some benchmarking tools may be capable of generating scripts or procedures for helping to bring a system into compliance. XCCDF must be able to hold or encapsulate the remediation scripts or texts, including several alternatives.
8. XCCDF documents might also be used in vulnerability scanners, to test whether a target system is vulnerable to a particular kind of attack. For this purpose, the XCCDF document would play the role of a vulnerability alert, but with the ability to both describe the problem and drive automated verification of its presence.

In addition to these use cases, an XCCDF document should be amenable to embedding inside other documents, and to having data expressed in other formats embedded inside of it. Also, as its name implies, XCCDF must be extensible – it must be possible for new functionality and features to be added to XCCDF-capable tools and data for those new features stored in XCCDF without breaking other tools.

2.1. Structure and Tailoring Requirements

To support tailoring by users, and generation of documents for users, XCCDF must allow authors to impose organization on a benchmark. Benchmark authors will need to arrange rules in order, and collect them into groups.

For benchmark structure, a benchmark author must be able to designate the order in which rules or groups are to be processed. As the simplest case, processing order can be simply the order in which the rules appear in the XCCDF document.

For tailoring, values, rules and groups will need descriptive and interrogative text to help a user make tailoring decisions. Two basic kinds of tailoring will be needed:

- Selectability – a tailoring action might select or deselect a rule or group of rules for inclusion or exclusion from the benchmark. For example, at a site where no FTP service is used, an auditor might choose to deselect all rules about secure configuration of the FTP server.

- Substitution – a tailoring action might substitute a locally-significant value for a general value in a rule. For example, at a site where all logs are sent to a designated logging host, the address of that log server might be substituted into a rule about audit configuration.

Once benchmarks can be tailored, the possibility arises that some rules within the same benchmark might conflict or be mutually exclusive. In other words, the author of a benchmark must be able to identify particular tailoring choices as incompatible, so that tailoring tools can take appropriate actions.

In addition to being able to specify rules, XCCDF must support structures that foster use and re-use of rules. To this end, XCCDF must provide a means for related rules to be grouped together, and for sets of rules and groups which should be applied in concert to be designated, named, and applied easily. Two realizations of this notion are benchmark levels, as provided in benchmarks distributed by the Center for Internet Security, and checklist baselines, as described in the NIST security checklist program [11].

For benchmark processing, there are two basic processing modes: rule checking, and document generation. It must be possible for a benchmark author to designate the modes under which a rule may be processed.

2.2. *Inheritance and Inclusion Requirements*

To support building up benchmarks from parts, XCCDF must support mechanisms for authors to extend (inherit from) existing rules and rule groups, in addition to expressing rules and groups in their entirety. Also, it must be possible for one benchmark to include all or part of another. There are several benchmarking use cases where inheritance and inclusion will be needed.

- An organization might choose to define a foundational benchmark for a family of platforms (e.g. Unix-like operating systems) and then extend them for specific members of the family (e.g. Solaris) or for specific roles (e.g. mail server).
- An analyst might choose to make an extended version of a benchmark, by adding some new rules and adjusting some others.
- If the sets of rules that constitute a benchmark come from several sources, it will be useful to be able to aggregate them using an inclusion mechanism.
- Within a benchmark, it might be desirable to share some of the descriptive material among several rules. With extension, this can be accomplished by creating a base rule, and then extending it with several different rule checks.
- For updating a benchmark, it will be convenient to be able to incorporate changes or additions using extension.
- To allow broader site-specific or enterprise-specific customization, it should be possible for a user to override or amend any portion of a benchmark rule.

The XCCDF specification does not include any mechanism for inclusion; instead, implementations of XCCDF tools should support the XML Inclusion (XInclude) facility standardized by the W3C [9].

2.3. Document and Report Formatting Requirements

Several of the main use cases for XCCDF benchmarks involve generation of reports or other documents for users to read. Authors will need mechanisms for formatting text, including images, and referencing other information resources. These mechanisms must be separable from the text itself, so that they can be filtered out by applications that do not support them. (XCCDF 1.1 currently satisfies these formatting requirements mainly by allowing inclusion of XHTML markup tags [3].)

For document formatting, a benchmark must be able to include arbitrary document text that does not contribute directly to the benchmarking process: introduction, rationale, warnings, and references are just some of the uses for extra text. Further, the text must be able to include intra-document and external references and links.

2.4. Rule Checking Requirements

The primary use for XCCDF will be performing security and operational checks on systems. Therefore, XCCDF must have access to very fine-grained and expressive mechanisms for checking the state of a system against rule criteria. The community seems to have reached an informal consensus that the model for this is to treat the state or configuration of a system as a collection of facts, and to treat expression of conditions and criteria as an operation or combination of operations against the collection. The operations used have varied with different existing applications; some rule checking systems use a database query operation model, while others use a pattern-matching model. At the least, any rule checking mechanism used for XCCDF must satisfy the following criteria:

- It must be able to express both positive and negative criteria -
A positive criterion means that if certain conditions are met, then the system satisfies the benchmark, while a negative criterion means if the conditions are met the system fails the benchmark. Experience has shown that both kinds are necessary when crafting security benchmarks.
- It must be able to 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.
- It must be able to incorporate tailoring values set by the user -
As described above, substitution is important for benchmark tailoring. Any XCCDF checking mechanism must support substitution of tailored values into its criteria or statements as well as tailoring of the selected set of rules.

It is not clear that a single rule specification scheme can be defined that will satisfy all uses of XCCDF. Therefore, the XCCDF definition must allow for use of different rule checking systems, and a means for identifying the checking system used in each rule. It is important that the rule checking system be defined separately from XCCDF itself, so that they can evolve separately and be used independently when necessary. This further implies the need to cleanly define the interface between XCCDF and the rule checking system, in terms of information passed from each to the other.

2.5. Test Results Requirements

A primary goal for XCCDF is to drive automated security testing and benchmark compliance checking tools. While an XCCDF benchmark document may comprise a main input to such a tool, standardized output is also important. XCCDF must provide a means for storing the results of compliance tests. Some of the information that would need to be stored is listed below.

- The benchmark used, along with any tailoring applied.
- Information about the target system to which to 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 compliance scores.
- References to lower-level details possibly stored in other output files.

2.6. Metadata and Security Requirements

Security benchmarks are fairly common, and some government and volunteer organizations have disclosed plans to create repositories of benchmarks. To facilitate discovery and retrieval of benchmarks in repositories and on the open Internet, XCCDF must support inclusion of metadata about a benchmark. Some of the metadata that must be supported includes: benchmark title, name of benchmark author(s), organization providing the benchmark, version number, release date, update URL, and a description.

A number of metadata standards already exist, it is preferable that XCCDF simply incorporate one of them rather than defining its own metadata model.

Application of a security benchmark is a very sensitive action in the management of an IT system. Therefore, some users may need to verify the integrity and provenance of a benchmark before using it. Also, if system vendors or government agencies define Rules and Groups, they may want to ensure the integrity and authenticity of the information they provide. Digital signatures are the natural mechanism to satisfy these integrity and proof-of-origin requirements. Fortunately, mature standards for digital signatures already exist that are suitable for asserting the authorship and protecting the integrity of benchmarks. XCCDF must provide a means to hold such signatures, and a uniform method for applying and validating them.

3. Data Model

The fundamental data model for XCCDF consists of four main object data types:

1. **Benchmark** –
An XCCDF document holds exactly one Benchmark object. A Benchmark holds descriptive text, and acts as a container for items and other objects.
2. **Item** –
An Item is a named constituent of a Benchmark; it has properties for descriptive text, and can be referenced by an id. There are several derived classes of Items.
 - **Group** –
This kind of Item can hold other Items. If a Group is unselected, then all of the items it contains are unselected. A Group may be selected or unselected.
 - **Rule** –
This kind of Item holds a rule checking definition, a scoring weight, and may also hold remediation text. A Rule may be selected or unselected.
 - **Value** –
This kind of Item is a named data value that can be substituted into other Item's property values. It can also have an associated data type and operator that expresses how the value should be used and how it can be tailored.
3. **Profile** –
A profile is a collection of attributed references to Rule, Group, and Value objects. It supports the requirement to allow definition of named levels or baselines in a benchmark (see Section 2.1).
4. **TestResult** –
A test result object holds the results of performing a compliance test against a single target device or system.

Figure 2, below, shows the data model relationships as a UML diagram.

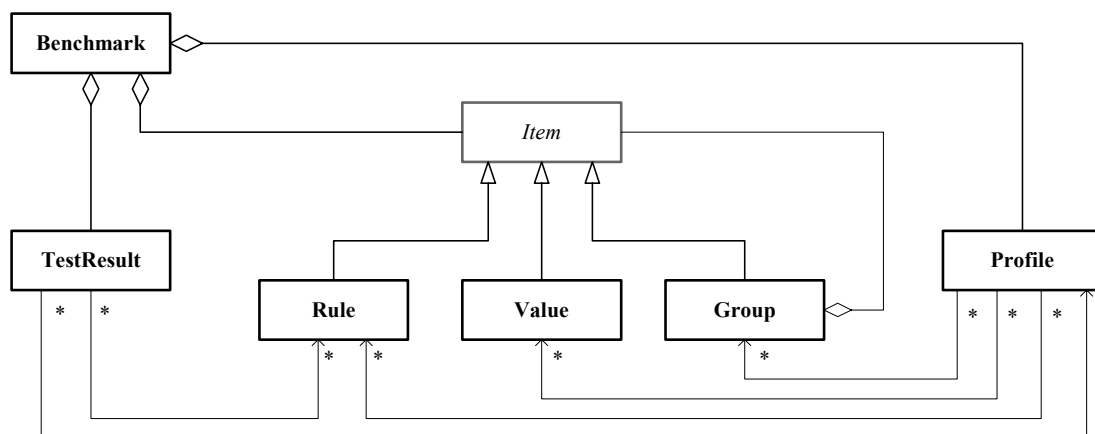


Figure 2 –XCCDF High-Level Data Model

As shown in Figure 2, one Benchmark can hold many Items, but each Item belongs to exactly one Benchmark. Similarly, a Group can hold many Items, but an Item may belong to only one Group. Thus, the Items in a benchmark form a tree, where the root node is the Benchmark, interior nodes are Groups, and the leaves are Values and Rules.

A Profile object references Rule, Value, and Group objects. A TestResult object references Rule objects, and may also reference a Profile object.

The definition of a Value, Rule, or Group can extend another Value, Rule, or Group. The extending item inherits property values from the extended item. This extension mechanism is separate and independent of grouping.

Group and Rule items can be marked by a benchmark author as selected or unselected. A Group or Rule that is not selected does not undergo processing. The author may also stipulate, for a Group, Rule, or Value, whether the end user is permitted to tailor it.

Rule items may have a scoring weight associated with them, which can be used by a benchmark checking tool to compute a target system's overall compliance score. Rule items may also hold remediation information.

Value items include information about current, default, and permissible values for the Value. Each of these properties of a Value can have an associated selector id, which is used when customizing the Value as part of a Profile. For example, a Value might be used to hold a Benchmark's lower limit for password length on some operating system. In a Profile for that operating system to be used in a closed lab, the default value might be 5, but in a Profile for that operating system to be used on the Internet, the default value might be 10.

3.1. Benchmark Structure

Typically, a Benchmark would hold one or more Groups, and each group would hold some Rules, Values, and additional child Groups. Figure 3 illustrates this relationship, and the order in which the contents of a Benchmark must appear.

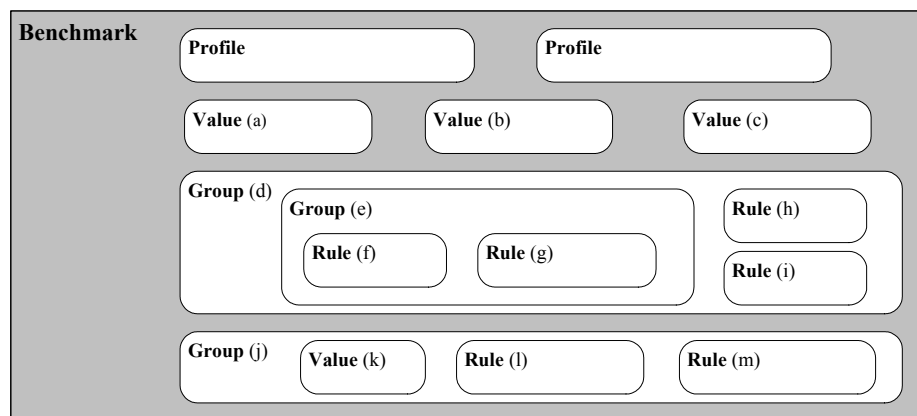


Figure 3 – Typical Structure of a Benchmark

Groups allow a benchmark author to collect related rules and values into a common structure and provide descriptive text and references about them. Further, groups allow benchmark users to select and deselect related rules together, helping to ensure commonality among users of the same benchmark. Lastly, groups affect benchmark

compliance scoring. As explained in Section 3.3, an XCCDF compliance score is calculated for each group, based on the rules and sub-groups in it. The overall XCCDF score for the benchmark is computed only from the scores on the immediate Group and Rule children of the Benchmark object. In the tiny benchmark shown in Figure 3, the benchmark score would be computed from the scores of Group (d) and Group (j). The score for Group (j) would be computed from Rule (l) and Rule (m).

Inheritance

The possible inheritance relations between Item object instances are constrained by the tree structure of the benchmark, but are otherwise independent of it. In other words, all extension relationships must be resolved before the benchmark can be used for compliance testing. An Item may only extend another Item of the same type that is ‘visible’ from its scope. An Item X is visible from another Item Y if and only if it meets one of the following conditions: (1) X and Y are siblings, (2) X is a sibling of some enclosing group of Y, or (3) X is visible to the scope of the direct children of any group extended by any enclosing group of Y.

For example, in the tiny benchmark structure shown in Figure 3, it would be legal for Rule (g) to extend Rule (f), and legal for Rule (f) to 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 Group (g), because they are not of the same type.

The ability for a Rule or Benchmark to be extended by another is a convenience for benchmark authors.

3.2. Object Detailed Contents

The tables below show the properties that make up each data type in the XCCDF data model. Note that the properties that comprise a Benchmark or Item are an ordered sequence of property values, and the order in which they appear determines the order in which they are processed.

Properties with a data type of “text” are string data that can include embedded formatting directives and hypertext links. Properties of type “string” may not include formatting.

Benchmark

Property	Type	Count	Description
id	identifier	1	benchmark identifier, mandatory
title	text	0-n	title of the benchmark document
description	text	0-n	text that describes the benchmark
version	string	1	version number of the benchmark
status	string+date	1-n	status of the benchmark (see below) and date at which it attained that status, mandatory.
resolved	boolean	0-1	True if benchmark has already undergone the resolution process (see Section 3.3)

Property	Type	Count	Description
notice	text	0-n	legal notices or copyright statements about this benchmark; each notice has a unique identifier and text value.
front-matter	text	0-n	text for the front of the benchmark document
rear-matter	text	0-n	text for the back of the benchmark document
reference	<i>special</i>	0-n	A bibliographic reference for the benchmark document: metadata or a simple string, plus an optional URL.
platform-definitions	<i>special</i>	0-1	A list of component definitions and platform definitions, each with an id.
platform	id	0-n	target platform(s) for this benchmark; this property may appear multiple times. The id refers to a platform definition.
plain-text	string+id	0-n	Reusable text blocks, each with a unique id. These can be included into other text blocks in the benchmark.
model	URI	0-n	Suggested scoring model or models to be used when computing a compliance score for this benchmark; optional.
profiles	Profile	0-n	Profiles that reference and customize sets of items in the Benchmark; optional
values	Value	0-n	Tailoring values that support rules and descriptions in the benchmark
groups	Group	0-n	Groups that comprise the benchmark, each group may contain additional values, groups, and rules.
rules	Rule	0-n	Rules that comprise the benchmark
test-results	TestResult	0-n	Benchmark test result records (one per test); optional
metadata	<i>special</i>	0-n	Discovery metadata for the benchmark (e.g. compliant with Dublin Core Metadata Initiative XML guidelines)
signature	<i>special</i>	0-1	Digital signatures asserting authorship and allowing verification of the integrity of the benchmark, optional

Conceptually, a Benchmark contains Group, Rule, and Value objects, and it may also contain Profile and TestResult objects. For ease of reading and simplicity of scoping, all Value objects must precede all Groups and Rules, which must precede all Profiles, which must precede all TestResults. These objects may be directly embedded in the Benchmark, or incorporated via W3C standard XML Inclusion [9].

The platform-definitions property consists of platform descriptions, each with a unique id. Benchmark, Group, Rule, and Profile objects may have platform properties that

identify the hardware and/or software products to which they apply. The Benchmark platform-definitions and platform properties are optional. Benchmark authors should use them to identify the systems or products to which their benchmarks apply. For more information about platform specification, consult [12].

Each status property consists of a status string and a date. Permissible string values are “accepted”, “draft”, “interim”, “incomplete”, and “deprecated”. Benchmark authors should mark their benchmarks with a status to indicate a level of maturity or consensus. A Benchmark may contain one or more status property values, for different status values.

Benchmark metadata allows authorship, publisher, support, and other information to be embedding a benchmark. Metadata should comply with existing commercial or government metadata specifications, to allow benchmarks to be discovered and indexed. The XCCDF data model allows multiple metadata properties for a Benchmark; each property should provide metadata compliant with a different specification. The primary metadata format, which should appear in all published benchmarks, is the simple Dublin Core Elements specification, as documented in [13].

The plain-text properties, new in XCCDF 1.1, allow commonly used text to be defined once and then re-used in multiple text blocks in the benchmark. Note that each plain-text must have a unique id, and that the ids of other Items and plain-text properties must not collide. [This restriction is imposed to permit easier implementation of document generation and reporting tools.]

Note that a digital signature, if any, applies only to the Object in which it appears, but after inclusion processing (note: it may be impractical to use inclusion and signatures together). Any digital signature format employed for XCCDF benchmarks 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 content of the benchmark. XCCDF tools that support signatures at all must support the W3C XML-Signature standard enveloped signatures [8].

Legal notice text is handled specially, as discussed in Section 3.3.

Item (abstract)

Property	Type	Count	Description
id	identifier	1	unique object identifier, mandatory
title	text	0-n	title of the Item (for human readers)
description	text	0-n	text that describes the Item
warning	text	0-n	a cautionary note or caveat about the Item
status	string+date	0-n	status of the item and date at which it attained that status (optional)
version	string+date	0-1	version number of this item, and update URI (optional)
question	string	0-n	interrogative text to present to the user during tailoring (optional)

Property	Type	Count	Description
cluster-id	identifier	0-1	an identifier to be used from a Profile to refer to multiple Groups and Rules, optional
hidden	boolean	0-1	whether this Item should be excluded from any generated documents (default: false)
prohibitChanges	boolean	0-1	whether tools should prohibit changes to this item during tailoring (default: false)
abstract	boolean	0-1	if true, then this Item is abstract and exists only to be extended (default: false)
reference	<i>special</i>	0-n	a reference to a document or resource where the user can learn more about the subject of this Item: content is Dublin Core metadata or a simple string, plus an optional URL
signature	<i>special</i>	0-1	digital signature over this Item, optional

There are several Item properties that give the benchmark author control over how items may be tailored and presented in documents. First, the hidden property simply prevents an Item from appearing in generated documents. For example, an author might set the hidden property on incomplete items in a draft benchmark. The prohibitChanges property advises tailoring tools that the benchmark author does not wish to allow end users to change anything about the Item. Lastly, a value of true for the abstract property denotes an item intended only for other items to extend. In many all cases, abstract items should also be hidden.

The cluster-id property is optional, but it provides a means to identify related Value, Group and Rule items throughout the Benchmark. Cluster-id values do not need to be unique: all the Items with the same cluster-id value belong to the same cluster. A selector in a Profile can refer to a cluster, thus making it easier for authors to create and maintain profiles in a complex benchmark. The cluster-id property is not inherited (see page 30).

Every Item may include one or more status properties. Each status property value represents a status that the Item has reached and the date at which it reached that status. Benchmark authors can use status elements to record the maturity or consensus level for Rules, Groups, and Values in the Benchmark. If an Item does not have an explicit status property value given, then its status is taken to be that of the Benchmark itself. The status property is not inherited (see page 30).

Group :: Item

Property	Type	Count	Description
requires	identifier	0-n	the id of another Group or Rule in the benchmark that must be selected for this Group to be applied and scored properly
conflicts	identifier	0-n	the id of another Group or Rule in the benchmark that must be unselected for this Group to be applied and scored properly

Property	Type	Count	Description
selected	boolean	1	whether this Group is currently selected for processing, default is true. This property may be overridden by a Profile.
rationale	text	0-n	descriptive text giving rationale or motivations for abiding by this Group
platform	identifier	0-n	A platform to which this Group applies, a reference to a platform-definition (see [12])
cluster-id	identifier	0-1	an identifier to be used from benchmark profiles to refer to multiple Groups and Rules, optional
extends	identifier	0-1	id of a Group on which to base this Group, optional
weight	float	0-1	the relative scoring weight of this Group, for computing a compliance score
values	Value	0-n	Values that belong to this Group, optional
groups	Group	0-n	Sub-groups under this Group, optional
rules	Rule	0-n	Rules that belong to this Group, optional

A Group can be based on (extend) another Group. This means that the extending Group includes all the Items of the extended Group, plus any defined inside the extending Group. Other properties behave differently, depending on their allowed count. For any property that is allowed to appear more than once, the extending Group gets the sequence of property values from the extended group, plus any of its own values for that property. For any property that is allowed to appear at most once, the extending Group gets its own value for the property if one appears, otherwise it gets the extended Group's value of that property. Items that belong to an extended group are treated specially: the id property of any Item copied as part of an extended group must be replaced with a new, uniquely generated id. A Group for which the abstract property is true exists only to be extended by other Groups, it should never appear in a generated document and none of the Rules defined in it should be checked in a compliance test. Abstract Group objects are removed during resolution, for more information see Section 3.3.

To give the benchmark author more control over inheritance for extending Groups (and other XCCDF objects), all textual properties that may appear more than once can bear an override attribute. For more information about inheritance overrides and extension, see Section 3.3.

The platform property of a Group indicates that the Group contains platform-specific items that apply to some set of (usually related) platforms. First, if a Group does not possess any platform properties, then it applies to the same set of platforms as its enclosing Group or the Benchmark. Second, for tools that perform compliance checking on a platform, any Group whose set of platform property values do not include the platform on which the compliance check is being performed should be treated as if their selected property were set to false. Third, any platform property value that appears on a Group should be a member of the set of platform property values of the enclosing

Benchmark. Last, if no platform properties appear anywhere on a Group or its enclosing Group or Benchmark, then the Group applies to all platforms.

The weight property denotes the importance of a Group relative to its sibling in the same Group or its siblings in the Benchmark (for a Rule that is a child of the Benchmark). Scoring is computed independently for each collection of sibling Groups and Rules, then normalized as part of the overall scoring process. For more information about scoring, see Section 3.3.

The requires and conflicts properties provide a means for benchmark authors to express dependencies among Rules and Groups. Their exact meaning depends on what sort of processing the benchmark is undergoing, but in general the following approach should be applied: if a Rule or Group is about to be processed, and any of the Rules or Groups identified in a requires property have a selected property value of false or any of the Items identified in a conflicts property have a selected property value of true, then processing for the item should be skipped and its selected property should be set to false.

