incomplete draft

NISTIR nnnn

Specification for the Extensible Configuration Checklist Description Format Platform Facts Definition (XCCDF-P)

Neal Ziring, Author,
National Security Agency

Timothy Grance, NIST Editor

National Security Agency

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

incomplete draft

NISTIR nnnn

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
February 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 Jeffrey, Director

Abstract

This document specifies a supplementary data model and XML representation for describing IT systems. It is primarily designed to supplement documents written in the Extensible Configuration Checklist Description Format (XCCDF), but it can be used independently for any XML-based system that needs a structured fact representation. Documents or document fragments written using this format consist of one or more named *facts*, arranged in a hierarchy. Each fact consists of a name, a data type, and a value. Every fact has a parent, and may have one or more children. A *platform* is a special kind of fact, which designates a condition that the system matches some well-known or stereotyped configuration or product installation.

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.

Effective application and use of security checklists depends on matching checklists and elements of them to the systems for which they are intended. The XCCDF-P specification provides a mechanism for expressing the necessary facts about systems.

Audience

The primary audience of the XCCDF-P 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 and XCCDF-P specifications.

Table of Contents

Acknowledgements

The editor would like to acknowledge the following individuals who contributed to the initial definition of XCCDF-P and its development: Stephen Quinn, David Waltermire, Matthew Kerr, and members of the DISA FSO VMS/Gold Disk team.

1. Introduction

Conformance to security conventions and best practices is an important metric in understanding and maintaining the security of IT systems. The XCCDF specification provides a language and data model for building best practice documents or checklists, and for containing the results of testing IT systems for conformance to them.

The variety and diversity of modern IT systems is very great. Usually, general security principles apply to wide ranges of systems, but the specific tests and remediation guidance apply very narrowly. For example, the general principle of "audit all user access" applies to most general-purpose operating systems, but the specific mechanisms to check that the principle is followed is different on virtually every one of them. Applying proper tests for conformance to security principles requires accurate and precise identification of the specific platform being tested.

The XCCDF 1.0 specification did not include a platform facts specification; instead, a supplement to XCCDF was designed by the Center for Internet Security (CIS). This specification is meant to replace that one, with more granularity and expressive structure.

Therefore, it is important for a checklist, and each rule in the checklist, to be qualified by the platforms to which it applies. An XCCDF checklist author must be able to define the criteria that distinguish the platforms, name the platforms, and then tag parts of the checklist with them. The XCCDF Platform (XCCDF-P) specification provides a data model and XML representation suitable for expressing qualifications and hierarchies of facts about platforms.

2. Requirements

Expressing a checklist and the results of checking requires that we be able to identify and distinguish the IT entity being checked, and the supporting physical or logical components that support it. There are three terms we need for this discussion.

- *Asset* - a physical computer or other independent network component that can be identified by one or more names or addresses. Example: a workstation, a router, a firewall appliance. The XCCDF Platform specification does not deal with Assets directly, but only as the physical basis for targets.
- *Target* - a host, system, appliance, application, network, or service tested for compliance with a benchmark. An Asset may also be a Target. Examples: a Windows PC, an Apache web server, a Juniper router.
- *Platform* - the software and hardware provenance and environment of a Target. This may include qualifications about the Asset, version information about the target, vendor information, and additional facts. A particular target may or may not be an instance of a defined platform. The intent of identifying a platform is usually to (1) identify parts of a benchmark that should be applied, and (2) provide a categorization of targets for the purposes of comparison or metrics. Examples: Microsoft Windows 2000 Pro, Cisco IOS 12.3, Oracle 9i Database Server on Sun Solaris 8.

A benchmark or checklist is typically designed to apply to a platform or a family of platforms. It is important for the checklist author to be able to clearly define the platforms to which the benchmark and its constituent rules apply. Satisfying this requirement has two parts:

1. Declaring the conditions or facts that define platforms; this structure needs to support boolean combining operators.
2. Marking XCCDF objects, such as Rules, Groups, and Profiles, as applicable for particular platforms. Each such object needs to accept markings for zero or more platforms. Any object without a marking inherits the marking from its enclosing Item.

2.1. Basic Functional Requirements

The XCCDF-P specification satisfies part 1, and aspects of the XCCDF specification satisfy part 2. The data model and XML representation of XCCDF-P should satisfy the following additional requirements.

- Able to distinguish between different kinds or families of platform facts (e.g. operating system version, installed services, hardware type, vendor, etc.)
- Each fact must be able to be marked with a distinct, potentially unique identifier.
- Able to express hierarchical relationships among facts. For example, fact "Solaris" would be a child of fact "Unix" which would be a child of fact "OS".