Rule :: Item

Property	Type	Count	Description
selected	boolean	1	If true, this Rule is selected to be checked as part of the benchmark when the benchmark is applied to a target system; an unselected rule is not get checked and does not contribute to scoring. Can be overridden by a Profile.
extends	id	0-1	The id of a Rule on which to base this Rule (must match the id of another Rule)
multiple	boolean	0-1	Whether this rule should be multiple instantiated. If false, then benchmark tools should avoid multiply instantiating this Rule.
role	string	0-1	Rule's role in scoring and reporting; one of the following: "full", "unscored", "unchecked". Can be overridden by a Profile.
severity	string	0-1	Severity level code, to be used for metrics & tracking. One of the following: "unknown" (default), "info", "low", "medium", "high". Can be overridden by a Profile.
weight	float	0-1	The relative scoring weight of this Rule, for computing a compliance score. Can be overridden by a Profile.
rationale	text	0-n	Some descriptive text giving rationale or motivations for complying with this Rule.
platform	identifier	0-n	A platform to which this Rule applies, a reference to a platform-definition (see [12]).
requires	identifier	0-n	The id of another Group or Rule in the benchmark that should be selected for this Rule to be applied and scored properly.

Property	Type	Count	Description
conflicts	identifier	0-n	The id of another Group or Rule in the benchmark that should be unselected for this Rule to applied and scored properly.
ident	string+URI	0-n	A long-term, globally meaningful name 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 denotes the organization or naming scheme which assigns the name.
profile-note	text + id	0-n	Descriptive text related to a particular Profile. This property allows a benchmark author to describe special aspects of the Rule related to one or more Profiles. It has an id that can be specified as part of a Profile.
fixtext	text+attrs	0-n	Prose that describes how to fix the problem of non-compliance with this rule. Each fixtext property may be associated with one or more fix property values.
fix	text+attrs	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.
check	<i>special</i>	0-n	The definition of, or a reference to, the target system check needed to test compliance with this Rule. A check consists of three parts: the checking system specification on which it is based, a list of Value objects to export, and the content of the check itself. If a Rule has several check properties, each must employ a different checking system.
complex-check	<i>special</i>	0-1	A complex check is a boolean expression of other checks. A most one complex-check may appear in a Rule. (see below)

A Rule can be based on (extend) another Rule. This means that the extending Rule inherits all the properties of the extended or base Rule, some of which it might override with new values. For any property that is allowed to appear more than once, the extending Rule gets the sequence of property values from the extended group, plus any of its own values for that property. For any property that is allowed to appear at most once, the extending Rule gets its own value for the property if one appears, otherwise it gets the extended Rule's value of that property. A Rule for which the abstract property is true should not be included in any generated document, nor should it be checked in any compliance test. Abstract rules are removed during resolution (see Section 3.3).

The weight property denotes the importance of a rule relative to its sibling in the same Group or its siblings in the Benchmark (for a Rule that is a child of the Benchmark). For more information about scoring, see Section 3.3.

Each ident property represents a binding to a globally meaningful name in some security domain; the string value of the property is the name, and a URI designates the scheme or organization that assigned the name. By giving the ident value, the benchmark author effectively declares that the Rule instantiates, implements, or remediates the issue for which the name was assigned. For example, the ident value might be a CVE identifier; the Rule would 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 web site.

The role property gives the benchmark author additional control over Rule processing during application of a benchmark. The default (“full”) means that the Rule is checked, contributes to scoring according to the scoring model, and appears in any output reports. The “unscored” role means that the Rule is checked, and appears in any output reports, but does not contribute to score computations. The “unchecked” role means that the Rule does not get checked, its rule result status is set to unknown, and it does not contribute to scoring, but it can appear in output reports.

The multiple property provides direction about multiple instantiation to a processing tool applying the Rule. By setting multiple to true, the Rule’s author is directing that separate components of the target to which the Rule can apply should be tested separately and the results recorded separately. By setting multiple to false, the author is directing test results of such components be combined. If the processing tool cannot perform multiple instantiation, or if multiple instantiation of the Rule is not applicable for the target system, then processing tools may ignore this property.

The platform properties of a Rule indicate the platforms to which the Rule applies. First, if a Rule does not possess any platform properties, then it applies to the same set of platforms as its enclosing Group or Benchmark. Second, for tools that perform compliance checking on a platform, any Rule whose set of platform property values do not include the platform on which the compliance check is being performed should be treated as if their selected property were set to false. Third, any platform property value that appears on a Rule should be a member of the set of platform property values of the enclosing Benchmark. Last, if no platform properties appear anywhere on a Rule or its enclosing Group or Benchmark, then the Rule applies to all platforms.

The check property consists of the following: a selector for use with Profiles, a URI that designates the checking system or engine, a set of export declarations, and the check content. The checking system URI tells a compliance checking tool what processing engine it must use to interpret or execute the check. The nominal or expected checking system is MITRE’s OVAL system (designated by <http://oval.mitre.org/>), but the XCCDF data model allows for alternative or additional checking systems. XCCDF also supports conveyance of tailoring values from the XCCDF processing environment down to the checking system, which is the purpose of the export declarations. Each export declaration maps an XCCDF Value object id to an external name or id for use by the checking system. The check content is an expression or document in the language of the checking system; it may appear inside the XCCDF document (an *enveloped* check) or it may appear as a reference (a *detached* check).

In place of a check property, XCCDF 1.1 allows a complex-check property. A complex check is a boolean expression whose individual terms are checks or complex-checks. This allows benchmark authors to re-use checks in more flexible ways, and to mix checks

written with different checking systems. At most one complex check may appear in a Rule; on inheritance, the extending Rule's complex-check replaces the extended Rule's complex-check. If both check properties and a complex-check property appear in a Rule, then the check properties must be ignored. The following operators are allowed for combining the constituents of a complex-check:

- **AND** – only if all terms must evaluate to Pass (true) then the complex-check evaluates to Pass.
- **OR** – if any term evaluates to Pass then the complex-check evaluates to Pass.

Truth-tables for the operators appear under their detailed descriptions in the next section. Note that each complex-check may also specify that the expression should be negated (boolean **not**).

The properties `fixtext` and `fix` exist to allow a benchmark author to specify a way to remediate non-compliance with a Rule. The `fixtext` property provides a prose description of the fix that needs to be made; in some cases this may be all that is possible to do in the benchmark (e.g. if the fix requires manipulation of a GUI, or installation of additional software). The `fix` property provides a direct means of changing the system configuration to accomplish the necessary change (e.g. a sequence of command-line commands, or a set of lines in a system scripting language like Bourne shell, or in a system configuration language like Windows INF format, or as a list of update or patch ID numbers).

The `fix` and `fixtext` properties are enhanced for XCCDF 1.1, to help tools support more sophisticated facilities for automated and interactive remediation of benchmark findings. The following attributes can be associated with a `fix` or `fixtext` property value:

- `strategy` – a keyword that denotes the method or approach for fixing the problem. This applies to both `fix` and `fixtext`. Permitted values: `unknown` (default), `configure`, `combination`, `disable`, `enable`, `patch`, `policy`, `restrict`, `update`.
- `disruption` – an estimate for how much disruption the application of this fix will impose on the target. This applies to `fix` and `fixtext`. Permitted values: `unknown`, `low`, `medium`, `high`.
- `reboot` – whether remediation will require a reboot or hard-reset of the target. This applies to `fix` and `fixtext`. Permitted values: `true` (1) and `false` (0).
- `system` – a URI representing the scheme, language, engine, or process for which the fix contents are written. XCCDF 1.1 will define several general-purpose URNs for this, but it is expected that tool vendors and system providers may need to define target-specific ones. This applies to `fix` only.
- `id/fixref` – these attributes will allow `fixtext` properties to be associated with specific `fix` properties (pair up explanatory text with specific `fix` procedures).
- `platform` – in case different `fix` scripts or procedures are required for different target platform types (e.g. different patches for Windows 2000 and Windows XP) this attribute allows a platform identifier to be associated with a `fix` property.

For more information, consult the detailed definition of the `fix` and `fixtext` elements in Section 4.2.

Value :: Item

Property	Type	Count	Description
value	string+id	1-n	the current value of this Value
default	string+id	0-n	default value of this Value object, optional
choices	list + id	0-n	a list of legal or suggested values for this Value object, to be used during tailoring and document generation, optional
type	identifier	0-1	the data type of the value: string, number, or boolean (default: string)
lower-bound	number+id	0-n	minimum legal value for this Value (applies only if type is 'number')
upper-bound	number+id	0-n	maximum legal value for this Value (applies only if type is 'number')
operator	string	0-1	the operator to be used for comparing this Value to some part of the test system's configuration (see list below).
match	regular expr.	0-n	a regular expression, which the value must match to be legal, optional (for more information, see [7])
interactive	boolean	0-1	tailoring for this Value should also be performed during Benchmark application, optional (default is false)
interfaceHint	identifier	0-1	user interface recommendation for tailoring, optional
source	URI	0-n	URI indicating where the benchmarking tool may acquire a value for this Value object, optional (may be the URI of a platform Fact, see [16])
extends	identifier	0-1	id of a Value on which to base this Value, optional

A Value is content that can be substituted into properties of other items, including the interior of structured check specifications and fix scripts. A tool may choose any convenient form to store a Value's value property, but the data type conveys how the value should be treated during benchmark compliance testing. The data type property may also be used to give additional guidance to the user or to validate the user's input. For example, if a Value object's type property was 'number', then a tool might choose to reject user tailoring input that was not composed of digits.

A Value object may extend another Value object. In such cases, the extending object receives all the properties of the extended object, and may override them where needed. A Value object with the abstract property true should never be included in any generated document, and may not be exported to any compliance checking engine.

If the interactive property is set, it is a hint to the benchmark checking tool to ask the user for a new value for the Value at the beginning of each application of the benchmark. The

checking tool is free to ignore the property if asking the user is not feasible or not supported. Similarly, the `interfaceHint` property allows the benchmark author to supply a hint to a benchmarking or tailoring tool about how the user might select or adjust the Value. The following identifiers are valid for the `interfaceHint` property: choice, textline, text, date, datetime.

When defining a Value object, the benchmark author may specify the operator to be used for checking compliance with the value. For example, one part of an OS benchmark might be checking that the configuration included a minimum password length; the Value object that holds the tailorable minimum could have type 'number' and operator 'greater than'. Exactly how Values are used in rules may depend on the capabilities of the checking system. Tailoring tools and document generation tools may ignore the operator property, therefore benchmark authors should included sufficient information in the description and question properties to make the role of the Value clear. The table below describes the operators permitted for each Value type.

Type	Available Operators	Remarks
number	equals, not equal, less than, greater than, less than or equal, greater than or equal	Default operator: equals
boolean	equals, not equal	Default operator: equals
string	equals, not equal, pattern match (pattern match means regular expression match; should comply with [7])	Default operator: equals

If a Value's `prohibitChanges` property is set to true, then it means that the Value's value may not be changed by the user. This might be used by benchmark authors in defining values that are integral to compliance, such as a timeout value, or it might be used by enterprise security officers in constraining a benchmark to more tightly reflect organizational or site security policies. (In the latter case, a security officer could use the extension facility to make an untailorable version of a Value object, without rewriting it.) A Value object can have a 'hidden' property; if the hidden property is true, then the Value should not appear in a generated document, but its value may still be used.

A Value object includes several properties that constrain or limit the values that the Value may be given: value, default, match, choices, upper-bound, and lower-bound.

Benchmark authors can use these Value properties to assist users in tailoring the Benchmark. These properties may appear more than once in a Value, and may be marked with a selector tag id. At most one of instance of each may omit its selector tag. For more information about selector tags, see the description of the Profile object below.

The default property holds a default value for the value property; tailoring tools may wish to present the default value to end users as a suggestion.

The match property provides a regular expression pattern that a tool may apply, during tailoring, to validate user input. The match property applies only when the Value type is 'string' or 'number'. For example, if the Value type was 'string', but the value was meant to be a Cisco IOS router interface name, then the Value match property 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".

The choices property holds a list of one or more particular values for the Value object; the choices property also bears a boolean flag, mustMatch, which indicates that the enumerated choices are the only legal ones (mustMatch=true) or that they are merely suggestions (mustMatch=false). The choices property 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”.

The upper-bound and lower-bound properties constrain the choices for Value items with a type property of ‘number’. For any other type, they are meaningless. The bounds they indicate are always inclusive. For example, if the lower-bound property for a Value is given as “3”, then 3 is a legal value.

The source property allows a benchmark author to supply a URI, possibly tool-specific, that indicates where a benchmarking tool may acquire values or value choices.

Profile

Property	Type	Count	Description
id	identifier	1	unique identifier for this Profile
note-tag	identifier	0-1	tag identifier to match profile-note properties in Rules, optional
title	string	1-n	title of the Item, for human readers
description	text	0-n	text that describes the Profile, optional
extends	identifier	0-1	id of a Profile on which to base this Profile, optional
abstract	boolean	0-1	if true, then this Profile exists solely to be extended by other Profiles, and may not be applied to a Benchmark directly; optional (default: false)
status	string+date	0-1	status of the Profile and date at which it attained that status, optional.
version	string+date	0-1	version of the Profile, with timestamp and update URI, optional.
prohibitChanges	boolean	0-1	whether tools should prohibit changes to this Profile (default: false)
platform	id	0-n	a target platform for this Profile; this may appear many times if the Profile applies to several platforms. The platform id is a reference to one listed in the Benchmark platform-definitions property.
reference	string+URL	0-n	a reference to a document or resource where the user can learn more about the subject of this Profile: a string and optional URL

Property	Type	Count	Description
selectors	<i>special</i>	0-n	references to Groups, Rules, and Values, see below (references may be the unique id of an Item, or a cluster id)
signature	<i>special</i>	0-1	digital signature over this Profile, optional

A Profile object is a named tailoring of a Benchmark. While a Benchmark can be tailored in place, by setting properties of various objects, only Profiles allow one Benchmark document to hold several independent tailorings. Each Profile contains a list of selectors which express a particular customization or tailoring of the Benchmark.

There are three kinds of selectors:

- select** - a Rule/Group selector
 This selector designates a Rule, Group, or cluster of Rules and Groups. It overrides the selected property on the designated items. It provides a means for including or excluding rules from the Profile.
- set-value** – a Value selector
 This selector overrides the value property of a Value object, without changing any of its other properties. It provides a means for directly specifying the value of a variable to be used in compliance checking or other benchmark processing. This selector may also be applied to the Value items in a cluster, in which case it overrides the value properties of all of them.
- refine-rule** – a Rule selector
 This selector allows the Profile author to override the scoring weight, severity, and role of a Rule.
- refine-value** – a Value selector
 This selector designates the Value constraints to be applied during tailoring, for a Value object or the Value members of a cluster. It provides a means for authors to impose different constraints on tailoring for different profiles.

A Profile can extend another Profile in the same Benchmark. The set of platform, reference, and selector properties of the extended Profile are prepended to the list of properties of the extending Profile. Inheritance of title, description, and reference properties are handled in the same way as for Item objects.

The note-tag property is a simple identifier. It specifies which profile-note properties on Rules should be associated with this Profile.

A Profile can have a status property. Benchmark authors can use the status property to record the maturity or consensus level of a Profile. If the status property is not given explicitly in a Profile definition, then the Profile is taken to have the same status as its parent Benchmark. Note that status properties are not inherited.

TestResult

Property	Type	Count	Description
id	identifier	1	identifier for this TestResults object
benchmark	URI	0-1	reference to Benchmark, mandatory if this TestResults object is in a file by itself; optional otherwise

Property	Type	Count	Description
version	string	0-1	the version number string copied from the Benchmark, optional
title	string	0-n	title of the test, for human readers
remark	string	0-n	remark the test, possibly by the person administering the test, optional
start-time	timestamp	0-1	Time when test began, optional
end-time	timestamp	1	Time when test was completed and the results recorded, mandatory
test-system	string	0-1	name of the test tool or program, optional
target	string	1	name of the system whose test results are recorded in this object, mandatory
target-address	string	0-n	network address of the target
target-facts	<i>special</i>	0-1	A sequence of named facts about the target system or platform, including a type qualifier, optional
platform	string	0-n	A sequence of platform identifiers, designating those platforms which the target system was found to meet, optional
profile	identifier	0-1	the identifier of the Benchmark profile used for the test, if any
set-value	string+id	0-n	specific settings for Value objects used during the test, one for each Value
rule-results	<i>special</i>	1-n	Outcomes of individual Rule tests, one per rule instance.
score	float+URI	1-n	an overall score for this benchmark test; at least one must appear
signature	<i>special</i>	0-1	digital signature over this TestResult object, optional

A TestResult object represents the results of a single application of the Benchmark to a single target platform. The properties of a TestResult object include test time, the identity and other facts about the system undergoing the test, and Benchmark information. If the test was conducted using a specific Profile of the Benchmark, then a reference to the Profile may be included. Also, multiple set-value properties may be included, giving the identifier and value for the Values that were used in the test.

The target-fact list is an optional part of the TestResult object. It contains a list of zero or more individual facts about the target system or platform. Each fact consists of the following: a name (URI), a type (“string”, “number”, or “boolean”), and the value of the fact itself.

The main content of a TestResult object is a collection of rule-result records, each giving the result of a single instance of a rule application against the target. The TestResult must include one rule-result record for each Rule that was selected in the resolved Benchmark; it may also include rule-result records for Rules that were unselected in the Benchmark. A rule-result record contains the properties listed below. For more information about benchmark application and scoring, see page 32.

Property	Type	Count	Description
rule-idref	identifier	1	identifier of a benchmark Rule (from the benchmark designated in the TestResult)
time	timestamp	0-1	time when application of this instance of this rule was completed, optional
version	string	0-1	the version number string copied from the version property of the Rule, optional
severity	string	0-1	the severity string code copied from the Rule; defaults to “unknown”, optional.
ident	string+URI	0-n	a globally meaningful name for the issue or vulnerability associated with this Rule, and a URI designating the system which assigned the name.
result	string	1	result of this test: one of status values listed below.
override	<i>special</i>	0-n	an XML block explaining how and why an auditor chose to override the rule’s result status, optional.
instance	strings	0-n	name of the target system component to which this result applies, for multiply instantiated rules. May also include context and hierarchy information for nested contexts. Optional.
message	string+code	0-1	a diagnostic message from the checking engine, with optional severity (this would normally appear only for results of “fail” or “error” results)
fix	string	0-1	fix script for this platform, if available (should appear only for “fail” results)
check	<i>special</i>	0-n	encapsulated or referenced results to detailed testing output from the checking engine (if any). If multiple checks were executed as part of a complex-check, then data for each may appear here.

The result of a single test may be one of the following:

- **pass** – the target system or system component satisfied all the conditions of the Rule; a pass result contributes to the weighted score and maximum possible score. [Abbreviation: P]
- **fail** – the target system or system component did not satisfy all the conditions of the Rule; a fail result contributes to the maximum possible score. [Abbreviation: F]
- **error** – the checking engine encountered a system error and could not complete the test, therefore the status of the target’s compliance with the Rule is not certain. This could happen, for example, if a benchmark testing tool were run with insufficient privileges. [Abbreviation: E]
- **unknown** – the testing tool encountered some problem and the result is unknown. (For example, a result of ‘unknown’ might be given if the benchmark testing tool were unable to interpret the output of the checking engine.) [Abbreviation: U]
- **notapplicable** – the Rule was not applicable to the target of the test. (For example, the 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.) Results with this status do not contribute to the benchmark score. [Abbreviation: N]
- **notchecked** – the Rule was not evaluated by the checking engine. This status is designed for Rules with a role of “unchecked”, and for Rules that have no check properties. It may also correspond to a status returned by a checking engine. Results with this status do not contribute to the benchmark score. [Abbreviation: K]
- **notselected** – the Rule was not selected in the Benchmark. Results with this status do not contribute to the benchmark score. [Abbreviation: S]
- **informational** – the Rule was checked, but the output from the checking engine is simply information for auditor or administrator, it is not a compliance category. This status is the default for Rules with a role of “unscored”. This status value is designed for Rules whose main purpose is to extract information from the target rather than test compliance. Results with this status do not contribute to the benchmark score. [Abbreviation: I]
- **fixed** – the Rule had failed, but was then fixed (possibly by a tool that can automatically apply remediation, or possibly by the human auditor). Results with this status should be scored the same as **pass**. [Abbreviation: X]

The instance property specifies the name of a target subsystem or component that passed or failed a Rule. This is important for Rules that apply to components of which a target might have several. For example, a Rule might specify a particular setting that needs to be applied on every interface of a firewall; for benchmark compliance results, a firewall target with three interfaces would have three rule-result elements with the same rule id, each with an independent value for the result property. For more discussion of multiply instantiated rules, see page 35.

The check property consists of the URI that designates the checking system, and detailed output data from the checking engine. The detailed output data can take the form of encapsulated XML or text data, or it can be a reference to an external URI. (Note: this is analogous to the form for referring to checking engine input defined for Rule objects.)

The override property provides a mechanism for an auditor to change the Rule result assigned by the benchmark checking tool. This is necessary when (a) checking a rule requires reviewing manual procedures or other non-IT conditions, and (b) when a benchmark check just gets the wrong answer. The override element contains the following properties.

Property	Type	Count	Description
time	timestamp	1	when the override was applied
authority	string	1	name or other identification for the human principal authorizing the override
old-result	string	1	the rule result status before this override.
new-result	string	1	the new, override rule result status.
remark	string	1	rationale or explanation text for why or how the override was applied.

XCCDF is not intended to be a database format for detailed results; the TestResult object offers a way to store the results of individual tests in modest detail, with the ability to reference lower level testing data.

3.3. Processing Models

The XCCDF specification is design to support automated benchmark processing by a variety of tools. There are four basic kinds of processing that a tool might apply to a XCCDF document:

1. Tailoring –
This kind of processing involves loading an XCCDF documents, allowing a user to set the value property of Value items and the selected property of all Items, and then generation of a tailored XCCDF output document.
2. Document Generation –
This kind of processing involves loading an XCCDF document and generating non-XCCDF output, usually in a form suitable for printing or human perusal.
3. Transformation –
This is the most open-ended of the processing types: it involves transforming an XCCDF document into a document in some other representation. Typically, a transformation process will involve some kind of stylesheet or specification that directs the transformation (e.g. an XSLT stylesheet). This kind of processing can be used in a variety of contexts, including document generation.
4. Compliance Checking –
This is the primary form of processing for XCCDF documents. It involves loading an XCCDF document, checking target systems or data representing them, and generating one or more compliance reports in non-XCCDF format. One of

the reports would usually include a compliance score, and another might be a fix script. Some tools might also generate other outputs or store compliance information in some kind of database.

5. Test Report Generation –

This form of processing can be done only to an XCCDF document that includes one or more TestResult objects. It involves loading the document, traversing the list of TestResult objects, and generating non-XCCDF output about selected ones.

Tailoring, document generation, and compliance checking all share a similar processing model consisting of two steps: loading and traversal. The processing sequence required for loading are described in the subsection below. Note that loading must be complete before traversal begins. When loading is complete, a Benchmark is said to be *resolved*.

Loading Processing Sequence

Before any loading begins, a tool should initialize an empty set of legal notices and an empty dictionary of object ids.