- Able to express facts about several different aspects of a platform together (e.g. express the notion of "A Windows 2003 server with IIS 6.0 and MS-SQL Server").
- Able to express every platform fact that the Center for Internet Security platform specification can express, with easy and automatic translation.
- Reduce or eliminate confusion or mis-identification based on string names (e.g. are "Windows XP Professional" and "WinXP" the same OS platform? Are "Java 5.0" and "J2RE 1.5.0" the same application platform?)
- Able to base the values of facts on external conditions at the time a target is actually being tested for conformance with a checklist.
  (XCCDF satisfies this sort of requirements by depending on an external target-specific checking engine. XCCDF-P uses the same approach.)
- In some cases, the checklist tools or other system processing the platform definitions and supporting facts may be unable to determine an accurate value. The processing model for XCCDF-P must include the notion of an unknown or undetermined fact.

The key purpose of the platform information in a benchmark is to identify the target systems to which the benchmark, or parts of it, can be correctly applied. Another purpose could be to support recording of platform information in the context of benchmark testing results – this specification does not directly address recording platform information, but the naming scheme for facts is sufficiently general that the names can be re-used for that purpose in the XCCDF 1.1 specification.

2.2. Common Dictionary of Fact Identifiers

It will also be advantageous for the community to agree on a basic tree of facts about common IT platforms. A common set of facts will help support interoperability and information sharing among platform vendors, security tool developers, government, and the rest of the security community. The immediate children of the root should include the four platform categorizations in the current CIS platform-definitions schema: application, service, OS, and hardware.

Facts in the community consensus dictionary could be used in the definitions of platforms without being defined explicitly in the XCCDF-P specification.

Appendix B contains a short list of core fact identifiers.

3. Data Model

The XCCDF-P data model consists of three conceptual objects: the Platform specification, Fact definitions, and Platform definitions. A Fact definition gives the name, type, and additional information about a particular aspect that a platform might exhibit or features it might possess. A Platform definition is a named logical combination of one or more Facts. A Platform specification is simply a list of Fact definitions followed by a list of Platform definitions.

Every Fact definition has a unique identifier. Facts are implicitly organized into a tree or hierarchy by their names, just like directories and files in a filesystem – a Fact's full name is a path down through the hierarchy. Figure 1, below, shows an example hierarchy. One Fact in the diagram is labeled with its full identifier to show how the identifier URI is constructed from the path down the tree. The root Fact always exists implicitly in any XCCDF-P Platform specification, and always has the value true.

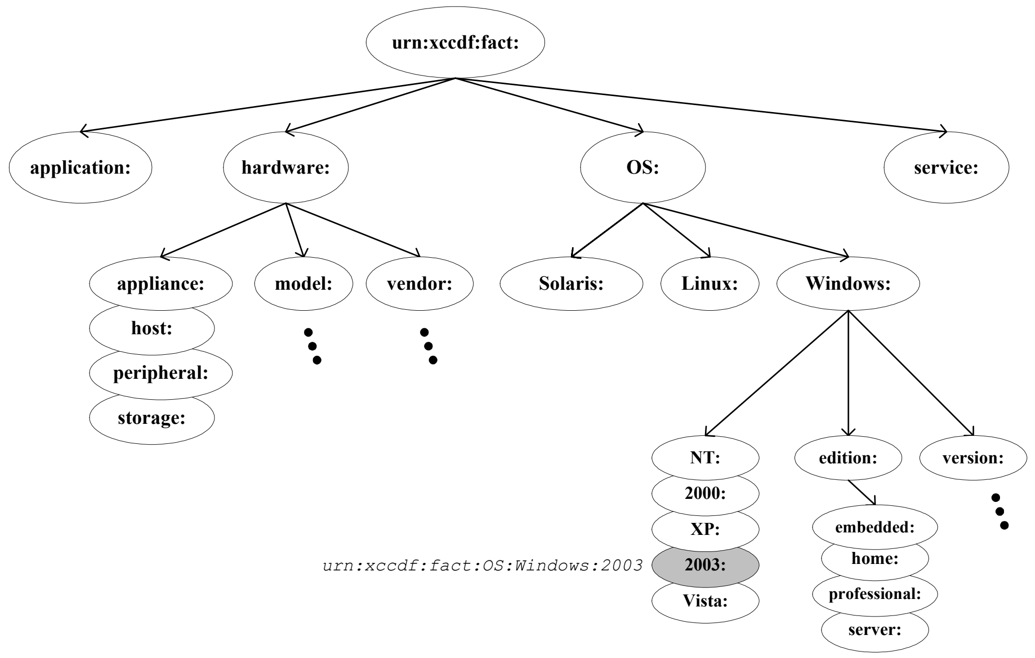

Figure 1 – Hierarchical Organization of Facts (partial sample)

Each Platform definition is a logical combination of facts. If the facts about a particular target satisfy the logical combination defined for a platform, then the target is an *instance* of that platform. A target system may be an instance of several different platforms. Platform definitions are not organized into a hierarchy, each one is independent.