Sub-Step	Description
Loading.Import	Import the XCCDF document into the program and build an initial internal representation of the Benchmark object, Groups, Rules, and other objects. If the file cannot be read or parsed, then Loading fails. (At this time, any inclusion processing specified with XInclude elements should be performed. The resulting XML information set, after inclusion, should be validated against the XCCDF schema given in Appendix A.) Go to the next step: Loading.Noticing.
Loading.Noticing	For each notice property of the Benchmark object, add the notice to the tool's set of legal notices. If a notice with an identical id value is already a member of the set, then replace it. If the benchmark's resolved property is set, then Loading succeeds, otherwise go to the next step: Loading.Resolve.Items.
Loading.Resolve.Items	For each Item in the Benchmark that has an extends property, resolve it by using the following steps: (1) if the Item is Group, resolve all the enclosed Items, (2) resolve the extended Item, (3) prepend the property sequence from the extended Item to the extending Item, (4) if the Item is a Group, assign values for the id properties of Items copied from the extended Group, (5) remove duplicate properties and apply property overrides, and (6) remove the extends property. If any Item's extends property 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 properties includes a loop, then Loading fails. Otherwise, go to Loading.Resolve.Abstract.
Loading.Resolve.Profiles	For each Profile in the Benchmark that has an extends property, resolve the set of properties in the extending Profile by applying the following steps: (1) resolve the extended Profile, (2) prepend the property sequence from the extended Profile to that of the extending Profile, (3) remove duplicate properties. If any Profile's extends property identifier does not match the identifier of another Profile in the Benchmark, then Loading fails. If the directed graph formed by

Sub-Step	Description
	the extends properties of Profiles includes a loop, then Loading fails. Otherwise, go to Loading.Resolve.Abstract.
Loading.Resolve.Abstract	For each Item in the Benchmark for which the abstract property is true, remove the Item. For each Profile in the Benchmark for which the abstract property is true, remove the Profile.
Loading.Resolve.Finalize	Set the benchmark resolved property to true; Loading succeeds.

If the Loading step succeeds for a XCCDF document, then the internal data model should be complete, and every Item should contain all of its own content. A XCCDF file for that has no extends properties is called a resolved document. Only resolved XCCDF documents should be subjected to Transformation processing.

XML Inclusion processing must happen before any validation or processing. Typically, it will be performed by the XML parser.

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 appended to the list inherited from the extending object.

There are four different inheritance processing models for Item and Profile properties.

- None – the property value or values are not inherited.
- Replacement – the property value is inherited, a property value explicitly defined on the extending object replaces an inherited value.
- Append – the property values are inherited from the extended object, additional values may be defined on the extending object.
- Override – the property values are inherited from the extended object, additional values may be defined on the extending object. An additional value can override (replace) an inherited value, if explicitly tagged as ‘override’.

The table below shows the inheritance processing model for each of the properties supported on Group, Rule, Value, and Profile objects.

Processing Model	Properties	Remarks
None	abstract, cluster-id, extends, id, signature, status	The extends property cannot be inherited.
Replacement	hidden, prohibitChanges, selected, version, weight, check, complex-check, role, severity, type, interactive	For the check property, checks from different systems are considered different properties.
Append	requires, conflicts, ident, fix, value, default, choices, operator, lower-bound, upper-bound, match	Additional rules may apply during Benchmark processing, tailoring, or report generation.

Processing Model	Properties	Remarks
Override	title, description, platform, question, rationale, warning, reference, fixtext	For properties that have a locale (xml:lang specified), values with different locales are considered to be different properties.

Every resolved document must satisfy the condition that every id attribute is unique. Therefore, it is very important that the Loading.Resolution step generate a fresh unique id for any Group, Rule, or Value object that gets created through extension of its enclosing Group. One way to do this would be to generate and assign a random unique id during sub-step (4) of Loading.Resolve.Items. Also note that it is necessary to assign an extends property to the newly created Items, based on the id or extends property of the Item that was copied (if the Item being copied has an extends property, then the new Item gets the same value for the extends property, otherwise, the new Item gets the id value of the Item being copied as its extends property).

The second step of processing is Traversal. The concept behind Traversal is basically a pre-order, depth-first walk through all the Items that make up a Benchmark. However, Traversal works slightly differently for each of the three kinds of processing, as described further below.

Benchmark Processing Algorithm

The id of a Profile may be specified as input for Benchmark processing.

Sub-Step	Description
Benchmark.Front	Process the properties of the Benchmark object.
Benchmark.Profile	If a Profile id was specified, then apply the settings in the Profile to the Items of the Benchmark.
Benchmark.Content	For each Item in the Benchmark object's items property, initiate Item.Process.
Benchmark.Back	Perform any additional processing of the Benchmark object properties.

The sub-steps Front and Back will be different for each kind of processing, and each tool may perform specialized handling of Benchmark properties. For document generation, Profiles may be processed separately as part of Benchmark.Back, to generate part of the output document.

Item Processing Algorithm

Sub-Step	Description
Item.Process	Check the contents of the requires and conflicts properties, and if any required Items are unselected or any conflicting Items are selected, then set the selected and allowChanges properties to false.
Item.Select	If any of the following conditions holds, cease processing of this Item. 1. If the processing type is Tailoring, and the optional property and selected property are both false.

Sub-Step	Description
	2. If the processing type is Document Generation, and the hidden property is true. 3. If the processing type is Compliance Checking, and the selected property is false. 4. If the processing type is Compliance Checking, and the current platform (if known by the tool) is not a member of the set of platforms for this Item.
Group.Front	If the Item is a Group, then process the properties of the Group.
Group.Content	If the Item is a Group, then for each Item in the Group's items property, initiate Item.Process.
Rule.Content	If the Item is a Rule, then process the properties of the Rule.
Value.Content	If the Item is a Value, then process the properties of the Value.

Processing the properties of an Item is the core of Benchmark processing. The list below describes some of the processing in more detail.

- For Tailoring, the key to processing is to query the user and incorporate their response into the data. For a Group or Rule, the user should be given a yes/no choice if the optional property is true. For a Value item, the user should be given a chance to supply a string value, possibly validated using the type property. The output of a tailoring tool will usually be another XCCDF file.
- 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 will depend on the tool and the target output format. In general, the selected and optional properties are not germane to Document Generation. The platform properties may be used during Document Generation for generation of platform-specific versions of a document.
- For Compliance Checking, the key to processing is applying the Rule checks to the target system or to collected data about the target system. Tools will vary in how they do this, and in how they generate output reports. It is also possible that some Rule checks will need to be applied to multiple contexts or features of the target system, generating multiple pass or fail results for a single Rule object.

Note that it is possible (but inadvisable) for a benchmark author to set up circular dependencies or conflicts using the requires and conflicts properties. To prevent ambiguity, tools must process the items of the benchmark in order, and must not change the selected property of any Rule or Group more than once during a processing session.

Substitution Processing

XCCDF supports the notion of named parameters, Value objects, which can be set by a user during the tailoring process, and then substituted into content specified elsewhere in the benchmark. XCCDF 1.1 also supports the notion of plain text definitions on a Benchmark, these are re-usable chunks of text that may be substituted into other texts using the substitutions facilities described here.

As described in the next section, a substitution is always indicated by a reference to the id of a particular Value object, plain-text definition, or other Item in the Benchmark.

During Tailoring and Document Generation, a tool should substitute the title property of the Value object for the reference in any text shown to the user or included in the document. At the tool author's discretion, the title may be followed by the Value object's value property, in parentheses, if the value is not empty. For plain-text definitions, any reference to the definition should be replaced by the string content of the definition.

Any appearance of the instance element in the content of a fix element should be replaced by a locale-appropriate string to represent a target system instance name.

During Compliance Checking, Value objects designated for export to the checking system are passed to it. In general, the interface between the XCCDF checking tool and the underlying checking system or engine must support passing the following properties of the Value: value, type, and operator.

During creation of TestResult objects on conclusion of Compliance Checking, any fix elements present in applied Rules, and matching the platform to which the compliance test was applied, should be subjected to substitution and the resulting string used as the value of the fix element for the rule-result element. Each sub element should be replaced by the value of the referenced Value object or plain-text definition actually used during the test. Each instance element should be replaced by the value of the rule-result instance element.

Rule Application and Compliance Scoring

When a benchmark compliance checking tool performs a compliance run against a system, it accepts as inputs the state of the system and a Benchmark, and produces some outputs, as shown below.

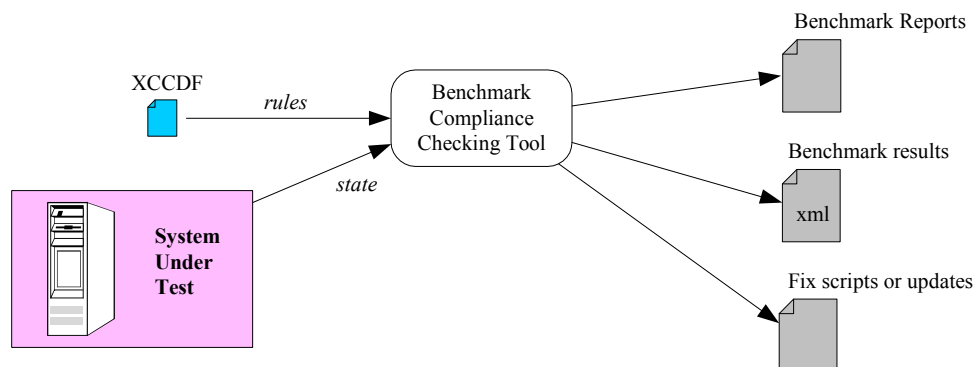


Figure 4 – Workflow for Checking Benchmark Compliance

- **Benchmark Report** – a human-readable report about compliance, including the compliance score, and a listing of which rules which passed and failed in 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; multiply-instantiated rules are discussed in more detail below. The report may also include recommended steps for improving compliance. The format of the benchmark report is not specified here; but might be some form of formatted or rich text (e.g. HTML).

- Benchmark results – a machine-readable file about compliance, meant for storage, long-term tracking, or incorporation into other reports (e.g. a site-wide compliance report). This file may be in XCCDF, using the TestResult object, or it may be in some tool-specific XML format.
- Fix scripts – machine-readable files, usually text, the application of which will remediate some or all of the non-compliance issues found by the tool. These scripts may be included in XCCDF TestResult objects.

Scoring and Results Model

The output or *result* of a single benchmark compliance test consists of four parts:

1. Rule result list – a vector V of result elements e , each element is a 6-tuple $e = \{r, p, I, t, F, O\}$, where:
 - r is the Rule id
 - p is the test result, one of {pass, fail, error, unknown, notapplicable, notchecked, notselected, informational, fixed}. A test whose result p is ‘error’ or ‘unknown’ is treated as ‘fail’ for the purposes of scoring; tool developers may wish to alert the user to erroneous and unknown test results. A test whose result p is one of {notapplicable, notchecked, informational, notselected} does not contribute to scoring in any way. A test whose result p is ‘fixed’ is treated as a pass for score computation.
 - I is the instance set, identifying the system components, files, interfaces, or subsystems to which the Rule was applied. Each element of I is a triple $\{n, c, p\}$, where n is the instance name, c is the optional instance context, and p is the optional parent context. The context c , when present, describes the scope or significance of the name n . The parent context p allows the members of I to express nested structure. I must be an empty set for tests that are not the result of multiply instantiated rules (see below).
 - t is the instant of time at which the result of the Rule application was decided.
 - F is the set of fixes, from the Rule’s fix properties, that should bring the target system into compliance (or at least closer to compliance) with the rule. F may be null if the Rule did not possess any applicable fix properties, and must be null when p is equal to pass. Each fix f in F consists of all the properties defined in the description of the Rule fix property: content, strategy, disruption, reboot, system, id, and platform.
 - O is the set of overrides, each o in O consisting of the five properties listed for the rule-result override property: time, authority, old-result, new-result, and remark. Overrides do not affect score computation.
2. Scores – a vector S , consisting of one or more score values s , each s a pair consisting of a real number and a scoring model identifier.
3. Identification – a vector of strings identify the Benchmark, Profile (if any), and target system to which the benchmark was applied.
4. Timestamps – two timestamps recording the beginning and the end of the interval when the benchmark was applied and the results compiled.

Each element of the pass/fail list V conveys the compliance of the system under test, or one component of it, with one rule of benchmark. Each rule has a weight, title, and other attributes as described above. Each element of V may include an instance name, which gives the name of a system component to which the pass or fail designation applies.

XCCDF 1.1 defines a default scoring model, two optional scoring models, and permits benchmark checking tools to support additional proprietary or consensus models. The default model computes a score is based on relative weights of sibling rules, as described in the next sub-section. A Benchmark may specify the scoring model to be used. In the absence of an explicit scoring model specified in the Benchmark, compliance checking tools must compute a score based on the default XCCDF model, and may compute additional scoring values based on other models.

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 Benchmark. This relative-to-siblings weighted scoring model is designed for flexibility and to foster independent authorship of collections of rules. Benchmark authors must keep the model in mind when assigning weights to Groups and Rules. For a very simple Benchmark consisting only of Rules and no Groups, weights may be omitted.

The fix scripts are collected from the fix properties of the rules in elements of V where p is False. A compliance checking or remediation tool may choose to concatenate, consolidate, and/or deconflict the fix scripts, mechanisms for doing so are outside the scope of this specification. In the simplest cases, tools must perform Value substitution on each rule's fix property before making it part of the output results.

Score Computation Algorithms

This sub-section describes the XCCDF default scoring model, which compliance checking tools must support, and two additional models that tools may support. Each scoring model is identified by a URI.

The Default Model

This model is identified by the URI “urn:xccdf:scoring:default”. It was the only model supported in XCCDF 1.0, and remains the default for compatibility.

The objects of an XCCDF benchmark form the nodes of a tree. The score computation algorithm simply computes a normalized weighted sum at each node, omitting Rules and Groups that are not selected, and Groups that have no selected Rules under them. The algorithm at each selected node is:

Sub-Step	Description
Score.Rule	If the node is a Rule, then assign a count of 1, and if the test result is ‘pass’ assign the node a score of 100, otherwise assign a score of 0.
Score.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.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

Sub-Step	Description
	add the child's weight value to the accumulator a .
Score.Group.Normalize	Normalized this node's score: compute $s = s / a$
Score.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, which is the Benchmark object.

The Flat Model

This model is identified by the URI “urn:xccdf:scoring:flat”.

Under this model, the set of rule results is treated as a vector V , as described above. The following algorithm is used to compute the score.

Sub-Step	Description
Score.Init	Initialize the score s to 0.0. Initialize the maximum score m to 0.0.
Score.Rules	For each element e in V where $e.p$ is not equal to ‘notapplicable’: - add the weight of rule $e.r$ to m . - if the value $e.p$ equals ‘pass’ or ‘fixed’, add the weight of the rule $e.r$ 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.

The Flat Unweighted Model

This model is identified by the URI “urn:xccdf:scoring:flat-unweighted”. It is computed in exactly the same way as the flat model, above, except that all weights are taken to be 1.0.

Multiply-Instantiated Rules

A security auditor applying a security benchmark to a system typically wants to know two things: how well does the system comply, and how can non-compliant items be reconciled (either fixed or determined not to be salient)?

Many benchmarks include rules that apply to system components. For example, a host OS benchmark would probably contain rules that apply to all users, and a router benchmark will contain rules that apply to all network interfaces. When the system holds many of such components, it is not adequate for a tool to inform the administrator or auditor that the rule failed, it should report exactly which components failed the rule.

A processing engine that performs a benchmark compliance test may deliver zero or more pass/fail triples, as described above. In the most common case, each compliance test rule will yield one result element. In a case where a rule was applied multiple times to multiple components of the system under test, a single rule could yield multiple result

elements. If each of multiple relevant components passes the rule, the processing engine may deliver a single result element with an instance set $I=\text{null}$. For the purposes of scoring, a rule contributes to the positive score only if all instances of that rule have a test result of 'pass'. (This is sometimes called "strict scoring". If any component of the target system fails a rule, then the entire Rule is considered to have failed.)

4. XML Representation

This section defines a concrete representation of the XCCDF data model in XML, using both core XML syntax and XML Namespaces.

4.1. XML Document General Considerations

The basic document format consists of a root “Benchmark” element, representing a Benchmark object. Its child elements are the contents of the benchmark object, as described in Section 3.2.

All the XCCDF elements in the document will belong to the XCCDF namespace, including the root element. The namespace URI corresponding to this version of the specification is “<http://checklists.nist.gov/xccdf/1.1>”. The namespace of the root Benchmark element serves to identify the XCCDF version for a document. Applications that process XCCDF can use the namespace URI to decide whether they can process a given document. If a namespace prefix is used, the suggested prefix string is “cdf”.

XCCDF attributes are not namespace qualified. All attributes begin with a lowercase letter, except the “Id” attribute (for compatibility with XML Digital Signatures[8]).

The example below illustrates the outermost structure of an XCCDF XML document.

Example 1 – Top-Level XCCDF XML

```
<?xml version="1.0" ?>
<cdf:Benchmark id="example1" xml:lang="en" Id="toSign"
  xmlns:htm="http://www.w3.org/1999/xhtml"
  xmlns:cdf="http://checklists.nist.gov/xccdf/1.1"
  xmlns:cdfp="http://www.cisecurity.org/xccdf/platform/0.2.3"/>
  <cdf:status date="2004-10-12">draft</cdf:status>
  <cdf:title>Example Benchmark File</cdf:title>
  <cdf:description>
    A <htm:b>Small</htm:b> Example
  </cdf:description>
  <cdf:version>0.2</cdf:version>
  <cdf:reference href="http://www.ietf.org/rfc/">
    Internet RFCs
  </cdf:reference>
</cdf:Benchmark>
```

Validation is strongly suggested but not required for tools that process XCCDF documents. The XML Schema attribute “schemaLocation” may be used to refer to the XCCDF Schema (see Appendix A).

Properties of XCCDF objects marked as type ‘text’ in Section 3.2 may contain embedded formatting, presentation, and hyperlink structure. XHTML Basic tags must be used to express formatting, presentation, and hyperlink structure for XCCDF XML document. In particular, the core modules noted in the XHTML Basic Recommendation are permitted in XCCDF documents, plus the Image module and the Presentation module. How an XCCDF processing tool handles embedded XHTML content in XCCDF text properties is

implementation-dependent, but at the least every tool must be able to process XCCDF files even when embedded XHTML elements are present. Tools 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.

4.2. XML Element Dictionary

This subsection describes each of the elements and attributes of XCCDF XML. Each description includes the parent elements feasible for that element, as well as the child elements it might normally contain. Most elements are in the XCCDF namespace, which for version 1.1 is “<http://checklists.nist.gov/xccdf/1.1>”. For a more precise definition, consult Appendix A.

Many of the elements listed below are described as containing formatted text (type ‘text’ in Section 3.2). These elements may contain Value substitutions, and formatting expressed as described in Section 4.3.

XML is case-sensitive. The XML syntax for XCCDF follows a common convention for representing object-oriented data models in XML: elements that correspond directly to object classes in the data model have names with initial caps. Mandatory attributes and elements are shown in bold. Child elements are listed in the order in which they must appear. Elements which are not part of the XCCDF namespace are shown in italics.

<Benchmark>

This is the root element of the XCCDF document, it must appear exactly once. It encloses the entire benchmark, and contains both descriptive information and benchmark structural information. The id attribute must be a unique identifier.

Content:	elements
Cardinality:	1
Parent Element:	<i>none</i>
Attributes:	id , resolved, Id (note: “Id” is needed only for digital signature security)
Child Elements:	status, title , description, notice, front-matter, rear-matter, reference, <i>platform-definitions</i> , platform, version, metadata, Profile, Value, Group, Rule, signature

Note that the order of Group and Rule child elements may matter for the appearance of a generated document. Therefore, Group and Rule children may be freely intermingled. All the other children must appear in the order shown, and multiple instances of a child element must be adjacent.

<Group>

A Group element contains descriptive information about a portion of a benchmark, as well as Rules, Values, and other Groups. A Group must have a unique id attribute in order to be referenced from other XCCDF documents or extended by other Groups. The id attribute must be a unique identifier. The extends attribute, if present, must have a

value equal to the id attribute of another Group. The cluster-id attribute is an id, it designates membership in a ‘cluster’ of Items, which are used for controlling Items via Profiles. The hidden and allowChanges attributes are of boolean type and default to false. The weight attribute is a positive real number.

Content:	elements
Cardinality:	0-n
Parent Elements:	Benchmark, Group
Attributes:	id , cluster-id, extends, hidden, prohibitChanges, selected , weight, Id
Child Elements:	status, version, title, description, warning, question, reference, rationale, platform, requires, conflicts, Value, Group, Rule

A Group must have an id attribute. All child elements are optional, but every group should have a title. Group and Rule children may be freely intermingled. All the other children must appear in the order shown, and multiple instances of a child element must be adjacent.

The extends attribute allows a benchmark author to define a group as an extension of another group. The example XML fragment below shows an example of an extended and extending Group.

Example 2 – A Simple XCCDF Group

```
<cdf:Group id="basegrp" selected="0" hidden="1">
  <cdf:title>Example Base Group</cdf:title>
  <cdf:reference>Consult the vendor documentation.</cdf:reference>
</cdf:Group>
<cdf:Group extends="basegrp" id="fileperm" selected="1" hidden="0">
  <cdf:title>File Permissions</cdf:title>
  <cdf:description>
    Rules related to file access control and
    user permissions.
  </cdf:description>
  <cdf:question>
    Include checks for file access controls?
  </cdf:question>
  <cdf:reference href="http://www.vendor.com/docs/perms.html">
    Administration manual, permissions settings reference
  </cdf:reference>
  . . .
</cdf:Group>
```

An XCCDF Group may only extend a Group that is within its visible scope. The visible scope includes sibling elements, siblings of ancestor elements, and the visible scope of any Group that an ancestor Group extended.

Note that circular dependencies of extension are not permitted.

<Rule>

A Rule element defines a single item to be checked as part of benchmark, or an extendable base definition for such items. A Rule must have a unique id attribute in order to be referenced from other XCCDF documents or extended by other Rules.

The id attribute must be a unique identifier. The extends attribute, if present, must have a value equal to the id attribute of another Rule. The weight attribute must be a positive real number. The other attributes are all boolean. Rules may not be nested.

Content:	elements
Cardinality:	0-n
Parent Elements:	Benchmark, Group
Attributes:	id, cluster-id, extends, hidden, prohibitChanges, role, selected, severity, weight, Id
Child Elements:	status, version, title, description, warning, question, reference, rationale, platform, requires, conflicts, ident, profile-note, fixtext, fix, complex-check, check

The check child of a Rule is the vital piece that specifies how to check compliance with a security practice or guideline, see the description of the check element below for more information. Example 3 shows a very simple Rule element.

Example 3 – A Simple XCCDF Rule

```
<cdf:Rule id="pwd-perm" selected="1" weight="6.5" severity="high">
  <cdf:title>Password File Permission</cdf:title>
  <cdf:description>Check the access control on the password
    file. Normal users should not be able to write to it.
  </cdf:description>
  <cdf:requires idref="passwd-exists"/>
  <cdf:fixtext>
    Set permissions on the passwd file to owner-write, world-read
  </cdf:fixtext>
  <cdf:fix strategy="restrict" reboot="0" disruption="low">
    chmod 644 /etc/passwd
  </cdf:fix>
  <cdf:check system="http://www.mitre.org/XMLSchema/oval">
    <cdf:check-content-ref href="ovaldefs.xml" name="OVAL123"/>
  </cdf:check>
</cdf:Rule>
```

An XCCDF Rule may only extend a Rule that is within its visible scope. The visible scope includes sibling Rules, Rules that are siblings of ancestor Groups, and the visible scope of any Group that an ancestor Group extended.

Circular dependencies of extension may not be defined.

<Value>

A Value element represents a named parameter whose title or value may be substituted into other strings in the benchmark (depending on the form of processing to which the

benchmark is being subjected), or it may represent a basis for the definition of such parameters via extension. A Value object must have a unique id attribute in order to be referenced for substitution or extension or for inclusion into another benchmark.

A Value object may appear as a child of the Benchmark, or as a child of a Group. Value objects may not be nested. The value and default child elements must appear first.

Content:	elements
Cardinality:	0-n
Parent Elements:	Benchmark, Group
Attributes:	id , cluster-id, extends, hidden, prohibitChanges, operator, type, interactive, interfaceHint, Id
Child Elements:	status, version, title, description, warning, question, reference, value , default, match, lower-bound, upper-bound, choices, source

The type attribute is optional, but if it appears it must be one of ‘number’, ‘string’, or ‘boolean’. A tool performing tailoring processing may use this type name to perform user input validation. Example 4, below, shows a very simple Value object.

Example 4 – Example of a Simple XCCDF Value

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

(Note that the match element applies only for validation during XCCDF tailoring, while the operator attribute applies only for rule checking. Do not get them confused.)

<Profile>

A Profile element encapsulates a tailoring of the Benchmark. It consists of an id, descriptive text properties, and zero or more selectors that refer to Group, Rule, and Value objects in the Benchmark. There are three selector elements: select, set-value, and refine-value.

Profile elements may only appear as direct children of the Benchmark element. A Profile may be defined as extending another Profile, using the extends attribute.

Content:	elements
Cardinality:	0-n
Parent Elements:	Benchmark
Attributes:	abstract, id , extends, prohibitChanges, Id, note-tag
Child Elements:	status, version, title , description, reference, platform, select, set-value, refine-value, refine-rule

Profiles are designed to support encapsulation of a set of tailorings. A Profile implicitly includes all the Groups and Rules in the Benchmark, and the select element children of the Profile affect which Groups and Rules are selected for processing when the Profile is in effect. The example below shows a very simple Profile.

Example 5 – Example of a Simple XCCDF Profile

```
<cdf:Profile id="strict" prohibitChanges="1"
            extends="lenient" note-tag="strict-tag">
  <cdf:title>Strict Security Settings</cdf:title>
  <cdf:description>
    Strict lockdown rules and values, for hosts deployed to
    high-risk environments.
  </cdf:description>
  <cdf:set-value idref="password-len">10</cdf:set-value>
  <cdf:refine-value idref="session-timeout" selector="quick"/>
  <cdf:refine-rule idref="session-auth-rule" selector="harsh"/>
  <cdf:select idref="password-len-rule" selected="1"/>
  <cdf:select idref="audit-cluster" selected="1"/>
  <cdf:select idref="telnet-disabled-rule" selected="1"/>
  <cdf:select idref="telnet-settings-cluster" selected="0"/>
</cdf:Value>
```

<TestResult>

The TestResult object encapsulates the result of applying a Benchmark to one target system. The TestResult element normally appears as the child of the Benchmark element, although it may also as the top-level element of a file.

Content:	elements
Cardinality:	0-n
Parent Elements:	Benchmark
Attributes:	id , start-time, end-time , Id
Child Elements:	title, remark, profile, target, target-address, target-facts, value, rule-result, score

The id attribute is a mandatory unique-id for a test result. The start-time and end-time attributes must have the format of a timestamp; the end-time attribute is mandatory, and gives the time that the application of the benchmark completed.

The example below shows a TestResult object with a couple of rule-result children.

Example 6 – Example of XCCDF Benchmark Test Results

```
<cdf:TestResult id="ios-test5" end-time="2004-09-25T7:45:02-04:00"
  xmlns:cdf="http://checklists.nist.gov/xccdf/1.1">
  <cdf:benchmark href="ios-sample-12.4.xccdf.xml"/>
  <cdf:title>Sample Results Block</cdf:title>
  <cdf:remark>Test run by Bob on Sept 25</cdf:remark>
  <cdf:target>lower.test.net</cdf:target>
```

```

<cdf:target-address>192.168.248.1</cdf:target-address>
<cdf:target-address>2001:8::1</cdf:target-address>
<cdf:target-facts>
  <cdf:fact type="string" name="urn:xccdf:fact:ethernet:MAC">
    02:50:e6:c0:14:39
  </cdf:fact>
  <cdf:fact name="urn:xccdf:fact:OS:IOS">1</cdf:fact>
</cdf:target-facts>
<cdf:set-value idref="exec-timeout-time">10</cdf:set-value>
<cdf:rule-result idref="ios12-no-finger-service"
  time="2004-09-25T13:45:00-04:00">
  <cdf:result>pass</cdf:result>
</cdf:rule-result>
<cdf:rule-result idref="req-exec-timeout"
  time="2004-09-25T13:45:06-04:00">
  <cdf:result>fail</cdf:result>
  <cdf:instance>console</cdf:instance>
  <cdf:fix>
    line console
    exec-timeout 10 0
  </cdf:fix>
</cdf:rule-result>
<cdf:score>67.5</cdf:score>
</cdf:TestResult>

```

<benchmark>

This simple element may only appear as the child of a TestResult. It indicates the Benchmark for which the TestResult records results. It has one attribute, which gives the URI of the benchmark XCCDF document. It must be an empty element.

Content	<i>none</i>
Cardinality:	0-1
Parent Elements:	TestResult
Attributes:	href
Child Elements:	<i>none</i>

The benchmark element should be used only in a standalone TestResult file (an XCCDF document file whose root element is TestResult).

<check>

This element holds a specification for how to check compliance with a Rule. It may only appear as a child of a Rule element. The child elements of this element specify the Values to pass to a checking engine, and the logic for the checking engine to apply. The logic may be embedded directly as inline text or XML data, or may be a reference to an element of an external file indicated by a URI. If the compliance checking system uses XML namespaces, then the system attribute for the system should be its namespace. The default or nominal content for a check element is a compliance test expressed as an OVAL definition or a reference to an OVAL definition, with the system attribute set to

the OVAL namespace. The check element is also used as part of a TestResult rule-result element, in that case it holds or refers to detailed output from the checking engine.

Content:	mixed
Cardinality:	0-n
Parent Elements:	Rule, rule-result
Attributes:	selector, system
Child Elements:	check-export, check-content, check-content-ref

A check element may have a selector attribute, which may be referenced from a benchmark Profile as a means of refining the application of the Rule. When Profiles are not used, then all check elements with non-empty selectors are ignored.

Several check elements may appear as children of the same Rule element. Sibling check elements must have different values for the combination of their selector and system attributes. A tool processing the benchmark for compliance checking must pick at most one check element to process for each Rule. The check element may contain zero or more check-export elements, and must contain either one check-content or one check-content-ref element.

When a check element is a child of a TestResult object, two special conditions apply: (1) check-export elements should not appear, and any present must be ignored, and (2) the selector attribute should not appear, and any selector attributes present must be ignored.

<check-export>

This specifies a mapping from an XCCDF Value object to a checking system variable. The value-id attribute must match the id attribute of a Value object in the benchmark.

Content:	none
Cardinality:	0-n
Parent Elements:	check
Attributes:	value-id, export-name
Child Elements:	<i>none</i>

<check-content>

This element holds the actual code of a benchmark compliance check, in the language or system specified by the check element's system attribute. Exactly one of check-content or check-content-ref must appear in each check element. The body of this element can be any XML, but cannot contain any XCCDF elements. XCCDF tools are not required to process this element; typically it will be passed to a checking system or engine.

Content:	<i>any non-XCCDF</i>
Cardinality:	0-1
Parent Elements:	check
Attributes:	<i>none</i>
Child Elements:	<i>special</i>

<check-content-ref>

This element points to a benchmark compliance check, in the language or system specified by the check element's system attribute. Exactly one of check-content or check-content-ref must appear in each check element. The href attribute identifies the document, and the optional name attribute may be used to refer to a particular part, element, or component of the document.

Content:	<i>none</i>
Cardinality:	0-1
Parent Elements:	check
Attributes:	href, name
Child Elements:	<i>none</i>

<choices>

The choices element may be a child of a Value, and it enumerates one or more legal values for the Value. If the boolean mustMatch attribute is true, then the list represents all the legal values; if mustMatch is absent or false, then the list represents suggested values but other values might also be legal (subject to the parent Value's upper-bound, lower-bound, or match attributes). The choices element may have a selector attribute that is used for tailoring via a Profile. The list given by this element is intended for use during tailoring and document generation, it has no role in benchmark compliance checking.

Content:	elements
Cardinality:	0-n
Parent Elements:	Value
Attributes:	mustMatch, selector
Child Elements:	choice

<choice>

This string element is used to hold a possible legal value for a Value object. It must appear as the child of a choices element, and has no attributes or child elements.

Content:	string
Cardinality:	1-n
Parent Elements:	choices
Attributes:	<i>none</i>
Child Elements:	<i>none</i>

<complex-check>

This element may only appear as a child of a Rule. It contains a boolean expression composed of operators (and, or, not) and individual checks.

Content:	elements
Cardinality:	0-1

Parent Elements: Rule
 Attributes: **operator**, negate
 Child Elements: complex-check, check

With an “AND” operator, the complex-check evaluates to Pass only if all of its enclosed terms (checks and complex-checks) evaluate to Pass. For purposes of evaluation, Pass (P) and Fixed (X) are considered equivalent. The truth table for “AND” is given below.

AND	P	F	U	E	N
P	P	F	U	E	P
F	F	F	F	F	F
U	U	F	U	U	U
E	E	F	U	E	E
N	P	F	U	E	N

The ‘OR’ operator evaluates to Pass if any of its enclosed terms evaluate to Pass. The truth table for “OR” is given below.

OR	P	F	U	E	N
P	P	P	P	P	P
F	P	F	U	E	F
U	P	U	U	U	U
E	P	E	U	E	E
N	P	F	U	E	N

If the negate attribute is set to true, then the result of the complex-check must be complemented (inverted). The full truth table for negation is given below.

	P	F	U	E	N
not	F	P	U	E	N

The example below show a complex-check with several components.

Example 7 – Example of XCCDF Complex Check

```
<cdf:complex-check operator="OR">
  <cdf:check system="http://oval.mitre.org/XMLSchema/oval">
    <cdf:check-content-ref href="xpDefs.xml" name="XP-P1"/>
  </cdf:check>
  <cdf:complex-check operator="AND" negate="1">
    <cdf:check system="http://oval.mitre.org/XMLSchema/oval">
      <cdf:check-content-ref href="xpDefs.xml" name="XP-CX"/>
    </cdf:check>
    <cdf:check system="http://www.cisecurity.org/xccdf/interactive/1.0">
      <cdf:check-content-ref href="xpInter.xml" name="XP-6"/>
    </cdf:check>
  </cdf:complex-check>
</cdf:complex-check>
```

<conflicts>

The conflicts element may be a child of any Group or Rule, and it specifies the id property of another Group, Rule, or Value whose selection conflicts with this one. Each conflicts element specifies a single conflicting Item using its idref attribute; if the semantics of the benchmark need multiple conflicts, then multiple conflicts elements may appear. A conflicts element must be empty.

Content:	<i>none</i>
Cardinality:	0-n
Parent Elements:	Group, Rule
Attributes:	idref
Child Elements:	<i>none</i>

<default>

This string element is used to hold the default or reset value of a Value object. It must appear as the child of a Value element, and has no child elements. This element may have a selector attribute, which may be used to designate different defaults for different benchmark Profiles.

Content:	string
Cardinality:	0-n
Parent Elements:	Value
Attributes:	selector
Child Elements:	<i>none</i>

<description>

This element provides the descriptive text for a Benchmark, Rule, Group, or Value. It has no attributes. Multiple description elements may appear with different values for their xml:lang attribute (see also next section).

Content:	mixed
Cardinality:	0-n
Parent Elements:	Benchmark, Group, Rule, Value, Profile
Attributes:	xml:lang, override
Child Elements:	sub, <i>xhtml elements</i>

<fact>

This element holds a single type-name-value fact about the target of a test. The name is a URI. Pre-defined named start with “urn:xccdf:fact”, but tool developers may define additional platform-specific and tool-specific facts.

Content:	string
Cardinality:	0-n

Parent Elements:	target-facts
Attributes:	name , type
Child Elements:	<i>none</i>

The following types are supported: “number”, “string”, and “boolean” (the default).

<fix>

This element may appear as the child of a Rule element, or a rule-result element. When it appears as a child of a Rule element, contains it string data for a command, script, or procedure that should bring the target into compliance with the Rule. It may not contain XHTML formatting. The fix element may contain XCCDF Value substitutions specified with the sub element, or instance name substitution specified with an instance element.

Content	mixed
Cardinality:	0-n
Parent Elements:	Rule, rule-result
Attributes:	id, disruption, platform, reboot, strategy, system
Child Elements:	instance, sub

The fix element supports several attributes that the Rule author can use to provide additional information about the remediation that the fix element contains. The attributes and their permissible values are listed below.

Attribute	Values
id	A local id for the fix, which allows fixtext elements to refer to it. These need not be unique; several fix elements might have the same id but different values for the other attributes.
strategy	A keyword that designates the approach or method that the fix uses. Allowed values: <ul style="list-style-type: none"> • unknown – default, strategy not defined. • configure – adjust target configuration/settings • patch – apply a patch, hotfix, update, etc. • disable – turn off or uninstall a target component • enable – turn on or install a target component • restrict – adjust permission, access rights, filters, or other access restrictions • policy – remediation requires out-of-band adjustments to policies or procedures • combination – strategy is a combination or two or more approaches
disruption	A keyword that designates the potential for disruption or degradation of target operation. Allowed values: <ul style="list-style-type: none"> • unknown – default, disruption not defined. • low – little or no disruption expected • medium – potential for minor or short-lived disruption • high – potential for serious disruption

Attribute	Values
reboot	Boolean – whether remediation will require a reboot or hard reset of the target (‘1’ means reboot required)
system	A URI that identifies the scheme, language, or engine for which the fix is written. Several general URIs are defined, but platform-specific URIs may be expected. (For a list of pre-defined fix system URIs, see Section 9.)
platform	A platform identifier; this should appear on a fix when the content applies to only one platform out of several to which the Rule could apply.

The platform attribute defines which platform the fix is intended for, if its parent Rule applied to multiple platforms. The value of the platform attribute should be one of the platform strings defined for the benchmark. If the fix’s platform attribute is not given, then the fix applies to all platforms to which its enclosing Rule applies.

As a special case, fix elements may also appear as children of a rule-result element in a TestResult. In this case, the fix element should not any child elements, its content should be a simple string. When a fix element is the child of rule-result, it is assumed to have been ‘instantiated’ by the testing tool, and any substitutions or platform selection already made.

<fixtext>

This element, which may only appear as a child of a Rule element, provides text that explains how to bring a target system into compliance with the Rule. Multiple instances may appear in a Rule, with different attribute values.

Content:	mixed
Cardinality:	0-n
Parent Elements:	Rule
Attributes:	xml:lang, fixref, disruption, reboot, strategy, override
Child Elements:	sub, <i>xhtml elements</i>

The fixtext element and its counterpart, the fix element, are fairly complex. They can accept a number of attributes that describe aspects of the remediation. The xml:lang attribute designates the locale for which the text was written; it is expected that fix elements usually will be locale-independent. The following attributes may appear on the fixtext element (for details about most of them, refer to the table under the fix element definition, p. 48).

Attribute	Values
fixref	A reference to the id of a fix element.
strategy	A keyword that designates the approach or method that the fix uses.

Attribute	Values
disruption	A keyword that designates the potential for disruption or degradation of target operation.
reboot	Boolean – whether the remediation described in the fixtext will require a reboot or reset of the target.

The fixtext element may contain XHTML elements, to aid in formatting.

<front-matter>

This element contains textual content intended for use during Document Generation processing only; it is introductory matter that should appear at or near the beginning of the generated document. Multiple instances may appear with different xml:lang values.

Content:	mixed
Cardinality:	0-n
Parent Elements:	Benchmark
Attributes:	xml:lang
Child Elements:	sub, <i>xhtml elements</i>

<ident >

This element contains a string (name) which is a long-term globally meaningful identifier in some naming scheme. The content of the element is the name, and the system attribute contains a URI which designates the organization or scheme that assigned the name (see section 9 for assigned URIs).

Content:	string
Cardinality:	0-n
Parent Elements:	Benchmark
Attributes:	system
Child Elements:	<i>none</i>

<instance>

The instance element may appear in two situations. First, it may appear as part of a TestResult, as a child of a rule-result element; in that situation it contains the name of a target component to which a Rule was applied, in the case of multiply-instantiated rules.

Content:	string
Cardinality:	0-n
Parent Elements:	rule-result
Attributes:	context, parentContext
Child Elements:	<i>none</i>

If the context attribute is omitted, the value of the context defaults to “undefined”. At most one instance child of a rule-result may have a context of “undefined”.

Second, the instance element may appear as part of a Rule, as a child of the fix element. In that situation it represents a place in the fix text where the name of a target component should be substituted, in the case of multiply-instantiated rules.

Content:	<i>none</i>
Cardinality:	0-n
Parent Elements:	fix
Attributes:	context
Child Elements:	<i>none</i>

If the context attribute is omitted, the value of the context defaults to “undefined”.

<lower-bound>

This element may appear zero or more times as a child of a Value element. It is used to constrain value input during tailoring, when the Value’s type is “number”. It contains a number; values supplied by the user for tailoring the benchmark must be no less than this number. This element may have a selector tag attribute, which identifies it for Value refinement by a Profile. If more than one lower-bound element appears as the child of a Value, at most one may omit the selector attribute.

Content:	number
Cardinality:	0-n
Parent Elements:	Value
Attributes:	selector
Child Elements:	<i>none</i>

<match>

This element may appear zero or more times as a child of a Value element. It is used to constrain value input during tailoring. It contains a regular expression that a user’s input for the value must match. This element may have a selector tag attribute, which identifies it for Value refinement by a Profile. If more than one match element appears as the child of a Value, at most one may omit the selector attribute.

Content:	string
Cardinality:	0-n
Parent Elements:	Value
Attributes:	selector
Child Elements:	<i>none</i>

<message>

This element may appear one or more times as a child of a rule-result element inside a TestResult object. It holds one informational or error message from the checking engine.

Content:	string
Cardinality:	0-n

Parent Elements:	rule-result
Attributes:	severity
Child Elements:	<i>none</i>

The severity attribute denotes the seriousness or conditions of the message. In XCCDF 1.1 there are three message severity values: “error”, “warning”, and “info”. These elements do not affect scoring; they are present merely to convey diagnostic information from the checking engine. XCCDF tools that deal with TestResult data may choose to display these messages to the user.

<metadata>

The metadata element may appear one or more times as a child of the Benchmark element. It contains document metadata expressed in XML. The default format for Benchmark document metadata is the Dublin Core Metadata Initiative (DCMI) Simple DC Element specification, as described in [10] and [13]. Tools, especially document generation tools, should be prepared to process Dublin Core metadata in this element.

Content:	element
Cardinality:	0-n
Parent Elements:	Benchmark
Attributes:	<i>none</i>
Child Elements:	<i>non-XCCDF</i> (Dublin Core elements recommended)

Another suitable metadata format for XCCDF benchmarks is the XML description format mandated by NIST for its Security Configuration Checklist Program [11].

Example 8 – Example of Benchmark Metadata Expressed with Dublin Core Elements

```
<cdf: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>
</cdf:metadata>
```

<model>

A specification for a suggested scoring model. This element may only appear as a child of a Benchmark, and has one mandatory attribute: the URI of the scoring model. Some models may need additional parameters; zero or more param elements may appear as children of the model element.

Content:	element
Cardinality:	0-n
Parent Elements:	Benchmark
Attributes:	system
Child Elements:	param

This specification defines the following three scoring model URIs; compliance checking tool developers may define additional models. For more information see Section 3.3.

- **urn:xccdf:scoring:default** – this is the default weighted aggregated model. All tools must support this model, and it is the default for Benchmarks that do not include any other model specifications.
- **urn:xccdf:scoring:flat** – this simple model computes the sum of the weights of rules that pass. All tools should support this model.
- **urn:xccdf:scoring:flat-unweighted** – this simplest of all models simply computes the number of rules that passed. It does not use weights. All tools should support this model.

<new-result>

An override rule result status. This element appears in an override element, inside a rule-result element, in a TestResult object. Its content must be one of the result status values listed in Section 3.2.

Content:	string
Cardinality:	1
Parent Elements:	override
Attributes:	<i>none</i>
Child Elements:	<i>none</i>

<notice>

This string element may only appear as the child of a Benchmark element, and supplies legal notice or copyright text about the benchmark document. It may not contain any child elements. The id attribute must be a unique identifier.

Content:	mixed
Cardinality:	0-n
Parent Elements:	Benchmark
Attributes:	id
Child Elements:	<i>xhtml elements</i>

The notice element may contain XHTML markup to give it internal structure and formatting.

<old-result>

This element holds an overridden rule result status. This element appears in an override element, inside a rule-result element, in a TestResult object. Its content must be one of the result status values listed in section 3.2.

Content:	string
Cardinality:	1
Parent Elements:	override

Attributes: *none*
Child Elements: *none*

<override>

This element may appear only as a child of a rule-result, and represents a human override of a benchmark rule check result. It consists of five parts: a timestamp, the name of the human authority for the override, the old and new result status values, and remark text.

Content: elements
Cardinality: 0-n
Parent Elements: rule-result
Attributes: **time**, authority
Child Elements: old-result, new-result, remark

The example below shows how an override block would appear in a rule-result.

Example 9 – Example of rule-result with an override

```
<cdf:rule-result idref="rule76"
  time="2005-04-26T14:38:19Z" severity="low">
  <cdf:result>pass</cdf:result>
  <cdf:override time="2005-04-26T15:03:20Z" authority="Bob Smith">
    <cdf:old-result>fail</cdf:old-result>
    <cdf:new-result>pass</cdf:new-result>
    <cdf:remark>
      Manual inspection showed this rule be 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'.
    </cdf:remark>
  </cdf:override>
  <cdf:instance context="registry">
    HKLM\SOFTWARE\Policies
  </cdf:instance>
  <cdf:check system="http://www.mitre.org/XMLSchema/oval">
    <cdf:check-content-ref href="oval-out.xml" name="OVAL123"/>
  </cdf:check>
</cdf:rule-result>
```

Note: if an override is added to a rule-result, then it will break any digital signature applied to the enclosing TestResult object.

<param>

This element may appear only as a child of a model element. It supplies parameter that the compliance checking tool will need when computing the score using that model. None of the scoring models defined in the XCCDF 1.1 specification require parameters, but proprietary models may.

Content:	string
Cardinality:	0-n
Parent Elements:	model
Attributes:	name
Child Elements:	<i>none</i>

Param elements with equal values for the name attribute may not appear as children of the same model element.

<plain-text>

This element holds a re-usable chunk of text for a Benchmark. It may be used anywhere that XHTML content or the XCCDF sub element may be used. Each plain-text definition must have a unique id.

Content:	string
Cardinality:	0-n
Parent Elements:	Benchmark,
Attributes:	id
Child Elements:	<i>none</i>

Note that plain-text definitions may only appear as children of Benchmark. The ids on plain-text definitions must not be equal to any of the ids on Value, Group, or Rule objects.

<platform>

The platform element specifies a target platform for which the benchmark or a particular Rule applies. This element has no content; the platform identifier appears as the attribute “idref”. Multiple platform elements may appear as children of a Benchmark, Group, or Rule, if the benchmark or item is suitable for multiple kinds of target systems.