3.1. The Platform Specification

A Platform specification consists of a list of Fact definitions followed by a list of Platform definitions. Platform specifications can be standalone documents, or they can be embedded within a containing document like a checklist.

The table below describes the properties that make up a Platform Specification object.

Platform Specification

Property	Type	Count	Description
facts	Fact objects	0-n	Fact definitions objects that make up the basis for this specification's Platform definitions
platforms	Platform objects	0-n	Platform definitions based on the Facts in this specification.

3.2. The Fact Definition Object

Each Fact definition has an identifier or name, some descriptive material including a title, and a fact value. In XCCDF-P 1.1, all Facts are boolean. The values of a Fact are true, false, and unknown.

The table below describes all the properties that make up the Fact definition object type.

Fact Definition

Property	Type	Count	Description
name	URI	1	an identifier which uniquely names this fact
title	text	0-n	a descriptive name for this fact (optional)
remark	text	0-n	additional remarks or description about this fact (optional)
check	*special*	0-n	specification for how to determine the value of the fact during checklist application, using a particular checking engine (optional)

The title and remarks that describe a Fact may be marked with a language locale, to support localized checklists or benchmarks.

Each Fact has a name. For simple Facts, such as whether a target platform has a particular OS installed, checklist testing tools should be able to set the value of the fact directly.

Each check property defines a means to determine the value of a Fact during application of a checklist or benchmark. Each check property consists of two parts: the checking system identifier, and the checking system content. The identifier is simply a URI that designates a particular checking system (e.g. OVAL). The content consists of an actual statement or input block for the checking system, or a reference to such a statement or input block in some other file. If the check passes, then the value of the Fact is true, but if the check fails, then the value of the Fact is false. If the check cannot be completed or returns an error, then the value of the Fact is unknown. This construction is identical to that used for defining Rule checks in XCCDF, as defined in [6].

3.3. The Platform Definition Object

Each platform definition has an identifier or name, some descriptive material including a title, and the definition content. Each piece of descriptive material may be marked with a language locale, to support localized checklists or benchmarks. The definition content is essentially a boolean expression that (1) allows specification authors to qualify applicable platforms, and (2) allows testing tools to determine whether any target system is an instance of that platform.

A definition's content is either a direct Fact reference or a compound test.

- Fact reference -
  a by-name reference to a particular Fact. If a Platform's definition content simply contains a fact reference, then the platform definition is satisfied by any target for which the Fact is true (for Facts of string or number type, the Fact is considered true if it does not have the value unknown).
- Logical test -
  a logical conjunction (AND) or disjunction (OR) of one or more Fact references, Fact tests, and logical tests. Individual logical tests can be negated (inverted). Nested logical tests allow the checklist author to express the definition of a platform as any logical combination of

The table below shows the properties of the platform definition object type.

Platform Definition

Property	Type	Count	Description
id	identifier	1	locally unique name for this platform; this identifier exists to allow references to the platform from other parts of a checklist
name	URI	0-1	global or enterprise-specific long-term name for this platform (optional)
title	text	0-n	descriptive title for the platform (optional)
remark	text	0-n	additional description or information about the platform (optional)
definition	*special*	1	the logical criteria to be used for determining if a target is an instance of this platform

If a Fact reference in a Platform definition does not correspond to any Fact listed in the XCCDF-P specification text, that is not an error. Facts may be defined implicitly by processing tools, or defined explicitly in external XCCDF-P Fact libraries, or may be community standardized or common Facts may be considered valid for use anywhere. Facts that are not explicitly defined in an XCCDF-P specification, and for which a particular processing tool does not possess a definition, must be considered to have the value unknown.

3.4. Evaluation Model

This sub-section describes the computational structure for using XCCDF-P specifications.

Values

A Fact can have one of three values: true, false, or uknown.