Content:	<i>none</i>
Cardinality:	0-n
Parent Elements:	Benchmark, Group, Rule, Value, TestResult
Attributes:	idref , override
Child Elements:	<i>none</i>

The platform element is optional. It refers to a platform defined in a platform-definitions element of the Benchmark. The structure of a platform definition is specified in the XCCDF platform schema (see [12]). The override attribute may appear only when the platform element is a child of a Rule or Group.

<platform-definitions>, <Platform-Specification>

Each of these elements contains information about platforms to which the Benchmark may apply. At most one of the two may appear, as a child of the Benchmark element. The <platform-definitions> element contains a set of platform component and platform

definitions using the CIS platform schema; it is permitted in XCCDF 1.1 for compatibility with 1.0. The <Platform-Specification> element contains a set of Fact and Platform definitions using the XCCDF-P schema (see [16]). The platform definitions listed under this element, using either schema, each have unique ids. Platform ids are used in platform and fix elements to designate the system or product to which a portion of a Benchmark applies.

Content:	elements
Cardinality:	0-1
Parent Elements:	Benchmark
Attributes:	<i>none</i>
Child Elements:	<i>special</i>

<profile>

This element specifies the Benchmark Profile used in applying a benchmark; it can appear only as a child of a TestResult element.

Content:	<i>none</i>
Cardinality:	0-1
Parent Elements:	TestResult
Attributes:	idref
Child Elements:	<i>none</i>

<profile-note>

This element holds descriptive text about how one or more Profiles affect a Rule. It can appear only as a child of the Rule element.

Content:	mixed
Cardinality:	0-n
Parent Elements:	Rule
Attributes:	tag , xml:lang
Child Elements:	sub, <i>xhtml elements</i>

The mandatory ‘tag’ attribute holds an identifier; Profiles can refer to this identifier using the ‘note-tag’ attribute.

<question>

This element specifies an interrogatory string with which to prompt the user during tailoring. It may also be included into a generated document. Note that this element may not contain any XCCDF child elements nor may it contain XHTML formatting elements. Multiple instances may appear with different xml:lang attributes.

Content:	string
Cardinality:	0-n
Parent Elements:	Group, Rule, Value

Attributes:	xml:lang, override
Child Elements:	<i>none</i>

For Rule and Group objects, the question text should be simple binary (yes/no) question, because tailoring for Rules and Groups is for selection only. For Value objects, the question should reflect the designed data value needed for tailoring.

<rationale>

This element, which may appear as a child of a Group or Rule element, provides text that explains why that Group or Rule is important to the security of a target platform.

Content:	mixed
Cardinality:	0-n
Parent Elements:	Group, Rule
Attributes:	xml:lang, override
Child Elements:	sub, <i>xhtml elements</i>

<rear-matter>

This element contains textual content intended for use during Document Generation processing only; it is concluding material that should appear at or near the end of the generated document. Multiple instances may appear with different xml:lang attributes.

Content:	mixed
Cardinality:	0-n
Parent Elements:	Benchmark
Attributes:	xml:lang
Child Elements:	sub, <i>xhtml elements</i>

<reference>

This element provides supplementary descriptive text for a Benchmark, Rule, Group, or Value. It may have a simple string value, or a value consisting of simple Dublin Core elements as described in [13]. It may also have an attribute, href, giving a URL for the referenced resource. Multiple reference elements may appear; a document generation processing tool may concatenate them, or put them into a reference list, and may choose to number them.

Content:	string or elements
Cardinality:	0-n
Parent Elements:	Benchmark, Group, Rule, Value, Profile
Attributes:	xml:lang, href
Child Elements:	<i>none or Dublin Core Elements</i>

References should be given as Dublin Core descriptions, a bare string is allowed for simplicity. If a bare string appears, then it is taken to be the string content for a Dublin Core 'title' element. For more information, consult [10].

<refine-rule>

This element specifies the selector tag to be applied when evaluating a Rule during use of a particular Profile. It has two mandatory attributes: the id of a Rule or item cluster, and the id of a Rule check selector tag.

Content:	<i>none</i>
Cardinality:	0-n
Parent Elements:	Profile
Attributes:	idref, selector
Child Elements:	<i>none</i>

The idref attribute must match the id attribute of a Value, or the cluster-id of one or more Items in the Benchmark. The idref attribute values of sibling refine-rule element children of a Profile must be different.

Selector tags apply only to the check children of a Rule. If the selector tag specified in a refine-rule element in a Profile does not match any of the selectors specified on any of the check children of a Rule, then the check child element without a selector attribute must be used.

<refine-rule>

This element adjusts the tailoring selector, weight, severity, and role properties of a Rule. It has four attributes: the id of a Rule (mandatory), and new values for weight, severity, and role (all optional).

Content:	<i>none</i>
Cardinality:	0-n
Parent Elements:	Profile
Attributes:	idref, weight, severity, role
Child Elements:	<i>none</i>

The idref attribute must match the id attribute of a Rule, or a cluster-id of one or more Items in the Benchmark. The idref attribute values of sibling refine-rule element children of a Profile must be different.

<refine-value>

This element specifies the selector tag to be applied when tailoring a Value during use of a particular Profile. It has two mandatory attributes: the id of a Value or item cluster, and the id of a Value selector tag.

Content:	<i>none</i>
Cardinality:	0-n
Parent Elements:	Profile
Attributes:	idref, selector
Child Elements:	<i>none</i>

The `idref` attribute must match the `id` attribute of a `Value`, or a `cluster-id` of one or more `Items` in the Benchmark. The `idref` attribute values of sibling `refine-value` element children of a `Profile` must be different.

Selector tags apply to the following child elements of `Value`: `choices`, `default`, `value`, `match`, `lower-bound`, and `upper-bound`. If the selector tag specified in a `refine-value` element in a `Profile` does not match any of the selectors specified on any of the `Value` children, then the child with no selector tag is used. The example below illustrates how selector tags and the `refine-value` element work.

Example 10 – Example of Profile `refine-value` Selector Tags

```
<cdf:Value id="pw-length" type="number" operator="equals">
  <cdf:title>Minimum password length policy</cdf:title>
  <cdf:value>8</cdf:value>
  <cdf:value selector="high">14</cdf:value>
  <cdf:lower-bound>8</cdf:lower-bound>
  <cdf:lower-bound selector="high">12</cdf:lower-bound>
</cdf:Value>
<cdf:Profile id="enterprise-internet">
  <cdf:title>Enterprise internet server profile</cdf:title>
  <cdf:refine-value idref="pw-length" selector="high"/>
</cdf:Profile>
<cdf:Profile id="home">
  <cdf:title>Home host profile</cdf:title>
</cdf:Profile>
```

<remark>

The `remark` element may appear as the child of a `TestResult` or `override` element; it contains a textual remark about the test.

Content:	string
Cardinality:	0-n (<code>TestResult</code>), 1 (<code>override</code>)
Parent Elements:	<code>TestResult</code> , <code>override</code>
Attributes:	<code>xml:lang</code>
Child Elements:	<i>none</i>

The `remark` content may not contain any XHTML tags or other structure, it must be a plain string.

<requires>

The `requires` element may be a child of any `Group` or `Rule`, and it specifies the `id` property of another `Group`, `Rule`, or `Value` which must be selected in order for this one to be selected. In a sense, the `requires` element is the opposite of the `conflicts` element. Each `requires` element specifies a single required `Item` by its `id`, using the `idref` attribute; if the semantics of the benchmark need multiple required items, then multiple `requires` elements may appear. A `requires` element must be empty.

Content:	<i>none</i>
Cardinality:	0-n
Parent Elements:	Group, Rule
Attributes:	idref
Child Elements:	<i>none</i>

<result>

This simple element holds the verdict of apply a Benchmark Rule to a target or component of a target. It may have only one of four values: “pass”, “fail”, “error”, or “unknown”.

Content:	string
Cardinality:	1
Parent Elements:	rule-result
Attributes:	<i>none</i>
Child Elements:	<i>none</i>

<rule-result>

This element holds the result of applying a Rule from the benchmark to a target system or component of a target system. It may only appear as the child of a TestResult element.

Content:	elements
Cardinality:	0-n
Parent Elements:	TestResult
Attributes:	idref , role, time, severity, version
Child Elements:	result , override, ident, message, instance, fix, check

The idref property of a rule-result element must refer to a Rule element in the Benchmark. The result child element expresses the result (pass, fail, error, etc..) of applying the Rule to the target system. If the Rule is multiply instantiated, the instance elements indicate the particular system component. If present, the override element provides information about a human override of a computed result status value.

<score>

This element contains the weighted score for a benchmark test, as a real number. Scoring models are defined in Section 3.3. This element may only appear as a child of a TestResult element.

Content:	string (non-negative number)
Cardinality:	1-n
Parent Elements:	TestResult
Attributes:	system, maximum
Child Elements:	<i>none</i>

The system attribute, a URI, identifies the scoring model (see the description of the model element on page 52 for a list of pre-defined models). If the system attribute does not appear, then the model used was the default model. The maximum attribute, a real number, gives the maximum possible value of the score for this benchmark test. If the maximum attribute does not appear, then it is taken to have a value of 100.

<select>

This element is part of a Profile; it overrides the selected attribute of a Rule or Group. Two attributes must be given with this element: the id of a Rule or Group (idref), and a boolean value (selected). If the boolean value is given as true, then the Rule or Group is selected in this Profile, otherwise it is unselected for this Profile.

Content:	<i>none</i>
Cardinality:	0-n
Parent Elements:	Profile
Attributes:	idref, selected
Child Elements:	<i>none</i>

The idref attribute 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. The idref attribute values of sibling select element children of a Profile must be different.

<set-value>

This element specifies a value for a Value object. It may appear as part of a Profile, in that case it overrides the value property of a Value object. It may appear as part of a TestResult, in that case it supplies the value used in the test. This element has one mandatory attributes and no child elements.

Content:	string
Cardinality:	0-n
Parent Elements:	Profile, TestResult
Attributes:	idref
Child Elements:	<i>none</i>

In the content of a Profile, the identifier given for the idref attribute may be a cluster id, in which case it applies only to the Value item members of the cluster; in the context of a TestResult, the identifier must match the id of a Value object in the Benchmark. The idref attribute values of sibling set-value element children of a Profile must be different.

<signature>

This element can hold an enveloped digital signature expressed according to the XML Digital Signature standard [8]. This element takes no attributes, and must contain exactly one element from the XML-Signature namespace.

Content:	elements
Cardinality:	0-1

Parent Elements:	Benchmark, Rule, Group, Value, Profile, TestResult
Attributes:	<i>none</i>
Child Elements	<i>Signature</i> (in XML-Signature namespace, see [8])

At most one enveloped signature can appear in an XCCDF benchmark document. If multiple signatures are needed, others must be detached signatures.

<source>

The source element contains a URI indicating where a tailoring or benchmarking tool might obtain the value, or information about the value, for a Value object. XCCDF does not attach any meaning to the URI, it may be an arbitrary community or tool-specific value, a pointer directly to a resource, or a reference to a platform Fact (see [16]).

Content:	<i>none</i>
Cardinality:	0-n
Parent Elements:	Value
Attributes:	uri
Child Elements:	<i>none</i>

<status>

This element provides a revision or standardization status for a benchmark, along with a date at which the benchmark attained that status. It must appear once in a Benchmark object, and may appear once in any Item. If an Item does not have its own status element, its status is that of its parent element. The permitted string values for status are “accepted”, “deprecated”, “draft”, “interim” and “incomplete”.

Content:	string (enumerated choices)
Cardinality:	0-n
Parent Elements:	Benchmark, Rule, Group, Value
Attributes:	date
Child Elements:	<i>none</i>

A Benchmark must have at least one status child element.

<sub>

This element represents a reference to a parameter value that may be set during tailoring. The element never has any content, and must have its single attribute, value. The value attribute must equal the id attribute of a Value object or plain-text definition.

Content:	<i>none</i>
Cardinality:	0-n
Parent Elements:	description, fix, fixtext, front-matter, rationale, rear-matter, title, warning
Attributes:	idref
Child Elements:	<i>none</i>

<target>

This element gives the name or description of a target system to which a Benchmark test was applied. It may only appear as a child of a TestResult element.

Content:	string
Cardinality:	1
Parent Elements:	TestResult
Attributes:	<i>none</i>
Child Elements:	<i>none</i>

<target-address>

This element gives the network address of a target system to which a Benchmark test was applied. It may only appear as a child of a TestResult element.

Content:	string
Cardinality:	0-n
Parent Elements:	TestResult
Attributes:	<i>none</i>
Child Elements:	<i>none</i>

<target-facts>

The TestResult object must be able to hold an arbitrary set of facts about the target of a test. This element holds those facts, each one of which is a fact element. It is an optional member of TestResult.

Content:	string
Cardinality:	0-1
Parent Elements:	TestResult
Attributes:	<i>none</i>
Child Elements:	fact

<title>

This element provides the descriptive title for a Benchmark, Rule, Group, or Value. It has no attributes. Multiple instances may appear with different languages (different values of the xml:lang attribute).

Content:	string
Cardinality:	0-n
Parent Elements:	Benchmark, Value, Group, Rule, Profile, TestResult
Attributes:	xml:lang, override
Child Elements:	<i>none</i>

This element may not contain XHTML markup.

<upper-bound>

This element may appear zero or more times as a child of a Value element. It is used to constrain value input during tailoring, when the Value's type is "number". It contains a number; values supplied by the user for tailoring the benchmark must be no greater than this number. This element may have a selector tag attribute, which identifies it for Value refinement by a Profile. If more than one upper-bound element appears as the child of a Value, at most one may omit the selector attribute.

Content:	number
Cardinality:	0-n
Parent Elements:	Value
Attributes:	selector
Child Elements:	<i>none</i>

<value>

This string element is used to hold the value of a Value object. It must appear as the child of a Value element, and no child elements. This element may have a selector tag attribute, which identifies it for Value refinement by a Profile. This element may appear more than once, but at most one of the sibling instances of this element may omit the selector tag.

Content:	string
Cardinality:	1-n
Parent Elements:	Value
Attributes:	selector
Child Elements:	<i>none</i>

<version>

This element gives a version number for a Benchmark, Group, Rule, Value, or Profile. The version number content may be any string. This element allows an optional time attribute, which is a timestamp of when the Object was defined. This element also allows an optional update attribute, which should be the URI specifying where updates to the Object may be obtained.

Content:	string
Cardinality:	1 (Benchmark), 0-1 (all others)
Parent Elements:	Benchmark, Group, Rule, Value, Profile
Attributes:	time, update
Child Elements:	<i>none</i>

<warning>

This element provides supplementary descriptive text for a Benchmark, Rule, Group, or Value. It has no attributes. Multiple warning elements may appear; processing tools should concatenate them for generating reports or documents (see also next section).

Content:	mixed
Cardinality:	0-n
Parent Elements:	Benchmark, Group, Rule, Value
Attributes:	xml:lang, override
Child Elements:	sub, <i>xhtml elements</i>

This element is 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”). Processing tools may present this information specially in generated documents.

4.3. Handling Text and String Content

This sub-section provides additional information about how XCCDF processing tools must handle textual content in Benchmarks.

XHTML Formatting and Locale

Some text-valued XCCDF elements may contain formatting specified with elements from the XHTML Core Recommendation.

Many of the string and textual elements of the XCCDF are listed as appearing multiple times under the same parent element. These elements, listed below, can have an `xml:lang` attribute that specifies the natural language locale for which they are written (e.g. “en” for English, “fr” for French, etc.). A processing tool should employ these attributes when possible during tailoring, document generation, and producing compliance reports, to create localized output. An example of using the `xml:lang` attribute is shown below.

Example 11 – A Simple Value Object with Questions in Different Languages

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

Multiple values for the same property in a single Item are handled differently, as described below. Multiple instances with different values of their `xml:lang` attribute are always permitted; an item with no value for the `xml:lang` attribute are taken to have the same language as the benchmark itself (as given by the `xml:lang` attribute on the Benchmark element).

Elements	Inheritance Behavior
description, title, fixtext, rationale, question, front-matter, rear-matter	At most one instance per language; inherited values with the same language get replaced.
warning, reference, notice	Multiple instances treated as an ordered list; inherited instances prepended to the list.

The platform element may also appear multiple times, each with a different id, to express the notion that a Rule, Group, Benchmark, or Profile applies to several different products or systems.

String Substitution and Reference Processing

There are three kinds of string substitution and one kind of reference processing that XCCDF document generation and reporting tools must support.

1. XCCDF **sub** element -

The sub element supports substitution of information from a Value object, or the string content of a plain-text definition. The formatting for a sub element reference to a Value object is implementation-dependent for document generation, as described in Section 3.3. Formatting for a sub element reference to a plain-text definition is very simple: the string content of the plain-text definition replaces the sub element.

2. XHTML **object** element -

The object element supports substitutions of a variety of information from another Item or Profile, or the string content of a plain-text definition. To avoid possible conflicts with uses of an XHTML object that should not be processed specially, all XCCDF object references must be a relative URI beginning with “#xccdf:”. The following URI values can be used to refer to things from an "object" element, using the "data" attribute:

- **#xccdf:value:id**
Insert the value of the thing with id *id*: the value for a Value or fact, the string contents for a plain-text block. When a URI of this form is used, the value of the reference should be substituted for the entire "object" element and its content (if any). In keeping with the standard semantics of XHTML, 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 "title" child of the Item with id *id*. Use the current language value locale setting, if any. When a URI of this form is used, the title string should be substituted for the entire "object" element and its content (if any). In keeping with the standard semantics of XHTML, if the *id* cannot be resolved, then the textual content of the "object" element should be retained.

3. XHTML anchor (**a**) element -

The anchor element can be used to create an intra-document link to another XCCDF Item or Profile. To avoid possible conflicts with uses of the XHTML

anchor element that should not be processed specially, all XCCDF anchor references must be a relative URI beginning with “#xccdf:” The following URI values can be used to refer to things from an "a" element, using the "href" attribute:

- **#xccdf:link:id**
Create 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.

5. Conclusions

The XCCDF specification defines a means for expressing security benchmarks in a way that should foster development of interoperable tools and content. It is designed to permit the same document to serve in several roles:

- source code for generation of publication documents and hardcopy,
- script for eliciting local security policy settings and values from a user,
- structure for containing and organizing code that drives system analysis and configuration checking engines,
- source code for text to appear in security policy compliance reports,
- a record of a benchmark test, including the results of applying various rules,
- structure for expressing compliance scoring/weighting decisions.

XCCDF 1.1 was designed as a compatible extension of 1.0, based on suggestions from early adopted and potential users. Many features have been added, but every valid 1.0 document should be a valid 1.1 document, once the namespace is adjusted. The forward compatibility will help ensure that benchmark content written for XCCDF 1.0 will not be made obsolete by the 1.1 specification.

Adoption of a common format should permit security professionals, security tool vendors, and system auditors to exchange information more quickly and precisely, and also permit greater automation of security testing and configuration checking.

6. Appendix A – XCCDF Schema

The XML Schema below describes XCCDF in a manner that should allow automatic validation of most aspects of the format. It is not possible to express all of the constraints that XCCDF imposes in a Schema, unfortunately, and a few of the constraints that it is possible to express have been omitted for simplicity.

Whether to validate is an implementation decision left to tool developers, but it is strongly recommended.

XCCDF Schema 1.1

```
<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:cdf="http://checklists.nist.gov/xccdf/1.1"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:cisp="http://www.cisecurity.org/xccdf/platform/0.2.3"
  xmlns:cdfp="http://checklists.nist.gov/xccdf-p/1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-Instance"
  targetNamespace="http://checklists.nist.gov/xccdf/1.1"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified">

  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      This schema defines the eXtensible Configuration Checklist
      Description Format (XCCDF), a data format for defining
      security benchmarks and checklists, and for recording
      the results of applying such benchmarks.
      For more information, consult the specification
      document, "Specification for the Extensible Configuration
      Checklist Description Format", version 1.1.

      This schema was developed by Neal Ziring, with ideas and
      assistance from David Waltermire. The following helpful
      individuals also contributed ideas to the definition
      of this schema: David Proulx, Andrew Buttner,
      Ryan Wilson, Matthew Kerr, Stephen Quinn.
      <version date="5 December 2005">1.1.0.18</version>
    </xsd:documentation>
  </xsd:annotation>

  <!-- Import base XML namespace -->
  <xsd:import namespace="http://www.w3.org/XML/1998/namespace"
    schemaLocation="xml.xsd">
    <xsd:annotation>
      <xsd:documentation xml:lang="en">
        Import the XML namespace because this schema uses
        the xml:lang and xml:base attributes.
      </xsd:documentation>
    </xsd:annotation>
  </xsd:import>

  <!-- Import Dublin Core metadata namespace -->
  <xsd:import namespace="http://purl.org/dc/elements/1.1/"
```

```

        schemaLocation="simpledc20021212.xsd">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
                Import the simple Dublin Core namespace because
                this schema uses it for benchmark metadata and for
                references.
            </xsd:documentation>
        </xsd:annotation>
    </xsd:import>

    <!-- Import CIS platform specification namespace -->
    <!-- This will be replaced by a reference to the XCCDF-P schema -->
    <xsd:import namespace="http://www.cisecurity.org/xccdf/platform/0.2.3"
        schemaLocation="platform-0.2.3.xsd">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
                Import the CIS platform schema, which we use for
                describing target IT platforms in the Benchmark.
                The CIS platform schema was designed by David
                Waltermire.
            </xsd:documentation>
        </xsd:annotation>
    </xsd:import>

    <!-- Import XCCDF-P platform specification namespace -->
    <xsd:import namespace="http://checklists.nist.gov/xccdf-p/1.0"
        schemaLocation="xccdfp-1.0.xsd">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
                Import the XCCDF-P platform schema, which we use
                for describing target IT platforms in the Benchmark.
                The CIS platform schema was designed by Neal Ziring
                using ideas and concepts developed by DISA, CIS, and
                others. For more information consult the document
                "Specification for the Extensible Configuration Checklist
                Description Format Platform Facts Definition (XCCDF-P)".
            </xsd:documentation>
        </xsd:annotation>
    </xsd:import>

    <!-- ***** -->
    <!-- ***** Benchmark Element ***** -->
    <!-- ***** -->
    <xsd:element name="Benchmark">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
                The benchmark tag is the top level element representing a
                complete security checklist, including descriptive text
                and test items.
            </xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="cdf:status" minOccurs="1" maxOccurs="1"/>
                <xsd:element name="title" type="cdf:textType"
                    minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element name="description" type="cdf:htmlTextWithSubType"
                    minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element name="notice" type="cdf:noticeType"

```

```

        minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="front-matter" type="cdf:htmlTextWithSubType"
    minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="rear-matter" type="cdf:htmlTextWithSubType"
    minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="reference" type="cdf:referenceType"
    minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="plain-text" type="cdf:plainTextType"
    minOccurs="0" maxOccurs="unbounded"/>
<xsd:choice minOccurs="0" maxOccurs="1">
    <xsd:element ref="cisp:platform-definitions"
        minOccurs="0" maxOccurs="1"/>
    <xsd:element ref="cdfp:Platform-Specification"
        minOccurs="0" maxOccurs="1"/>
</xsd:choice>
<xsd:element name="platform" type="cdf:idrefType"
    minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="version" type="cdf:versionType"
    minOccurs="1" maxOccurs="1"/>
<xsd:element name="metadata" type="cdf:metadataType"
    minOccurs="0" maxOccurs="unbounded"/>
<xsd:element ref="cdf:model"
    minOccurs="0" maxOccurs="unbounded"/>
<xsd:element ref="cdf:Profile"
    minOccurs="0" maxOccurs="unbounded"/>
<xsd:element ref="cdf:Value"
    minOccurs="0" maxOccurs="unbounded"/>
<xsd:choice minOccurs="0" maxOccurs="unbounded">
    <xsd:element ref="cdf:Group"/>
    <xsd:element ref="cdf:Rule"/>
</xsd:choice>
<xsd:element ref="cdf:TestResult"
    minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="signature" type="cdf:signatureType"
    minOccurs="0" maxOccurs="1"/>
</xsd:sequence>
<xsd:attribute name="id" type="xsd:NCName" use="optional"/>
<!-- the 'Id' attribute is needed for XML-Signature -->
<xsd:attribute name="Id" type="xsd:ID" use="optional"/>
<xsd:attribute name="resolved" type="xsd:boolean"
    default="false" use="optional"/>
<xsd:attribute ref="xml:lang"/>
</xsd:complexType>

<xsd:key name="noticeIdKey">
    <xsd:annotation><xsd:documentation xml:lang="en">
        Legal notices must have unique id values.
    </xsd:documentation></xsd:annotation>
    <xsd:selector xpath="cdf:notice"/>
    <xsd:field xpath="@id"/>
</xsd:key>

<xsd:key name="itemIdKey">
    <xsd:annotation><xsd:documentation xml:lang="en">
        Items must have unique id values, and also they
        must not collide
    </xsd:documentation></xsd:annotation>
    <xsd:selector xpath="./cdf:Value|./cdf:Group|./cdf:Rule"/>
    <xsd:field xpath="@id"/>

```

```

</xsd:key>

<xsd:key name="modelSystemKey">
  <xsd:annotation><xsd:documentation xml:lang="en">
    Model system attributes must be unique.
  </xsd:documentation></xsd:annotation>
  <xsd:selector xpath="./cdf:model"/>
  <xsd:field xpath="@system"/>
</xsd:key>

<xsd:key name="valueIdKey">
  <xsd:annotation><xsd:documentation xml:lang="en">
    Value item ids are special keys, need this for
    the valueIdKeyRef keyref below.
  </xsd:documentation></xsd:annotation>
  <xsd:selector xpath="./cdf:Value"/>
  <xsd:field xpath="@id"/>
</xsd:key>

<xsd:key name="ruleIdKey">
  <xsd:annotation><xsd:documentation xml:lang="en">
    Rule items have a unique key, we need
    this for the ruleIdKeyRef keyref below.
    (Rule key refs are used by rule-results.)
  </xsd:documentation></xsd:annotation>
  <xsd:selector xpath="./cdf:Rule"/>
  <xsd:field xpath="@id"/>
</xsd:key>

<xsd:key name="selectableItemIdKey">
  <xsd:annotation><xsd:documentation xml:lang="en">
    Group and Rule item ids are special keys, we
    need this for the requiresIdKeyRef keyref below.
  </xsd:documentation></xsd:annotation>
  <xsd:selector xpath="./cdf:Group | ./cdf:Rule"/>
  <xsd:field xpath="@id"/>
</xsd:key>

<xsd:key name="profileIdKey">
  <xsd:annotation><xsd:documentation xml:lang="en">
    Profile objects have a unique id, it is used
    for extension, too.
  </xsd:documentation></xsd:annotation>
  <xsd:selector xpath="./cdf:Profile"/>
  <xsd:field xpath="@id"/>
</xsd:key>

<xsd:key name="platformDefnKey">
  <xsd:annotation><xsd:documentation xml:lang="en">
    Platform-definitions have a unique id, it is used
    from the platform element and fix element. This
    definition allows use of either the CIS Platform
    specification, or the newer XCCDF-P Platform
    specification.
  </xsd:documentation></xsd:annotation>
  <xsd:selector
    xpath="./cisp:platform-definitions/cisp:platform-definition |
    ./cdfp:Platform-Specification/cdfp:Platform"/>
  <xsd:field xpath="@id"/>

```

```

</xsd:key>

<xsd:keyref name="valueIdKeyRef" refer="cdf:valueIdKey">
  <xsd:annotation><xsd:documentation xml:lang="en">
    Check-export elements must reference existing values.
  </xsd:documentation></xsd:annotation>
  <xsd:selector xpath="."/><xsd:check/cdf:check-export"/>
  <xsd:field xpath="@value-id"/>
</xsd:keyref>

<xsd:keyref name="subValueKeyRef" refer="cdf:valueIdKey">
  <xsd:annotation><xsd:documentation xml:lang="en">
    Sub elements must reference existing Value ids.
  </xsd:documentation></xsd:annotation>
  <xsd:selector xpath="."/><xsd:sub"/>
  <xsd:field xpath="@value"/>
</xsd:keyref>

<xsd:keyref name="ruleIdKeyRef"
  refer="cdf:ruleIdKey">
  <xsd:annotation><xsd:documentation xml:lang="en">
    The rule-result element idref must refer to an
    existing Rule.
  </xsd:documentation></xsd:annotation>
  <xsd:selector xpath="."/><xsd:TestResult/cdf:rule-result"/>
  <xsd:field xpath="@idref"/>
</xsd:keyref>

<xsd:keyref name="requiresIdKeyRef"
  refer="cdf:selectableItemIdKey">
  <xsd:annotation><xsd:documentation xml:lang="en">
    The requires element idref must refer to an existing
    Group or Rule.
  </xsd:documentation></xsd:annotation>
  <xsd:selector xpath="."/><xsd:requires"/>
  <xsd:field xpath="@idref"/>
</xsd:keyref>

<xsd:keyref name="profileIdKeyRef"
  refer="cdf:profileIdKey">
  <xsd:annotation><xsd:documentation xml:lang="en">
    The requires a profile element in a TestResult
    element to refer to an existing Profile
  </xsd:documentation></xsd:annotation>
  <xsd:selector xpath="."/><xsd:TestResult/profile"/>
  <xsd:field xpath="@idref"/>
</xsd:keyref>

<xsd:keyref name="platformIdKeyRef"
  refer="cdf:platformDefnKey">
  <xsd:annotation><xsd:documentation xml:lang="en">
    The platform element idref attribute must refer to
    an existing cisp:platform-definition id or
    cdpf:Platform id.
  </xsd:documentation></xsd:annotation>
  <xsd:selector xpath="."/><xsd:platform"/>
  <xsd:field xpath="@idref"/>
</xsd:keyref>

```



```

    <xsd:keyref name="fixPlatformIdKeyRef"
      refer="cdf:platformDefnKey">
      <xsd:annotation><xsd:documentation xml:lang="en">
        The fix element attribute platform must refer to an
        existing platform-definition.
      </xsd:documentation></xsd:annotation>
      <xsd:selector xpath="//cdf:Rule/cdf:fix"/>
      <xsd:field xpath="@platform"/>
    </xsd:keyref>

  </xsd:element>

  <xsd:complexType name="noticeType" mixed="true">
    <xsd:annotation>
      <xsd:documentation xml:lang="en">
        Data type for legal notice element that has text
        content and a unique id attribute.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:any namespace="http://www.w3.org/1999/xhtml"
        minOccurs="0" maxOccurs="unbounded"
        processContents="skip"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:NCName"/>
    <xsd:attribute ref="xml:base"/>
    <xsd:attribute ref="xml:lang"/>
  </xsd:complexType>

  <xsd:complexType name="plainTextType">
    <xsd:annotation>
      <xsd:documentation xml:lang="en">
        Data type for a reusable text block, with an
        unique id attribute.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="id" type="xsd:NCName" use="required"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>

  <xsd:complexType name="referenceType" mixed="true">
    <xsd:annotation>
      <xsd:documentation xml:lang="en">
        Data type for a reference citation, an href
        URL attribute (optional), with content of text
        or simple Dublin Core elements.
        Elements of this type can also have an override
        attribute to help manage inheritance.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:any namespace="http://purl.org/dc/elements/1.1/"
        processContents="lax"
        minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="href" type="xsd:anyURI"/>
  </xsd:complexType>

```

```

    <xsd:attribute name="override" type="xsd:boolean"/>
</xsd:complexType>

<xsd:complexType name="signatureType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      XML-Signature over the Benchmark; note that this
      will always be an 'enveloped' signature, so the
      single element child of this element should be
      dsig:Signature.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:any namespace="http://www.w3.org/2000/09/xmldsig#"
      processContents="skip"
      minOccurs="1" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="metadataType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      Metadata for the Benchmark, should be Dublin Core
      or some other well-specified and accepted metadata
      format. If Dublin Core, then it will be a sequence
      of simple Dublin Core elements.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:choice minOccurs="1" maxOccurs="1">
      <xsd:any namespace="http://purl.org/dc/elements/1.1/"
        minOccurs="1" maxOccurs="unbounded"/>
      <!-- the namespace URI below is from a draft version of the
        schema designed to support the checklist metadata
        requirements from NIST special pub 800-70. -->
      <xsd:any namespace="http://checklists.nist.gov/sccf/0.1"
        processContents="skip"
        minOccurs="1" maxOccurs="unbounded"/>
    </xsd:choice>
  </xsd:sequence>
</xsd:complexType>

<!-- ***** -->
<!-- ***** Global elements and types ***** -->
<!-- ***** -->
<xsd:element name="status">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      The acceptance status of an Item with an optional date attribute
      that signifies the date of the status change.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="cdf:statusType">
        <xsd:attribute name="date" type="xsd:date"
          use="optional"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>

```

```

    </xsd:complexType>
</xsd:element>

<xsd:element name="model">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      A suggested scoring model for a Benchmark, also
      encapsulating any parameters needed by the model.
      Every model is designated with a URI, which
      appears here as the system attribute.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="param" type="cdf:paramType"
        minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="system" type="xsd:anyURI"
      use="required"/>
  </xsd:complexType>

  <xsd:key name="paramNameKey">
    <xsd:annotation><xsd:documentation xml:lang="en">
      Parameter names must be unique.
    </xsd:documentation></xsd:annotation>
    <xsd:selector xpath="./cdf:param"/>
    <xsd:field xpath="@name"/>
  </xsd:key>
</xsd:element>

<xsd:complexType name="paramType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      Type for a scoring model parameter: a name and a
      string value.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="name" type="xsd:NCName" use="required"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:simpleType name="statusType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      The possible status codes for an Benchmark or Item to be
      inherited from the parent element if it is not defined.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="accepted"/>
    <xsd:enumeration value="deprecated"/>
    <xsd:enumeration value="draft"/>
    <xsd:enumeration value="incomplete"/>
    <xsd:enumeration value="interim"/>
  </xsd:restriction>

```

```

</xsd:simpleType>

<xsd:complexType name="versionType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      Type for a version number, with a timestamp attribute
      for when the version was made.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="time" type="xsd:dateTime" use="optional"/>
      <xsd:attribute name="update" type="xsd:anyURI" use="optional"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<!-- ***** -->
<!-- ***** Text Types ***** -->
<!-- ***** -->
<xsd:complexType name="textType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      Type for a string with an xml:lang attribute.
      Elements of this type can also have an override
      attribute to help manage inheritance.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute ref="xml:lang"/>
      <xsd:attribute name="override" type="xsd:boolean"
        use="optional" default="0"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="htmlTextType" mixed="true">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      Type for a string with XHTML elements and xml:lang attribute.
      Elements of this type can also have an override
      attribute to help manage inheritance.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:any namespace="http://www.w3.org/1999/xhtml"
      minOccurs="0" maxOccurs="unbounded"
      processContents="skip"/>
  </xsd:sequence>
  <xsd:attribute ref="xml:lang"/>
  <xsd:attribute name="override" type="xsd:boolean"
    use="optional" default="0"/>
</xsd:complexType>

<xsd:complexType name="htmlTextWithSubType" mixed="true">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      Type for a string with embedded Value substitutions

```

```

        and XHTML elements, and an xml:lang attribute.
        Elements of this type can also have an override
        attribute to help manage inheritance.
        [Note: this definition is rather loose, it allows
        anything whatsoever to occur insides XHTML tags inside
        here. Further, constraints of the XHTML schema do not
        get checked! It might be possible to solve this using
        XML Schema redefinition features.]
    </xsd:documentation>
</xsd:annotation>
<xsd:choice minOccurs="0" maxOccurs="unbounded">
    <xsd:element name="sub" type="cdf:idrefType"/>
    <xsd:any namespace="http://www.w3.org/1999/xhtml"
        processContents="skip"/>
</xsd:choice>
<xsd:attribute ref="xml:lang"/>
<xsd:attribute name="override" type="xsd:boolean"
    use="optional" default="0"/>
</xsd:complexType>

<xsd:complexType name="profileNoteType" mixed="true">
    <xsd:annotation>
        <xsd:documentation xml:lang="en">
            Type for a string with embedded Value substitutions
            and XHTML elements, an xml:lang attribute, and a
            profile-note tag.
        </xsd:documentation>
    </xsd:annotation>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="sub" type="cdf:idrefType"/>
        <xsd:any namespace="http://www.w3.org/1999/xhtml"
            processContents="skip"/>
    </xsd:choice>
    <xsd:attribute ref="xml:lang"/>
    <xsd:attribute name="tag" type="xsd:NCName" use="required"/>
</xsd:complexType>

<xsd:complexType name="textWithSubType" mixed="true">
    <xsd:annotation>
        <xsd:documentation xml:lang="en">
            Type for a string with embedded Value substitutions
            and XHTML elements, and an xml:lang attribute.
            Elements of this type can also have an override
            attribute to help manage inheritance.
        </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
        <xsd:element name="sub" type="cdf:idrefType"
            minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute ref="xml:lang"/>
    <xsd:attribute name="override" type="xsd:boolean"
        use="optional" default="0"/>
</xsd:complexType>

<xsd:complexType name="idrefType">
    <xsd:annotation>
        <xsd:documentation xml:lang="en">
            Data type for elements that have no content,

```

```

        just a mandatory id reference.
    </xsd:documentation>
</xsd:annotation>
<xsd:attribute name="idref" type="xsd:NCName" use="required"/>
</xsd:complexType>

<xsd:complexType name="overrideableIdrefType">
    <xsd:annotation>
        <xsd:documentation xml:lang="en">
            Data type for elements that have no content,
            just a mandatory id reference, but also have
            an override attribute for controlling inheritance.
        </xsd:documentation>
    </xsd:annotation>
    <xsd:complexContent>
        <xsd:extension base="cdf:idrefType">
            <xsd:attribute name="override" type="xsd:boolean"
                use="optional" default="0"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<!-- ***** -->
<!-- ***** Item Element (Base Class) ***** -->
<!-- ***** -->
<xsd:element name="Item" type="cdf:itemType" >
    <xsd:annotation>
        <xsd:documentation xml:lang="en">
            Type element type imposes constraints shared by all
            Groups, Rules and Values. The itemType is abstract, so
            the element Item can never appear in a valid XCCDF document.
        </xsd:documentation>
    </xsd:annotation>
</xsd:element>

<xsd:complexType name="itemType" abstract="1">
    <xsd:annotation>
        <xsd:documentation xml:lang="en">
            This abstract item type represents the basic data shared by all
            Groups, Rules and Values
        </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
        <xsd:element ref="cdf:status" minOccurs="0" maxOccurs="1"/>
        <xsd:element name="version" type="cdf:versionType"
            minOccurs="0" maxOccurs="1"/>
        <xsd:element name="title" type="cdf:textWithSubType"
            minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="description" type="cdf:htmlTextWithSubType"
            minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="warning" type="cdf:warningType"
            minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="question" type="cdf:textType"
            minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="reference" type="cdf:referenceType"
            minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:NCName" use="required"/>

```

```

<xsd:attribute name="hidden" type="xsd:boolean"
    default="false" use="optional"/>
<xsd:attribute name="prohibitChanges" type="xsd:boolean"
    default="false" use="optional"/>
<xsd:attribute name="abstract" type="xsd:boolean"
    default="false" use="optional"/>
<xsd:attribute name="extends" type="xsd:NCName"
    default="false" use="optional"/>
<xsd:attribute name="cluster-id" type="xsd:NCName"
    use="optional"/>
<xsd:attribute ref="xml:lang"/>
<xsd:attribute ref="xml:base"/>
<xsd:attribute name="Id" type="xsd:ID" use="optional"/>
</xsd:complexType>

<!-- ***** -->
<!-- ***** Selectable Item Type (Base Class) ***** -->
<!-- ***** -->
<xsd:complexType name="selectableItemType" abstract="true">
    <xsd:annotation>
        <xsd:documentation xml:lang="en">
            This abstract item type represents the basic data shared by all
            Groups and Rules. It extends the itemType given above.
        </xsd:documentation>
    </xsd:annotation>
    <xsd:complexContent>
        <xsd:extension base="cdf:itemType">
            <xsd:sequence>
                <xsd:element name="rationale"
                    type="cdf:htmlTextWithSubType"
                    minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element name="platform"
                    type="cdf:overrideableIdrefType"
                    minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element name="requires" type="cdf:idrefType"
                    minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element name="conflicts" type="cdf:idrefType"
                    minOccurs="0" maxOccurs="unbounded"/>
            </xsd:sequence>
            <xsd:attribute name="selected" type="xsd:boolean"
                default="true" use="optional"/>
            <xsd:attribute name="weight" type="cdf:weightType"
                default="1.0" use="optional"/>
            <xsd:attribute name="severity" type="cdf:severityEnumType"
                default="unknown" use="optional"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<!-- ***** -->
<!-- ***** Group Element ***** -->
<!-- ***** -->
<xsd:element name="Group" type="cdf:groupType"/>

<xsd:complexType name="groupType">
    <xsd:annotation>
        <xsd:documentation xml:lang="en">
            Data type for the Group element that represents a grouping of

```

```

        Groups, Rules and Values.
    </xsd:documentation>
</xsd:annotation>
<xsd:complexContent>
    <xsd:extension base="cdf:selectableItemType">
        <xsd:sequence>
            <xsd:element ref="cdf:Value"
                minOccurs="0" maxOccurs="unbounded"/>
            <xsd:choice minOccurs="0" maxOccurs="unbounded">
                <xsd:element ref="cdf:Group"/>
                <xsd:element ref="cdf:Rule"/>
            </xsd:choice>
            <xsd:element name="signature" type="cdf:signatureType"
                minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
    </xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<!-- ***** -->
<!-- ***** Rule Element ***** -->
<!-- ***** -->
<xsd:element name="Rule" type="cdf:ruleType">
    <xsd:unique name="ruleCheckSelectorKey">
        <xsd:selector xpath="./cdf:check"/>
        <xsd:field xpath="@selector"/>
        <xsd:field xpath="@system"/>
    </xsd:unique>
</xsd:element>
<xsd:complexType name="ruleType">
    <xsd:annotation>
        <xsd:documentation xml:lang="en">
            Data type for the Rule element that represents a
            specific benchmark test.
        </xsd:documentation>
    </xsd:annotation>
    <xsd:complexContent>
        <xsd:extension base="cdf:selectableItemType">
            <xsd:sequence>
                <xsd:element name="ident" type="cdf:identType"
                    minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element name="profile-note" minOccurs="0"
                    type="cdf:profileNoteType"
                    maxOccurs="unbounded"/>
                <xsd:element name="fixtext" type="cdf:fixTextType"
                    minOccurs="0" maxOccurs="unbounded"/>
                <xsd:element name="fix" type="cdf:fixType"
                    minOccurs="0" maxOccurs="unbounded"/>
                <xsd:choice>
                    <xsd:element name="check" type="cdf:checkType"
                        minOccurs="0" maxOccurs="unbounded"/>
                    <xsd:element name="complex-check" minOccurs="0"
                        type="cdf:complexCheckType" maxOccurs="1"/>
                </xsd:choice>
                <xsd:element name="signature" type="cdf:signatureType"
                    minOccurs="0" maxOccurs="1"/>
            </xsd:sequence>
            <xsd:attribute name="role" type="cdf:roleEnumType"

```



```

        use="optional" default="full"/>
        <xsd:attribute name="multiple" type="xsd:boolean"
            use="optional"/>
    </xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<!-- ***** -->
<!-- ***** Rule-related Types ***** -->
<!-- ***** -->
<xsd:complexType name="identType">
    <xsd:annotation>
        <xsd:documentation xml:lang="en">
            Type for a long-term globally meaningful identifier,
            consisting of a string (ID) and a URI of the naming
            scheme within which the name is meaningful.
        </xsd:documentation>
    </xsd:annotation>
    <xsd:simpleContent>
        <xsd:extension base="xsd:string">
            <xsd:attribute name="system" type="xsd:anyURI" use="required"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="warningType" mixed="true">
    <xsd:annotation>
        <xsd:documentation xml:lang="en">
            Data type for the warning element under the Rule
            object, a rich text string with substitutions
            allowed, plus an attribute for the kind of
            warning.
        </xsd:documentation>
    </xsd:annotation>
    <xsd:complexContent>
        <xsd:extension base="cdf:htmlTextWithSubType">
            <xsd:attribute name="category"
                type="cdf:warningCategoryEnumType"
                use="optional" default="general"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:simpleType name="warningCategoryEnumType">
    <xsd:annotation>
        <xsd:documentation xml:lang="en">
            Allowed warning category keywords for the warning
            element. The allowed categories are:
            general=broad or general-purpose warning (default
                for compatibility for XCCDF 1.0)
            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
        </xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="general"/>
        <xsd:enumeration value="functionality"/>
        <xsd:enumeration value="performance"/>
        <xsd:enumeration value="hardware"/>
        <xsd:enumeration value="legal"/>
        <xsd:enumeration value="regulatory"/>
    </xsd:restriction>
</xsd:simpleType>

```

```

        or compliance implications
        management=warning about impacts to the mgmt
        or administration of the target system
        audit=warning about impacts to audit or logging
        dependency=warning about dependencies between
        this Rule and other parts of the target
        system, or version dependencies.
    </xsd:documentation>
</xsd:annotation>
<xsd:restriction base="xsd:string">
    <xsd:enumeration value="general"/>
    <xsd:enumeration value="functionality"/>
    <xsd:enumeration value="performance"/>
    <xsd:enumeration value="hardware"/>
    <xsd:enumeration value="legal"/>
    <xsd:enumeration value="regulatory"/>
    <xsd:enumeration value="management"/>
    <xsd:enumeration value="audit"/>
    <xsd:enumeration value="dependency"/>
</xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="fixTextType" mixed="true">
    <xsd:annotation>
        <xsd:documentation xml:lang="en">
            Data type for the fixText element that represents
            a rich text string, with substitutions allowed, and
            a series of attributes that qualify the fix.
        </xsd:documentation>
    </xsd:annotation>
    <xsd:complexContent>
        <xsd:extension base="cdf:htmlTextWithSubType">
            <xsd:attribute name="fixref" type="xsd:NCName"
                use="optional"/>
            <xsd:attribute name="reboot" type="xsd:boolean"
                use="optional" default="0"/>
            <xsd:attribute name="strategy" type="cdf:fixStrategyEnumType"
                use="optional" default="unknown"/>
            <xsd:attribute name="disruption" type="cdf:ratingEnumType"
                use="optional" default="unknown"/>
            <xsd:attribute name="complexity" type="cdf:ratingEnumType"
                use="optional" default="unknown"/>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="fixType" mixed="true">
    <xsd:annotation>
        <xsd:documentation xml:lang="en">
            Type for a string with embedded Value and instance
            substitutions and an optional platform id ref attribute, but
            no embedded XHTML markup.
            The platform attribute should refer to a platform-definition
            element in the platform-definitions child of the Benchmark.
        </xsd:documentation>
    </xsd:annotation>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="sub" type="cdf:idrefType"/>
    </xsd:choice>
</xsd:complexType>