A Platform always has a strictly boolean value: true or false. The value true implies that the target system under evaluation satisfies the definition. There is no 'unknown' for Platform definitions; if the definition cannot be satisfied completely then it is deemed false.

Evaluation Procedure

During application of a benchmark or checklist, or any other processing that requires decisions based on Platform definitions, a tool evaluating an XCCDF-P specification must perform the follow steps.

1. Eval.Facts – assign a value to all Fact definitions.
2. Eval.Platforms - use the Fact values to assign a value to all Platform definitions.

The procedure for assigning a value to all Facts requires traversing the hierarchy; when a boolean Fact receives a value of false or unknown, that imposes the value unknown on all its descendants.

The full procedure is described below. Note that tools need not implement the algorithm in this way, but their results must match the results of this algorithm.

Sub-Step	Description
Eval.Facts.Initialize	Initialize the tree of Fact definitions: assign the value true to the root Fact, assign the value unknown to all other Facts.
Eval.Facts.Traverse.Init	Create an empty queue of Facts.
Eval.Facts.Traverse.Prepop	Add the root Fact to the queue with the value True. Add all pre-defined and common Facts to the queue with the value unknown.
Eval.Facts.Traverse.Process	Remove a Fact from the head of the queue. Process the Fact using one of the following rules: • If the Fact's parent has the value false or unknown, then assign this Fact the value unknown. • If the Fact possesses a check property written for a checking system that the processor can support, then evaluate the check property using the checking system. • If the processor can assign a true or false value based on internal checking or built-in facts, then assign the value. • Otherwise, assign the value unknown. Add all children of this Fact defined in the Platform Specification to the queue. If the queue is empty, proceed to Eval.Platforms.Initialize, otherwise go to Eval.Facts.Traverse.Process.
Eval.Platforms.Initialize	For each Platform definition, assign the value false.

Sub-Step	Description
Eval.Platforms.Process	For each Platform definition, apply the following rules: • If the definition is a fact reference, and the reference has the value true, then assign the Platform the value true. • If the definition is a logical test, then compute the value of the test; if the result is true then assign the Platform the value true.

Applying the rules in step Eval.Platforms.Process requires explicit directions for how to compute the boolean value for a Platform including Facts with values of unknown. Every fact reference, fact test, and logical test must evaluate to true, false, or unknown. Rules for evaluation are:

- Fact references -
  If the referenced Fact does not exist, then the value of the Fact reference is unknown. If the referenced Fact is boolean, then the value of the Fact reference is simply the value of the Fact. If the referenced Fact is numeric type, then the value of the Fact reference is unknown if the Fact is unknown, false if the Fact has the value 0, and true otherwise. If the referenced Fact has string type, then the value of the Fact reference is unknown if the Fact is unknown, false if the Fact has the value of “” (empty string), and true otherwise.
- Logical tests -
  Each logical test has an operator, AND or OR, and may also be negated. Every term of the test has a value of true, false, or unknown. The truth tables below given the result for each possible combination.

Normal (not negated):

AND	T	F	U
T	T	F	U
F	F	F	F
U	U	F	U

OR	T	F	U
T	T	T	T
F	T	F	U
U	T	U	U

Negated:

~AND	T	F	U
T	F	T	U
F	T	T	T
U	U	T	U

~OR	T	F	U
T	F	F	F
F	F	T	U
U	F	U	U

If a Platform definition has the value true, then the target is said to be an instance of that Platform.

4. XML Representation

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

XCCDF-P elements belong to the namespace “http://checklists.nist.gov/xccdf-p/1.1”. When an XCCDF-P specification is a standalone XML document, then all elements in the document must belong to the XCCDF-P namespace. The recommended namespace prefix is “cdfp”.

The example below gives an idea of how a simple XCCDF-P specification looks.