```

```

        <xsd:element name="instance" type="cdf:instanceFixType"/>
    </xsd:choice>
    <xsd:attribute name="id" type="xsd:NCName" use="optional"/>
    <xsd:attribute name="reboot" type="xsd:boolean"
        use="optional" default="0"/>
    <xsd:attribute name="strategy" type="cdf:fixStrategyEnumType"
        use="optional" default="unknown"/>
    <xsd:attribute name="disruption" type="cdf:ratingEnumType"
        use="optional" default="unknown"/>
    <xsd:attribute name="complexity" type="cdf:ratingEnumType"
        use="optional" default="unknown"/>
    <xsd:attribute name="system" type="xsd:anyURI" use="optional"/>
    <xsd:attribute name="platform" type="xsd:NCName" use="optional"/>
</xsd:complexType>

<xsd:simpleType name="fixStrategyEnumType">
    <xsd:annotation>
        <xsd:documentation xml:lang="en">
            Allowed strategy keyword values for a Rule fix or
            fixtext. The allowed values are:
                unknown= strategy not defined (default for forward
                    compatibility for XCCDF 1.0)
                configure=adjust target config or settings
                patch=apply a patch, hotfix, or update
                policy=remediation by changing policies/procedures
                disable=turn off or deinstall something
                enable=turn on or install something
                restrict=adjust permissions or ACLs
                policy=out-of-band changes to policy/procedures
                update=install upgrade or update the system
                combination=combo of two or more of the above
        </xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="unknown"/>
        <xsd:enumeration value="configure"/>
        <xsd:enumeration value="combination"/>
        <xsd:enumeration value="disable"/>
        <xsd:enumeration value="enable"/>
        <xsd:enumeration value="patch"/>
        <xsd:enumeration value="policy"/>
        <xsd:enumeration value="restrict"/>
        <xsd:enumeration value="update"/>
    </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ratingEnumType">
    <xsd:annotation>
        <xsd:documentation xml:lang="en">
            Allowed rating values values for a Rule fix
            or fixtext: disruption, complexity, and maybe overhead.
            The possible values are:
                unknown= rating unknown or
                    impossible to estimate (default for
                        forward compatibility for XCCDF 1.0)
                low = little or no potential for disruption,
                    very modest complexity
                medium= some chance of minor disruption,
                    substantial complexity
        </xsd:documentation>
    </xsd:annotation>

```

```

        high = likely to cause serious disruption,
            extremely complex
    </xsd:documentation>
</xsd:annotation>
<xsd:restriction base="xsd:string">
    <xsd:enumeration value="unknown"/>
    <xsd:enumeration value="low"/>
    <xsd:enumeration value="medium"/>
    <xsd:enumeration value="high"/>
</xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="instanceFixType">
    <xsd:annotation>
        <xsd:documentation xml:lang="en">
            Type for an instance element in a fix element.
            The instance element inside a fix element
            designates a spot where the name of the instance
            should be substituted into the fix template to
            generate the final fix data. The instance element
            in this usage has one optional attribute: context.
        </xsd:documentation>
    </xsd:annotation>
    <xsd:attribute name="context" type="xsd:string"
        default="undefined" use="optional"/>
</xsd:complexType>

<xsd:complexType name="complexCheckType">
    <xsd:annotation>
        <xsd:documentation xml:lang="en">
            The type for an element that can contains a boolean
            expression based on checks. This element can have
            only complex-check and check elements as children.
            It has two attributes: operator and negate. The
            operator attribute can have values "OR" or "AND",
            and the negate attribute is boolean. See the
            specification document for truth tables for the
            operators and negations. Note: complex-check is
            defined in this way for conceptual equivalence with OVAL.
        </xsd:documentation>
    </xsd:annotation>
    <xsd:choice minOccurs="1" maxOccurs="unbounded">
        <xsd:element name="check" type="cdf:checkType"/>
        <xsd:element name="complex-check" type="cdf:complexCheckType"/>
    </xsd:choice>
    <xsd:attribute name="operator"
        type="cdf:ccOperatorEnumType" use="required"/>
    <xsd:attribute name="negate" default="0"
        type="xsd:boolean" use="optional"/>
</xsd:complexType>

<xsd:simpleType name="ccOperatorEnumType">
    <xsd:annotation>
        <xsd:documentation xml:lang="en">
            The type for the allowed operator names for the
            complex-check operator attribute. For now,
            we just allow boolean AND and OR as operators.
            (The complex-check has a separate mechanism for
            inversion.)
        </xsd:documentation>
    </xsd:annotation>

```

```

        </xsd:documentation>
      </xsd:annotation>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="OR"/>
        <xsd:enumeration value="AND"/>
      </xsd:restriction>
    </xsd:simpleType>

    <xsd:complexType name="checkType">
      <xsd:annotation>
        <xsd:documentation xml:lang="en">
          Data type for the check element, a checking system
          specification URI, and XML content.
        </xsd:documentation>
      </xsd:annotation>
      <xsd:sequence>
        <xsd:element name="check-export" type="cdf:checkExportType"
          minOccurs="0" maxOccurs="unbounded"/>
        <xsd:choice minOccurs="1" maxOccurs="1">
          <xsd:element name="check-content"
            type="cdf:checkContentType"/>
          <xsd:element name="check-content-ref"
            type="cdf:checkContentRefType"/>
        </xsd:choice>
      </xsd:sequence>
      <xsd:attribute name="system" type="xsd:anyURI" use="required"/>
      <xsd:attribute name="selector" default=""
        type="xsd:string" use="optional"/>
      <xsd:attribute ref="xml:base"/>
    </xsd:complexType>

    <xsd:complexType name="checkExportType">
      <xsd:annotation>
        <xsd:documentation xml:lang="en">
          Data type for the check-export element, which
          specifies a mapping between an XCCDF internal Value
          id and a value name to be used by the checking
          system or processor.
        </xsd:documentation>
      </xsd:annotation>
      <xsd:attribute name="value-id" type="xsd:NCName" use="required"/>
      <xsd:attribute name="export-name" type="xsd:string" use="required"/>
    </xsd:complexType>

    <xsd:complexType name="checkContentRefType">
      <xsd:annotation>
        <xsd:documentation xml:lang="en">
          Data type for the check-content-ref element, which
          points to the code for a detached check in another file.
          This element has no body, just a couple of attributes:
          href and name. The name is optional, if it does not appear
          then this reference is to the entire other document.
        </xsd:documentation>
      </xsd:annotation>
      <xsd:attribute name="href" type="xsd:anyURI" use="required"/>
      <xsd:attribute name="name" type="xsd:string"/>
    </xsd:complexType>

```

```

<xsd:complexType name="checkContentType" mixed="true">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      Data type for the check-content element, which holds
      the actual code of an enveloped check in some other
      (non-XCCDF) language. This element can hold almost
      anything; XCCDF tools do not process its content directly.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:choice minOccurs="0" maxOccurs="unbounded">
    <xsd:any namespace="##other" processContents="skip"/>
  </xsd:choice>
</xsd:complexType>

<xsd:simpleType name="weightType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      Data type for a Rule's weight, a non-negative real number.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:restriction base="xsd:decimal">
    <xsd:minExclusive value="0.0"/>
    <xsd:totalDigits value="3"/>
  </xsd:restriction>
</xsd:simpleType>

<!-- ***** -->
<!-- ***** Value Element ***** -->
<!-- ***** -->
<xsd:element name="Value" type="cdf:valueType">
  <xsd:unique name="valueSelectorKey">
    <xsd:selector xpath="./cdf:value"/>
    <xsd:field xpath="@selector"/>
  </xsd:unique>
  <xsd:unique name="defaultSelectorKey">
    <xsd:selector xpath="./cdf:default"/>
    <xsd:field xpath="@selector"/>
  </xsd:unique>
</xsd:element>

<xsd:complexType name="valueType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      Data type for the Value element that represents
      a tailorable string, numeric, or boolean value in
      the Benchmark.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="cdf:itemType">
      <xsd:sequence>
        <xsd:element name="value" type="cdf:selStringType"
          minOccurs="1" maxOccurs="unbounded"/>
        <xsd:element name="default" type="cdf:selStringType"
          minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="match" type="cdf:selStringType"
          minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="lower-bound" type="cdf:selNumType"
          minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

```

```

        <xsd:element name="upper-bound" type="cdf:selNumType"
            minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="choices" type="cdf:selChoicesType"
            minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="source" type="cdf:uriRefType"
            minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="signature" type="cdf:signatureType"
            minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="type" type="cdf:valueTypeType"
        default="string" use="optional"/>
    <xsd:attribute name="operator" type="cdf:valueOperatorType"
        default="equals" use="optional"/>
    <xsd:attribute name="interactive" type="xsd:boolean"
        default="0" use="optional"/>
    <xsd:attribute name="interfaceHint" use="optional"
        type="cdf:interfaceHintType"/>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<!-- ***** -->
<!-- ***** Value-related Types ***** -->
<!-- ***** -->
<xsd:complexType name="selChoicesType">
    <xsd:annotation>
        <xsd:documentation xml:lang="en">
            The choice element specifies a list of legal or
            suggested choices for a Value object. It holds
            one or more choice elements, a mustMatch attribute,n
            and a selector attribute.
        </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
        <xsd:element name="choice" type="xsd:string"
            minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="mustMatch" type="xsd:boolean" use="optional"/>
    <xsd:attribute name="selector" default=""
        type="xsd:string" use="optional"/>
</xsd:complexType>

<xsd:complexType name="selStringType">
    <xsd:annotation>
        <xsd:documentation xml:lang="en">
            This type is for an element that has string content
            and a selector attribute. It is used for some of
            the child elements of Value.
        </xsd:documentation>
    </xsd:annotation>
    <xsd:simpleContent>
        <xsd:extension base="xsd:string">
            <xsd:attribute name="selector" default=""
                type="xsd:string" use="optional"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>

```

```

<xsd:complexType name="selNumType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      This type is for an element that has numeric content
      and a selector attribute. It is used for two of
      the child elements of Value.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:simpleContent>
    <xsd:extension base="xsd:decimal">
      <xsd:attribute name="selector" default=""
        type="xsd:string" use="optional"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="uriRefType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      Data type for elements that have no content,
      just a mandatory URI.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:attribute name="uri" type="xsd:anyURI" use="required"/>
</xsd:complexType>

<xsd:simpleType name="valueTypeType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      Allowed data types for Values, just string, numeric,
      and true/false.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="number"/>
    <xsd:enumeration value="string"/>
    <xsd:enumeration value="boolean"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="valueOperatorType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      Allowed operators for Values. Note that most of
      these are valid only for numeric data, but the
      schema doesn't enforce that.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="equals" />
    <xsd:enumeration value="not equal" />
    <xsd:enumeration value="greater than" />
    <xsd:enumeration value="less than" />
    <xsd:enumeration value="greater than or equal" />
    <xsd:enumeration value="less than or equal" />
    <xsd:enumeration value="pattern match" />
  </xsd:restriction>
</xsd:simpleType>

```



```

<xsd:simpleType name="interfaceHintType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      Allowed interface hint values. When an
      interfaceHint appears on the Value, it
      provides a suggestion to a tailoring or
      benchmarking tool about how to present
      the UI for adjusting a Value.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="choice"/>
    <xsd:enumeration value="textline"/>
    <xsd:enumeration value="text"/>
    <xsd:enumeration value="date"/>
    <xsd:enumeration value="datetime"/>
  </xsd:restriction>
</xsd:simpleType>

<!-- ***** -->
<!-- ***** Profile Element ***** -->
<!-- ***** -->
<xsd:element name="Profile" type="cdf:profileType">
  <!-- selector key constraints -->
  <xsd:unique name="itemSelectKey">
    <xsd:selector xpath="./cdf:select"/>
    <xsd:field xpath="@idref"/>
  </xsd:unique>
  <xsd:unique name="refineRuleKey">
    <xsd:selector xpath="./cdf:refine-rule"/>
    <xsd:field xpath="@idref"/>
  </xsd:unique>
  <xsd:unique name="refineValueKey">
    <xsd:selector xpath="./cdf:refine-value"/>
    <xsd:field xpath="@idref"/>
  </xsd:unique>
  <xsd:unique name="setValueKey">
    <xsd:selector xpath="./cdf:set-value"/>
    <xsd:field xpath="@idref"/>
  </xsd:unique>
</xsd:element>

<xsd:complexType name="profileType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      Data type for the Profile element, which holds a
      specific tailoring of the Benchmark.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element ref="cdf:status" minOccurs="0" maxOccurs="1"/>
    <xsd:element name="version" type="cdf:versionType"
      minOccurs="0" maxOccurs="1"/>
    <xsd:element name="title" type="cdf:textWithSubType"
      minOccurs="1" maxOccurs="unbounded"/>
    <xsd:element name="description" type="cdf:htmlTextWithSubType"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="reference" type="cdf:referenceType"
      minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

```

```

<xsd:element name="platform" type="cdf:idrefType"
  minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="select" type="cdf:profileSelectType"
  minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="set-value" type="cdf:profileSetValueType"
  minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="refine-value"
  type="cdf:profileRefineValueType"
  minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="refine-rule"
  type="cdf:profileRefineRuleType"
  minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="signature" type="cdf:signatureType"
  minOccurs="0" maxOccurs="1"/>
</xsd:sequence>
<xsd:attribute name="id" type="xsd:NCName" use="required"/>
<xsd:attribute name="prohibitChanges" type="xsd:boolean"
  default="false" use="optional"/>
<xsd:attribute name="abstract" type="xsd:boolean"
  default="false" use="optional"/>
<xsd:attribute name="note-tag" type="xsd:NCName"
  use="optional"/>
<xsd:attribute name="extends" type="xsd:NCName" use="optional"/>
<xsd:attribute ref="xml:base"/>
<xsd:attribute name="Id" type="xsd:ID" use="optional"/>
</xsd:complexType>

<!-- ***** -->
<!-- ***** Profile-related Types ***** -->
<!-- ***** -->
<xsd:complexType name="profileSelectType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      Type for the select element in a Profile; all it has
      are two attributes, no content. The two attributes
      are 'idref' which refers to a Group or Rule, and
      'selected' which is boolean.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:attribute name="idref" type="xsd:NCName" use="required"/>
  <xsd:attribute name="selected" type="xsd:boolean" use="required"/>
  <xsd:attribute name="role" type="cdf:roleEnumType" use="optional"/>
</xsd:complexType>

<xsd:complexType name="profileSetValueType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      Type for the set-value element in a Profile; it
      has one attribute and string content. The
      attribute is 'idref' which refers to a Value.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="idref" type="xsd:NCName"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

```

```

<xsd:complexType name="profileRefineValueType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      Type for the refine-value element in a Profile; all
      it has are two attributes, no content. The two
      attributes are 'idref' which refers to a Value
      and 'selector' which designates certain element
      children of the Value.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:attribute name="idref"
    type="xsd:NCName" use="required"/>
  <xsd:attribute name="selector"
    type="xsd:string" use="required"/>
</xsd:complexType>

<xsd:complexType name="profileRefineRuleType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      Type for the refine-rule element in a Profile; all
      it has are four attributes, no content. The main
      attribute is 'idref' which refers to a Rule, and
      three attributes that allow the Profile author to
      adjust aspects of how a Rule is processed during
      a benchmark run: weight, severity, and role.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:attribute name="idref" type="xsd:NCName" use="required"/>
  <xsd:attribute name="weight" type="cdf:weightType" use="optional"/>
  <xsd:attribute name="severity"
    type="cdf:severityEnumType" use="optional"/>
  <xsd:attribute name="role"
    type="cdf:roleEnumType" use="optional"/>
</xsd:complexType>

<!-- ***** -->
<!-- ***** TestResult Element ***** -->
<!-- ***** -->
<xsd:element name="TestResult" type="cdf:testResultType"/>

<xsd:complexType name="testResultType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      Data type for the TestResult element, which holds
      the results of one application of the Benchmark.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element name="benchmark" minOccurs="0" maxOccurs="1">
      <xsd:complexType>
        <xsd:attribute name="href" type="xsd:anyURI" use="required"/>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="title" type="cdf:textType"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="remark" type="cdf:textType"
      minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="profile" type="cdf:idrefType"
      minOccurs="0" maxOccurs="1"/>
  </xsd:sequence>
</xsd:complexType>

```

```

<xsd:element name="target" type="xsd:string"
  minOccurs="1" maxOccurs="1"/>
<xsd:element name="target-address" type="xsd:string"
  minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="target-facts" type="cdf:targetFactsType"
  minOccurs="0" maxOccurs="1"/>
<xsd:element name="platform" type="cdf:idrefType"
  minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="set-value" type="cdf:profileSetValueType"
  minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="rule-result" type="cdf:ruleResultType"
  minOccurs="0" maxOccurs="unbounded">
  <!-- Each context name in an instance must be unique. -->
  <xsd:key name="instanceContextKey">
    <xsd:selector xpath="cdf:instance"/>
    <xsd:field xpath="@context"/>
  </xsd:key>
  <!-- parentContext must refer to valid sibling context -->
  <xsd:keyref name="parentKeyRef" refer="cdf:instanceContextKey">
    <xsd:selector xpath="./cdf:instance"/>
    <xsd:field xpath="@parentContext"/>
  </xsd:keyref>
</xsd:element>
<xsd:element name="score" type="cdf:scoreType"
  minOccurs="1" maxOccurs="unbounded"/>
<xsd:element name="signature" type="cdf:signatureType"
  minOccurs="0" maxOccurs="1"/>
</xsd:sequence>
<xsd:attribute name="id" type="xsd:NCName" use="required"/>
<xsd:attribute name="start-time" type="xsd:dateTime" use="optional"/>
<xsd:attribute name="end-time" type="xsd:dateTime" use="required"/>
<xsd:attribute name="version" type="xsd:string" use="optional"/>
<xsd:attribute name="Id" type="xsd:ID" use="optional"/>
</xsd:complexType>

<xsd:complexType name="scoreType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      Type for a score value in a TestResult, the content is a
      real number and the element can have two optional attributes.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:simpleContent>
    <xsd:extension base="xsd:decimal">
      <xsd:attribute name="system" type="xsd:anyURI"
        use="optional"/>
      <xsd:attribute name="maximum" type="xsd:decimal"
        use="optional"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="targetFactsType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      This element holds a list of facts about the target
      system or platform. Each fact is an element of
      type factType. Each fact must have a name, but
      duplicate names are allowed. (For example, if you

```

```

        had a fact about MAC addresses, and the target
        system had three NICs, then you'd need three
        instance of the "urn:xccdf:fact:ethernet:MAC" fact.)
    </xsd:documentation>
</xsd:annotation>
<xsd:sequence>
    <xsd:element name="fact" type="cdf:factType"
        minOccurs="0" maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>

<xsd:complexType name="factType">
    <xsd:annotation>
        <xsd:documentation xml:lang="en">
            Element type for a fact about a target system: a
            name-value pair with a type. The content of the element
            is the value, the type attribute gives the type. This
            is an area where XML schema is weak: we can't make the
            schema validator check that the content matches the type.
        </xsd:documentation>
    </xsd:annotation>
    <xsd:simpleContent>
        <xsd:extension base="xsd:string">
            <xsd:attribute name="name" type="xsd:anyURI"
                use="required"/>
            <xsd:attribute name="type" type="cdf:valueTypeType"
                default="boolean" use="optional"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="ruleResultType">
    <xsd:annotation>
        <xsd:documentation xml:lang="en">
            This element holds all the information about the
            application of one rule to a target. It may only
            appear as part of a TestResult object.
        </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
        <xsd:element name="result" type="cdf:resultEnumType"
            minOccurs="1" maxOccurs="1"/>
        <xsd:element name="override" type="cdf:overrideType"
            minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="ident" type="cdf:identType"
            minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="message" type="cdf:messageType"
            minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="instance" type="cdf:instanceResultType"
            minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="fix" type="cdf:fixType"
            minOccurs="0" maxOccurs="unbounded"/>
        <!-- will we need a new restricted form for this? -->
        <xsd:element name="check" type="cdf:checkType"
            minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="idref" type="xsd:NCName" use="required"/>
    <xsd:attribute name="role" type="cdf:roleEnumType"
        use="optional"/>

```

```

    <xsd:attribute name="severity" type="cdf:severityEnumType"
        use="optional"/>
    <xsd:attribute name="time" type="xsd:dateTime" use="optional"/>
    <xsd:attribute name="version" type="xsd:string" use="optional"/>
</xsd:complexType>

<xsd:complexType name="instanceResultType">
    <xsd:annotation>
        <xsd:documentation xml:lang="en">
            Type for an instance element in a rule-result.
            The content is a string, but the element may
            also have two attribute: context and parentContext.
            This type records the details of the target system
            instance for multiply instantiated rules.
        </xsd:documentation>
    </xsd:annotation>
    <xsd:simpleContent>
        <xsd:extension base="xsd:string">
            <xsd:attribute name="context" default="undefined"
                type="xsd:string" use="optional"/>
            <xsd:attribute name="parentContext"
                type="xsd:string" use="optional"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>

<xsd:complexType name="overrideType">
    <xsd:annotation>
        <xsd:documentation xml:lang="en">
            Type for an override block in a rule-result.
            It contains five mandatory parts: time, authority,
            old-result, new-result, and remark.
        </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
        <xsd:element name="old-result" type="cdf:resultEnumType"
            minOccurs="1" maxOccurs="1"/>
        <xsd:element name="new-result" type="cdf:resultEnumType"
            minOccurs="1" maxOccurs="1"/>
        <xsd:element name="remark" type="cdf:textType"
            minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="time" type="xsd:dateTime" use="required"/>
    <xsd:attribute name="authority" type="xsd:string" use="required"/>
</xsd:complexType>

<xsd:complexType name="messageType">
    <xsd:annotation>
        <xsd:documentation xml:lang="en">
            Type for a message generated by the checking
            engine or XCCDF tool during benchmark testing.
            Content is string plus required severity attribute.
        </xsd:documentation>
    </xsd:annotation>
    <xsd:simpleContent>
        <xsd:extension base="xsd:string">
            <xsd:attribute name="severity" type="cdf:msgSevEnumType"
                use="required"/>
        </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>

```

```

    </xsd:simpleContent>
</xsd:complexType>

<xsd:simpleType name="msgSevEnumType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      Allowed values for message severity.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="error"/>
    <xsd:enumeration value="warning"/>
    <xsd:enumeration value="info"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="resultEnumType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      Allowed result indicators for a test, several possibilities:
      pass= the test passed, target complies w/ benchmark
      fail= the test failed, target does not comply
      error= an error occurred and test could not complete,
             or the test does not apply to this platform
      unknown= could not tell what happened, results
                with this status are not to be scored
      notapplicable=Rule did not apply to test target
      fixed=rule failed, but was later fixed (score as pass)
      notchecked=Rule did not cause any evaluation by
                  the checking engine (role of "unchecked")
      notselected=Rule was not selected in the Benchmark,
                  and therefore was not checked (selected="0")
      informational=Rule was evaluated by the checking
                    engine, but isn't to be scored (role of "unscored")
    </xsd:documentation>
  </xsd:annotation>
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="pass"/>
    <xsd:enumeration value="fail"/>
    <xsd:enumeration value="error"/>
    <xsd:enumeration value="unknown"/>
    <xsd:enumeration value="notapplicable"/>
    <xsd:enumeration value="notchecked"/>
    <xsd:enumeration value="notselected"/>
    <xsd:enumeration value="informational"/>
    <xsd:enumeration value="fixed"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="severityEnumType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      Allowed severity values for a Rule.
      there are several possible values:
      unknown= severity not defined (default for forward
                compatibility for XCCDF 1.0)
      info = rule is informational only, failing the
             rule does not imply failure to conform to
             the security guidance of the benchmark.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="info"/>
    <xsd:enumeration value="error"/>
    <xsd:enumeration value="warning"/>
    <xsd:enumeration value="unknown"/>
  </xsd:restriction>
</xsd:simpleType>

```

```

        (usually would also have a weight of 0)
        low = not a serious problem
        medium= fairly serious problem
        high = a grave or critical problem
    </xsd:documentation>
</xsd:annotation>
<xsd:restriction base="xsd:string">
    <xsd:enumeration value="unknown"/>
    <xsd:enumeration value="info"/>
    <xsd:enumeration value="low"/>
    <xsd:enumeration value="medium"/>
    <xsd:enumeration value="high"/>
</xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="roleEnumType">
    <xsd:annotation>
        <xsd:documentation xml:lang="en">
            Allowed checking and scoring roles for a Rule.
            There are several possible values:
                full = if the rule is selected, then check it and let
                    the result contribute to the score and appear in
                    reports. (this is the default, for
                    compatibility for XCCDF 1.0)
                unscored = check the rule, and include the results in
                    any report, but do not include the result in
                    score computations (in the default scoring model
                    the same effect can be achieved with weight=0)
                unchecked = don't check the rule, just force the
                    result status to 'unknown'. Do include
                    the rule's information in any reports.
        </xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="full"/>
        <xsd:enumeration value="unscored"/>
        <xsd:enumeration value="unchecked"/>
    </xsd:restriction>
</xsd:simpleType>
</xsd:schema>

```

Testing

The XCCDF 1.1 schema has been checked for syntax and tested with the Apache Xerces 2.6 schema-validating parser and with Altova XMLSpy 2005 release 3.