Example 1 – Small XCCDF-P Specification

```
<?xml version=″1.0″ ?>

<cdfp:Platform-Specification
    xmlns:cdfp="http://checklists.nist.gov/xccdf-p/1.1">

   <!-- Facts -->
   <cdfp:Fact name="urn:xccdf:fact:OS"/>
   <cdfp:Fact name="urn:xccdf:fact:OS:Windows">
        <cdfp:title>Microsoft Windows(tm) OS</cdfp:title>
   </cdfp:Fact>
   <cdfp:Fact name="urn:xccdf:fact:OS:Windows:XP">
        <cdfp:check system="http://oval.mitre.org/XMLSchema/oval">
            <cdfp:check-content-ref
                  href="winTests.xml" name="OVAL-P-032"/>
        </cdfp:check>
   </cdfp:Fact>
   <cdfp:Fact name="urn:xccdf:fact:OS:Unix"/>
   <cdfp:Fact name="urn:xccdf:fact:OS:Unix:Solaris">
        <cdfp:title>Sun Solaris(tm) OS</cdfp:title>
   </cdfp:Fact>
   <cdfp:Fact name="urn:xccdf:fact:OS:edition:trusted">
        <cdfp:title>Trusted Solaris OS</cdfp:title>
        <cdfp:check system="http://oval.mitre.org/XMLSchema/oval">
            <cdfp:check-content-ref
                    href="solTests.xml" name="OVAL-P-751"/>
        </cdfp:check>
   </cdfp:Fact>
   <cdfp:Fact type="string" name="urn:xccdf:fact:OS:vendor"/>
   <cdfp:Fact name="urn:xccdf:fact:service:network:NTDomain">
        <cdfp:check system="http://oval.mitre.org/XMLSchema/oval">
            <cdfp:check-content-ref
                    href="winTests.xml" name="OVAL-P-096"/>
        </cdfp:check>
   </cdfp:Fact>

   <!-- Platforms -->
   <cdfp:Platform id="xp-cl"
```

```
              name="urn:xccdf:platform:WinXP:client">
       <cdfp:title>Windows XP Domain Client</cdfp:title>
       <cdfp:logical-test operator="AND" negate="0">
           <cdfp:fact-ref name="urn:xccdf:fact:OS:Windows:XP"/>
           <cdfp:fact-ref
                  name="urn:xccdf:fact:service:network:NTDomain"/>
        </cdfp:logical-test>
   </cdfp:Platform>
   <cdfp:Platform id="tsol8"
              name="urn:xccdf:platform:Sun:Solaris:8:Trusted">
        <cdfp:title xml:lang="en">Trusted Solaris 8</cdfp:title>
        <cdfp:logical-test operator="AND" negate="0">
          <cdfp:fact-ref
                name="urn:xccdf:fact:OS:Unix:Solaris:Trusted"/>
          <cdfp:fact-ref name="urn:xccdf:fact:OS:MajorVersion:2"/>
          <cdfp:fact-ref name="urn:xccdf:fact:OS:MajorVersion:8"/>
        </cdfp:logical-test>
    </cdfp:Platform>
</cdfp:Platform-Specification>
```

The rest of this section describes the individual elements that may appear in an XCCDF-P specification. For full details, see the schema given in Appendix A.

<Platform-Specification>

This element acts as the document element for standalone XCCDF-P documents. When an XCCDF-P specification is embedded in other XML, this element may be omitted.

Content:	elements
Cardinality:	1
Parent Elements:	*none*
Attributes:	*none*
Child Elements:	Fact, Platform

(Note: under the current draft of the XCCDF 1.1 specification and schema [9], this element must be used as the container for platform definitions.)

<Fact>

The Fact element represents a single item of information about a target platform. It has a name which is a URI, a type, optional title and remarks, and optional content.

Content:	elements
Cardinality:	0-n
Parent Elements:	Platform-Specification
Attributes:	**name**
Child Elements:	title, remark, check

All Facts are boolean, which means that the value of a Fact may be true, false, or unknown. The value of a Fact is an aspect of processing a Platform Specification with respect to a given target system, it does not appear in the XML representation.

The title and remark elements may be localized with an xml:lang attribute.

The check element is a container for input to an external checking engine (e.g. MITRE's OVAL [7]); the engine is presumed to be capable of examining the target platform and determining the value for the Fact. Several check elements are permitted, each one marked with a URI that designates a different possible checking engine.

<Platform>

The Platform element represents the description or qualifications of a particular IT platform type. The definition of a Platform is simply a logical composition of one or more Facts.

Content:	elements
Cardinality:	0-n
Parent Elements:	Platform-Specification
Attributes:	**id**, **name**
Child Elements:	title, remark, check

<check>

The check element may appear only as a child of a Fact. It represents a mean to acquire or ascertain the value for a Fact using an external checking system or engine. The engine must be specified by a URI in the 'system' attribute.

Content:	elements
Cardinality:	0-n
Parent Elements:	Fact
Attributes:	**system**
Child Elements:	check-content, check-content-ref

Either a check-content or a check-content-ref child element must appear.

<check-content>

As the child of a check element, the check-content element holds a statement, expression, or specification for obtaining the value of a Fact. The contents must be expressed in the language of the checking engine.