7. Appendix B – Sample Benchmark File

The sample below illustrates some of the concepts of XCCDF. It gives a few simple rules about configuration of a Cisco IOS router, based on material from the publicly available NSA Router Security Configuration Guide.

```
<?xml version="1.0" encoding="UTF-8"?>
<cdf:Benchmark id="ios-test-1" resolved="0" xml:lang="en"
  xmlns:cdf="http://checklists.nist.gov/xccdf/1.1"
  xmlns:cdfp="http://checklists.nist.gov/xccdf-p/1.0"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:htm="http://www.w3.org/1999/xhtml"
  xmlns:dsig="http://www.w3.org/2000/09/xmldsig#"
  xsi:schemaLocation="http://checklists.nist.gov/xccdf/1.1 xccdf-
1.1_27nov05.xsd http://checklists.nist.gov/xccdf-p/1.0 xccdfp-1.0.xsd">

  <cdf:status date="2005-11-27">draft</cdf:status>
  <cdf:title>XCCDF Sample for Cisco IOS</cdf:title>
  <cdf:description>
    This document defines a small set of rules for securing Cisco
    IOS routers. The set of rules constitute a <htm:i>benchmark</htm:i>.
    A benchmark usually represents an industry consensus of best
    practices. It lists steps to be taken as well as rationale for
    them. This example benchmark is merely a small subset of the
    rules that would be necessary for securing an IOS router.
  </cdf:description>

  <cdf:notice id="Sample-Terms-Of-Use" xml:lang="en">
    This document may be copied and used subject to the
    subject to the NIST terms of use
    (http://www.nist.gov/public\_affairs/disclaim.htm)
    and the NSA Legal Notices
    (http://www.nsa.gov/notices/notic00004.cfm?Address=/).
  </cdf:notice>
  <cdf:front-matter>
    <htm:p>
      This benchmark assumes that you are running IOS 11.3 or later.
    </htm:p>
  </cdf:front-matter>
  <cdf:reference href="http://www.nsa.gov/ia/">
    NSA Router Security Configuration Guide, Version 1.1b
  </cdf:reference>
  <cdf:reference>
    <dc:title>Hardening Cisco Routers</dc:title>
    <dc:creator>Thomas Akin</dc:creator>
    <dc:publisher>O'Reilly and Associates</dc:publisher>
    <dc:identifier>http://www.ora.com/</dc:identifier>
  </cdf:reference>

  <cdfp:Platform-Specification>
    <cdfp:Fact name="urn:xccdf:fact:OS"/>
    <cdfp:Fact name="urn:xccdf:fact:OS:IOS">
      <cdfp:title>Cisco IOS</cdfp:title>
    </cdfp:Fact>
```

```

<cdfp:Fact type="number" name="urn:xccdf:fact:OS:MajorVersion"/>
<cdfp:Fact type="number" name="urn:xccdf:fact:OS:MinorVersion"/>
<cdfp:Platform id="cisco-ios-12">
  <cdfp:title>Cisco IOS 12</cdfp:title>
  <cdfp:logical-test operator="AND">
    <cdfp:fact-ref name="urn:xccdf:fact:OS:IOS"/>
    <cdfp:fact-test name="urn:xccdf:fact:OS:MajorVersion"
      operator="equals">
      12
    </cdfp:fact-test>
  </cdfp:logical-test>
</cdfp:Platform>
<cdfp:Platform id="cisco-ios-11">
  <cdfp:title>Cisco IOS 12</cdfp:title>
  <cdfp:logical-test operator="AND">
    <cdfp:fact-ref name="urn:xccdf:fact:OS:IOS"/>
    <cdfp:fact-test name="urn:xccdf:fact:OS:MajorVersion"
      operator="equals">
      11
    </cdfp:fact-test>
  </cdfp:logical-test>
</cdfp:Platform>
</cdfp:Platform-Specification>

<cdf:platform idref="cisco-ios-12"/>
<cdf:version>0.1.14</cdf:version>
<cdf:model system="urn:xccdf:scoring:default"/>
<cdf:model system="urn:xccdf:scoring:flat"/>
<cdf:model system="urn:bindview.com:scoring:relative">
  <cdf:param name="floor">0.0</cdf:param>
  <cdf:param name="ceiling">1000</cdf:param>
</cdf:model>
<cdf:Profile id="profile1" prohibitChanges="1" note-tag="lenient">
  <cdf:title>Sample Profile No. 1</cdf:title>
  <cdf:set-value idref="exec-timeout-time">30</cdf:set-value>
  <cdf:refine-value idref="buffered-logging-level"
    selector="lenient"/>
  <cdf:select idref="mgmt-plane" selected="0"/>
  <cdf:select idref="ctrl-plane" selected="1"/>
  <cdf:select idref="finger" selected="1"/>
</cdf:Profile>
<cdf:Profile id="profile2" extends="profile1" note-tag="strict">
  <cdf:title override="1">Sample Profile No. 2</cdf:title>
  <cdf:set-value idref="exec-timeout-time">10</cdf:set-value>
  <cdf:refine-value idref="buffered-logging-level" selector="strict"/>
  <cdf:refine-rule idref="no-tcp-small-servers"
    weight="0.8" severity="medium"/>
  <cdf:select idref="mgmt-plane" selected="1"/>
  <cdf:select idref="data-plane" selected="1"/>
</cdf:Profile>

<cdf:Value id="exec-timeout-time" type="number"
  operator="less than or equal">
  <cdf:title>IOS - line exec timeout value</cdf:title>
  <cdf:description>
    The length of time, in minutes, that an interactive session

```

```

    should be allowed to stay idle before being terminated.
  </cdf:description>
  <cdf:question>Session exec timeout time (in minutes)</cdf:question>
  <cdf:value>10</cdf:value>
  <cdf:default>15</cdf:default>
  <cdf:lower-bound>1</cdf:lower-bound>
  <cdf:upper-bound>60</cdf:upper-bound>
</cdf:Value>

<cdf:Group id="mgmt-plane" selected="1" prohibitChanges="1" weight="3">
  <cdf:title>Management Plane Rules</cdf:title>
  <cdf:description>
    Services, settings, and data streams related to setting up
    and examining the static configuration of the router, and the
    authentication and authorization of administrators/operators.
  </cdf:description>
  <cdf:requires idref="no-directed-broadcast"/>
  <cdf:Rule id="no-finger-service-base" selected="0" weight="5.0"
    prohibitChanges="1" hidden="1" abstract="1"
    cluster-id="finger">
    <cdf:title>IOS - no IP finger service</cdf:title>
    <cdf:description>
      Disable the finger service, it can reveal information
      about logged in users to unauthorized parties.
    </cdf:description>
    <cdf:question>Prohibit the finger service</cdf:question>
    <cdf:fixtext fixref="no-finger" xml:lang="en">
      Turn off the finger service altogether, <htm:i>nobody</htm:i>
      uses it anyway.
    </cdf:fixtext>
    <cdf:check system="http://oval.mitre.org/XMLSchema/oval">
      <cdf:check-content-ref href="iosDefns.xml" name="OVAL1002"/>
    </cdf:check>
  </cdf:Rule>

  <cdf:Rule id="ios11-no-finger-service"
    selected="0" prohibitChanges="1"
    hidden="0" weight="5" extends="no-finger-service-base">
    <cdf:title override="1">IOS 11 - no IP finger service</cdf:title>
    <cdf:platform idref="cisco-ios-11"/>
    <cdf:fix id="no-finger" system="urn:xccdf:fix:system:commands"
      disruption="low" strategy="disable">
      no service finger
    </cdf:fix>
  </cdf:Rule>

  <cdf:Rule id="ios12-no-finger-service"
    selected="0" prohibitChanges="1"
    hidden="0" weight="5" extends="no-finger-service-base">
    <cdf:title override="1">IOS 12 - no IP finger service</cdf:title>
    <cdf:platform idref="cisco-ios-12"/>
    <cdf:fix id="no-finger" system="urn:xccdf:fix:system:commands"
      disruption="low" strategy="disable">
      no ip finger
    </cdf:fix>
  </cdf:Rule>

```

```

<cdf:Rule id="req-exec-timeout" selected="1" weight="8" multiple="1">
  <cdf:title>Require exec timeout on admin sessions</cdf:title>
  <cdf:description>
    Configure each administrative access line to terminate idle
    sessions after a fixed period of time determined by local policy
  </cdf:description>
  <cdf:question>Require admin session idle timeout</cdf:question>
  <cdf:profile-note tag="lenient">
    Half an hour
  </cdf:profile-note>
  <cdf:profile-note tag="strict">
    Ten minutes or less
  </cdf:profile-note>
  <cdf:fix strategy="configure" disruption="low"
    system="urn:xccdf:fix:commands">
    line vty 0 4
    exec-timeout <cdf:sub idref="exec-timeout-time"/>
  </cdf:fix>
  <cdf:check system="http://oval.mitre.org/XMLSchema/oval">
    <cdf:check-export value-id="exec-timeout-time"
      export-name="var-2"/>
    <cdf:check-content-ref href="iosDefns.xml" name="OVAL708"/>
  </cdf:check>
</cdf:Rule>
</cdf:Group>

<cdf:Group id="ctrl-plane" selected="1" prohibitChanges="1" weight="3">
  <cdf:title>Control Plane Rules</cdf:title>
  <cdf:description>
    Services, settings, and data streams that support the
    operation and dynamic status of the router.
  </cdf:description>
  <cdf:question>Check rules related to system control</cdf:question>

  <cdf:Value id="buffered-logging-level" type="string"
    operator="equals" prohibitChanges="0"
    interfaceHint="choice">
    <cdf:title>Logging level for buffered logging</cdf:title>
    <cdf:description>
      Logging level for buffered logging; this setting is
      a severity level. Every audit message of this
      severity or more (worse) will be logged.
    </cdf:description>
    <cdf:question>Select a buffered logging level</cdf:question>
    <cdf:value selector="strict">informational</cdf:value>
    <cdf:value selector="lenient">warning</cdf:value>
    <cdf:value>notification</cdf:value>
    <cdf:choices mustMatch="1">
      <cdf:choice>warning</cdf:choice>
      <cdf:choice>notification</cdf:choice>
      <cdf:choice>informational</cdf:choice>
    </cdf:choices>
    <cdf:source uri="urn:OS:Cisco:IOS:logging:levels"/>
  </cdf:Value>

```

```

<cdf:Rule id="no-tcp-small-servers" selected="1"
    prohibitChanges="1" weight="7">
  <cdf:title>Disable tcp-small-servers</cdf:title>
  <cdf:description>
    Disable unnecessary services such as echo, chargen, etc.
  </cdf:description>
  <cdf:question>Prohibit TCP small services</cdf:question>
  <cdf:fixtext>
    Disable TCP small servers in IOS global config mode.
  </cdf:fixtext>
  <cdf:fix>no service tcp-small-servers</cdf:fix>
  <cdf:check system="http://oval.mitre.org/XMLSchema/oval">
    <cdf:check-content-ref href="iosDefns.xml" name="OVAL1000"/>
  </cdf:check>
</cdf:Rule>

<cdf:Rule id="no-udp-small-servers" selected="1" role="full"
    prohibitChanges="1" weight="5.7">
  <cdf:title>Disable udp-small-servers</cdf:title>
  <cdf:description>
    Disable unnecessary UDP services such as echo, chargen, etc.
  </cdf:description>
  <cdf:question>Forbid UDP small services</cdf:question>
  <cdf:fixtext>
    Disable UDP small servers in IOS global config mode.
  </cdf:fixtext>
  <cdf:fix>no service udp-small-servers</cdf:fix>
  <cdf:check system="http://oval.mitre.org/XMLSchema/oval">
    <cdf:check-content-ref href="iosDefns.xml" name="OVAL1001"/>
  </cdf:check>
</cdf:Rule>

<cdf:Rule id="enabled-buffered-logging-at-level" selected="1"
    prohibitChanges="0" weight="8.5">
  <cdf:title xml:lang="en">
    Ensure buffered logging enabled at proper level
  </cdf:title>
  <cdf:description>
    Make sure that buffered logging is enabled, and that
    the buffered logging level to one of the appropriate
    levels, Warning or higher.
  </cdf:description>
  <cdf:question>Check buffered logging and level</cdf:question>
  <cdf:fix>
    logging on
    logging buffered <cdf:sub idref="buffered-logging-level"/>
  </cdf:fix>
  <cdf:complex-check operator="AND" negate="1">
    <cdf:check system="http://oval.mitre.org/XMLSchema/oval">
      <cdf:check-export value-id="buffered-logging-level"
        export-name="var-4"/>
      <cdf:check-content-ref href="iosDefns.xml"
        name="org.cisecurity.cisco.ios.logging.buf.level"/>
    </cdf:check>
    <cdf:check system="http://oval.mitre.org/XMLSchema/oval">
      <cdf:check-content-ref href="iosDefns.xml"

```

```

        name="org.cisecurity.cisco.ios.logging.enabled"/>
    </cdf:check>
</cdf:complex-check>
</cdf:Rule>
</cdf:Group>

<cdf:Group id="data-plane" selected="1" prohibitChanges="1" weight="2">
    <cdf:title>Data Plane Level 1</cdf:title>
    <cdf:description>
        Services and settings related to the data passing through
        the router (as opposed to directed to it). Basically, the
        data plane is for everything not in control or mgmt planes.
    </cdf:description>
    <cdf:question>Check rules related to data flow</cdf:question>

    <cdf:Group id="routing-rules" selected="1" prohibitChanges="1">
        <cdf:title>Routing Rules</cdf:title>
        <cdf:description>
            Rules in this group affect traffic forwarded through the
            router, including router actions taken on receipt of
            special data traffic.
        </cdf:description>
        <cdf:question>Apply standard forwarding protections</cdf:question>

        <cdf:Rule id="no-directed-broadcast" weight="7" multiple="1"
            selected="1" prohibitChanges="1">
            <cdf:title>IOS - no directed broadcasts</cdf:title>
            <cdf:description>
                Disable IP directed broadcast on each interface.
            </cdf:description>
            <cdf:question>Forbid IP directed broadcast</cdf:question>
            <cdf:fixtext>
                Disable IP directed broadcast on each interface
                using IOS interface configuration mode.
            </cdf:fixtext>
            <cdf:fix>
                interface <cdf:instance context="interface"/>
                no ip directed-broadcast
            </cdf:fix>
            <cdf:check system="http://oval.mitre.org/XMLSchema/oval">
                <cdf:check-content-ref href="iosDefns.xml" name="OVAL1101"/>
            </cdf:check>
        </cdf:Rule>
    </cdf:Group>
</cdf:Group>

<cdf:TestResult id="ios-test-5"
    end-time="2004-09-25T13:45:02-04:00">
    <cdf:benchmark href="ios-sample-v1.1.xccdf.xml"/>
    <cdf:title>Sample Results Block</cdf:title>
    <cdf:remark>Test run by Bob on Sept 25</cdf:remark>
    <cdf:target>lower.test.net</cdf:target>
    <cdf:target-address>192.168.248.1</cdf:target-address>
    <cdf:target-address>2001:8::1</cdf:target-address>
    <cdf:target-facts>
        <cdf:fact type="string" name="urn:xccdf:fact:ethernet:MAC">

```

```

        02:50:e6:c0:14:39
    </cdf:fact>
    <cdf:fact name="urn:xccdf:fact:OS:IOS">1</cdf:fact>
    <cdf:fact type="number"
        name="urn:xccdf:fact:OS:IOS:major-version">
        12
    </cdf:fact>
    <cdf:fact type="number"
        name="urn:xccdf:fact:OS:IOS:minor-version">
        3
    </cdf:fact>
    <cdf:fact type="string" name="urn:xccdf:fact:OS:IOS:release">
        12.3(14)T
    </cdf:fact>
</cdf:target-facts>
<cdf:set-value idref="exec-timeout-time">10</cdf:set-value>
<cdf:rule-result idref="iosl2-no-finger-service"
    time="2004-09-25T13:45:00-04:00">
    <cdf:result>pass</cdf:result>
</cdf:rule-result>
<cdf:rule-result idref="req-exec-timeout"
    time="2004-09-25T13:45:06-04:00">
    <cdf:result>pass</cdf:result>
    <cdf:override time="2004-09-25T13:59:00-04:00"
        authority="Neal Ziring">
        <cdf:old-result>fail</cdf:old-result>
        <cdf:new-result>pass</cdf:new-result>
        <cdf:remark>Test override</cdf:remark>
    </cdf:override>
    <cdf:instance context="line">console</cdf:instance>
    <cdf:fix>
        line console
        exec-timeout 10 0
    </cdf:fix>
</cdf:rule-result>
<cdf:rule-result idref="iosl2-no-finger-service">
    <cdf:result>notselected</cdf:result>
</cdf:rule-result>
<cdf:score system="urn:xccdf:model:default">67.5</cdf:score>
<cdf:score system="urn:xccdf:model:flat"
    maximum="214">
    157.5
</cdf:score>
</cdf:TestResult>
</cdf:Benchmark>

```

8. Appendix C: Pre-defined URIs

The following URLs and URNs are defined for XCCDF 1.1.

Long-term Identification Systems

These URIs may appear as the value of the system attribute of an ident element in a Rule.

`http://cve.mitre.org/`

MITRE's Common Vulnerabilities and Exposures – the identifier value should be a CVE number or CVE candidate number.

`http://www.cert.org/`

CERT Coordination Center – the identifier value should be a CERT advisory identifier (e.g. "CA-2004-02").

`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")

`http://www.kb.cert.org/`

US-CERT vulnerability notes database – the identifier values should be a vulnerability note number (e.g. "709220").

`http://iase.disa.mil/IAalerts/`

DISA Information Assurance Alerts (IAVA) – the identifier value should be a DOD IAVA identifier.

Check Systems

These URIs may appear as the value of the system attribute of a check element (see p. 43)

`http://oval.mitre.org/XMLSchema/oval`

MITRE's Open Vulnerability Assessment Language (see [15]).

`http://www.cisecurity.org/xccdf/interactive/1.0`

Center for Internet Security interactive query check system, used for asking the user questions about the target system during application of a benchmark.

Scoring Models

These URIs may appear as the value of the system attribute on the model element or a score element (see pp. 52 and 60).

`urn:xccdf:scoring:default`

This specifies the default (XCCDF 1.0) scoring model.

`urn:xccdf:scoring:flat`

This specifies the flat, weighted scoring model.

`urn:xccdf:scoring:flat-unweighted`

This specifies the flat scoring model with weights ignored (all weights set to 1).

Remediation Systems

The URIs represent remediation sources, mechanisms, schemes, or providers. They may appear as the system attribute on a fix element (see p. 48).

urn:xccdf:fix:commands

This specifies that the content of the fix element is 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

This specifies that the content of the fix element is a list of one or more URLs. The resources identified by the URLs should be applied in order to bring the system into compliance with the Rule.

urn:xccdf:fix:script:language

A URN of this form specifies that the content of the fix element is 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 URN of this form specifies that the content of the fix element is a patch identifier, in proprietary format as defined by the vendor. The vendor string should be the common short name of the vendor (e.g. “Sun”, “Microsoft”) or the vendor’s DNS name (e.g. “hp.com”, “apache.org”).

9. References

- [1] Fallside, David C., *XML Schema Part 0: Primer*, W3C Recommendation, May 2001. (<http://www.w3.org/>)
- [2] Buttner, Andrew, “<Oval SQL='false'>”, presentation, The MITRE Corporation, October 2003.
- [3] Baker, Mark *et al*, *XHTML Basic*, W3C Recommendation, December 2000. (<http://www.w3.org/>)
- [4] Bray, Tim *et al*, *Namespaces in XML*, W3C Recommendation, January 1999. (<http://www.w3.org/>)
- [5] Jones, George, “Introduction to RAT”, presentation, Center for Internet Security, October 2003.
- [6] Calabrese, Chris, “VulnTrack”, presentation at 1st XCCDF Workshop, October 2003.
- [7] Davis, Mark “Unicode Regular Expressions”, Unicode Technical Recommendation No. 18, version 9, January 2004.
- [8] Bartel *et al*, “XML – Signature Syntax and Processing”, W3C Recommendation, February 2002. (<http://www.w3.org/>)
- [9] Marsh, J. and Orchard, D. “XML Inclusions (XInclude) Version 1.0”, W3C Candidate Recommendation, April 2004. (<http://www.w3.org/>)
- [10] Hillmann, Diane, “Using Dublin Core”, DCMI, August 2003.
- [11] “Security Configuration Checklists Program for IT Products”, NIST Special Publication 800-70, August 2004. (<http://checklists.nist.gov/>)
- [12] Waltermire, David “XCCDF Platform Specification”, Center for Internet Security, September 2004.
- [13] Johnston, P. and Powell, A. “Guidelines for implementing Dublin Core in XML”, DCMI, April 2003.
- [14] Ziring, N. and Wack, J. “Specification for the Extensible Configuration Checklist Description Format (XCCDF)”, NIST IR 7188, January 2005.
- [15] Wojcik, M., Proulx, D., Baker, J. and Roberge, R. “Introduction to OVAL”, The MITRE Corporation, July 2005.
- [16] Ziring, N. and Grance, T. “Specification for the Extensible Configuration Checklist Description Format Platform Facts Definition (XCCDF-P) version 1.1”, work in progress, February 2006.