Content:	*mixed*
Cardinality:	0-1
Parent Elements:	check
Attributes:	*none*
Child Elements:	*any from non-XCCDF-P namespace*

<check-content-ref>

As the child of a check element, the check-content-ref element provides a reference to an external resource for obtaining the value of a Fact. There is no content; attributes of the element provide the exact reference to the external resource.

Content:	*none*
Cardinality:	0-1
Parent Elements:	check
Attributes:	**href**, name
Child Elements:	*none*

The mandatory 'href' attribute points to the external resource (e.g. file). The 'name' attribute provides an identifier for a segment, section, subroutine, component, or element of the external resource. If the name attribute is not supplied, then the reference applies to the entire resource.

<fact-ref>

This element appears as a child of a Platform definition or a logical test, it is simply a reference to a Fact that always evaluates to a boolean result.

Content:	*none*
Cardinality:	0-n
Parent Elements:	Platform, logical-test
Attributes:	**name**
Child Elements:	*none*

If the name attribute refers to a Fact that does not exist or has the value unknown, then the reference has value unknown. If the name attribute refers to a numeric Fact, then the reference is false if the Fact has value 0 and true otherwise. If the name attribute refers to a string Fact, then the reference is false if the Fact has value "" and true otherwise.

<logical-test>

This element appears as a child of Platform definition, and may also be nested to create more complex logical tests. The content consists of one or more elements: fact-ref, fact-test, and logical-test children are permitted. The logical operator to be applied, and optional negation of the operator, are given as attributes.

Content:	elements
Cardinality:	0-n
Parent Elements:	Platform, logical-test
Attributes:	**operator**, negate
Child Elements:	fact-ref, fact-test, logical-test

The following operators are permitted: "AND", "OR".

<remark>

This element may appear inside a Fact or Platform, and simply provides some additional description. Zero or more remark elements may appear inside a Fact or Platform element. To support development of benchmarks and checklists for multiple languages, this element supports the ‘xml:lang’ attribute.

Content:	string
Cardinality:	0-n
Parent Elements:	Fact, Platform
Attributes:	xml:lang
Child Elements:	*none*

<title>

This element may appear inside a Fact or Platform, and provides a human-readable title for the Fact or Platform. To support development of benchmarks and checklists for multiple languages, this element supports the ‘xml:lang’ attribute. At most one title element should appear for each language.

Content:	string
Cardinality:	0-n
Parent Elements:	Fact, Platform
Attributes:	xml:lang
Child Elements:	*none*

5. Conclusions

The XCCDF-P specification provides a powerful and extensible framework for qualifying IT and network platforms. The division between Facts and Platform definitions built from them gives XCCDF-P specifications fine-grained expressiveness and flexibility.

While XCCDF-P is designed to support XCCDF, it can be used in any context where an application needs detailed qualification of a target platform.

<<more here>>

6. Appendix A – XCCDF-P Schema

The XML Schema below describes XCCDF-P in a manner that should allow automatic validation of most aspects of the format.   Whether to validate is an implementation decision left to tool developers, but it is strongly recommended.

XCCDF-P Schema 1.1

```
<?xml version="1.0">

<xsd:schema
 targetNamespace="http://checklists.nist.gov/xccdf-p/1.1"
 elementFormDefault="qualified" attributeFormDefault="unqualified"
 xmlns:cdfp="http://checklists.nist.gov/xccdf-p/1.1"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xml="http://www.w3.org/XML/1998/namespace">

    <xsd:annotation>
      <xsd:documentation>
        This is an XML Schema for defining information
        structure about IT platforms, mainly for use with
        the eXtensible Common Checklist Description Format
        (XCCDF).  This version of the XCCDF Platform
        Specifcation (XCCDF-P) is designed to be used
        with XCCDF 1.0 or 1.1, and may also be used
        with other XML data formats that need to describe
        aspects of IT product and system platforms.

        This specification was written by Neal Ziring, based
        on ideas from the DISA FSO VMS/Gold Disk team, from
        David Waltermire and David Proulx, and from Drew
        Buttner.
        <version date="25 January 2006">1.1.0.0</version>
      </xsd:documentation>
    </xsd:annotation>

    <!-- ************************************************************ -->
    <!-- External Schemas                                           * -->
    <!-- ************************************************************ -->
    <xsd:import namespace="http://www.w3.org/XML/1998/namespace"
                schemaLocation="xml.xsd">
      <xsd:annotation>
        <xsd:documentation xml:lang="en">
          Import the XML namespace schema so that we can use
          the xml: attribute groups (particularly xml:lang).
        </xsd:documentation>
      </xsd:annotation>
    </xsd:import>

    <!-- ************************************************************ -->
```

```
<!-- Top-level Object Elements                                      * -->
<!-- ************************************************************** -->

<xsd:element name="Platform-Specification">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      This element can act as a top-level container for the
      Fact definitions and Platform definitions that make up
      a full XCCDF-P specification.  It should be used only
      when a XCCDF-P spec is being distributed as a
      standalone document.  In XCCDF, for example, the
      XCCDF platform-definitions element would act as the
      container for the XCCDF-P Facts and Platforms.

      This element schema used to include a keyref for
      Fact names, but it has been removed to allow for
      pre-defined Fact dictionaries.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="cdfp:Fact"
                   minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="cdfp:Platform"
                   minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:key name="factURIKey">
    <xsd:selector xpath="./cdfp:Fact"/>
    <xsd:field xpath="@name"/>
  </xsd:key>
</xsd:element>

<xsd:element name="Fact">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      This element denotes a single named Fact.  Every fact
      has the following:
            - name, a URI, which must be a unique key
            - title, arbitrary text with xml:lang, optional
            - remark, arbitrary text with xml:lang, optional
            - check, XML content, optional
    </xsd:documentation>
  </xsd:annotation>
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="title" type="cdfp:textType"
                   minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="remark" type="cdfp:textType"
                   minOccurs="0" maxOccurs="unbounded"/>
```

```
          <xsd:element name="check" type="cdfp:checkType"
                       minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:attribute name="name" type="xsd:anyURI"
                       use="required"/>
      </xsd:complexType>
       <xsd:unique name="factCheckSystemKey">
          <xsd:selector xpath="./cdfp:check"/>
          <xsd:field xpath="@system"/>
       </xsd:unique>
    </xsd:element>

    <xsd:element name="Platform">
      <xsd:annotation>
        <xsd:documentation xml:lang="en">
          This element denotes a single Platform definition.
          A Platform definition represents the qualifications
          an IT asset or target must have to be considered an
          instance of a particular Platform.  A Platform has
          the following:
                - id, a locally unique id
                - name, a URI, which must be a unique key
                - title, arbitrary text with xml:lang, optional
                - remark, arbitrary text with xml:lang, optional
                - definition ref, either a fact ref, a fact test,
                  or a logical test
        </xsd:documentation>
      </xsd:annotation>
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="title" type="cdfp:textType"
                       minOccurs="0" maxOccurs="unbounded"/>
          <xsd:element name="remark" type="cdfp:textType"
                       minOccurs="0" maxOccurs="unbounded"/>
          <xsd:choice minOccurs="1" maxOccurs="1">
            <xsd:element name="fact-ref" type="cdfp:factRefType"
                         minOccurs="1" maxOccurs="1"/>
            <xsd:element name="logical-test" type="cdfp:logicTestType"
                         minOccurs="1" maxOccurs="1"/>
          </xsd:choice>
        </xsd:sequence>
        <xsd:attribute name="name" type="xsd:anyURI"
                       use="optional"/>
        <xsd:attribute name="id" type="xsd:NCName"
                       use="required"/>
      </xsd:complexType>
    </xsd:element>

    <!-- ************************************************************** -->
    <!-- Supporting Element Types                                     * -->
    <!-- ************************************************************** -->
```

```
<xsd:complexType name="factRefType">
    <xsd:annotation>
        <xsd:documentation xml:lang="en">
          Type for a reference to Fact; the reference
          is always by name.  This is the type for the
          element fact-ref, which can appear in a Platform
          definition or in a logical-test in a Platform
          definition.
        </xsd:documentation>
    </xsd:annotation>
    <xsd:attribute name="name" type="xsd:anyURI"
                   use="required"/>
</xsd:complexType>


<!-- logicTestType -->
<xsd:complexType name="logicTestType">
    <xsd:annotation>
        <xsd:documentation xml:lang="en">
          Type for a test against several Facts; the content
          is one or more fact-refs, fact-tests, and nested
          logical-tests.  Allowed operators are AND and OR.
          The negate attribute, if set, makes the test
          its logical inverse (so you get NAND and NOR).
          Note that the output of a logical-test is always
          TRUE or FALSE, Unknowns map to FALSE.
        </xsd:documentation>
    </xsd:annotation>
    <xsd:choice minOccurs="1" maxOccurs="unbounded">
        <xsd:element name="fact-ref" type="cdfp:factRefType"
                     minOccurs="1" maxOccurs="1"/>
        <xsd:element name="fact-test" type="cdfp:factTestType"
                     minOccurs="1" maxOccurs="1"/>
        <xsd:element name="logical-test" type="cdfp:logicTestType"
                     minOccurs="1" maxOccurs="1"/>
    </xsd:choice>
    <xsd:attribute name="operator" use="required"
                   type="cdfp:logicOperatorEnumType"/>
    <xsd:attribute name="negate" use="optional"
                   type="xsd:boolean" default="0"/>
</xsd:complexType>


<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.  The check
        element may appear inside a Fact, giving a means
        to ascertain the value of that Fact using a
```

```
            particular checking engine.  (This checkType is
            based on the one in XCCDF, but is somewhat simpler.
            It does not include the notion of exporting values
            from the scope of an XCCDF document to the checking
            engine.)
            </xsd:documentation>
        </xsd:annotation>
        <xsd:sequence>
            <xsd:choice minOccurs="1" maxOccurs="1">
              <xsd:element name="check-content"
                           type="cdfp:checkContentType"/>
              <xsd:element name="check-content-ref"
                           type="cdfp:checkContentRefType"/>
            </xsd:choice>
        </xsd:sequence>
        <xsd:attribute name="system" 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-P tools do not process the contents.
            </xsd:documentation>
        </xsd:annotation>
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
            <xsd:any namespace="##other" processContents="skip"/>
        </xsd:choice>
    </xsd:complexType>


    <!-- ************************************************************** -->
    <!-- Supporting Simple Types                                      * -->
    <!-- ************************************************************** -->
    <xsd:simpleType name="logicOperatorEnumType">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
              Allowed operators for logic tests: we only
              have two, AND and OR.  They're capitalized
              for consistency with usage in OVAL v4.
            </xsd:documentation>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="OR" />
            <xsd:enumeration value="AND" />
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:complexType name="textType">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
              Type for a string with an xml:lang attribute.
            </xsd:documentation>
        </xsd:annotation>
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
              <xsd:attribute ref="xml:lang"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>

</xsd:schema>
```

Testing

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

7. Appendix B: Initial Dictionary of Fact Identifiers

This list below provides informal definitions for an initial dictionary of common Fact identifiers. The identifiers listed here may be used in XCCDF-P documents without explicit definitions.

`urn:xccdf:fact`

boolean – root fact, always true.

`urn:xccdf:fact:application`

Root for facts about applications and other software, always true. Facts under this subtree are arranged by application type, and each type has the same subtree of facts below it. The following application types are pre-defined:

`browser:`	**`mailagent:`**	**`calendar:`**
`officesuite:`	**`mediaplayer:`**	**`wordprocessor:`**
`java:`	**`clr:`**	**`spreadsheet:`**
`pdfreader:`	**`backup:`**	**`presentation:`**
`encryption:`	**`imaging:`**	**`terminal:`**
`chat:`	**`vm:`**	**`collaboration:`**
`server:`	**`dbms:`**	**`proxy:`**

Each of these Facts is true if the target has an application of that type installed. For example, if the target has a Java virtual machine installed, then the Fact **`urn:xccdf:fact:application:java:`** would be true. In general, each of these Facts has a child Fact for the instance of the application with further child Facts for version information. (see the **`brower:`** Fact description below).

`urn:xccdf:fact:application:browser:`

True if the target has a web browser. The subtree below this Fact is organized by names of browsers, and each has several child Facts. The following browser names are pre-defined:

`ie:`	**`netscape:`**	**`mozilla:`**
`firefox:`	**`opera:`**	**`safari:`**
`camino:`	**`aol:`**	

`urn:xccdf:fact:application:mailagent:`

True if the target has a mail user agent (MUA). The subtree below this Fact is organized by names of mail tools, and each has several child Facts (as described below). The following mail agent names are pre-defined:

`outlook:`	**`groupwise:`**	**`notes:`**
`mozilla:`	**`thunderbird:`**	**`evolution:`**
`pine:`	**`outlookexpress:`**	

`urn:xccdf:fact:application:officesuite:`

True if the target has an office automation suite installed. The subtree below this Fact is organized by names of office suites, and each has several child Facts. The following office suite names are pre-defined:

msoffice:	**openoffice:**	**perfectoffice:**
staroffice:	**koffice:**	**iwork:**

urn:xccdf:fact:application:mediaplayer:

True if the target has a video, animation, or audio player installed. The subtree below this Fact is organized by names of such player tools, and each has several child Facts. The following media player names are pre-defined:

msmedia:	**real:**	**quicktime:**
flash:	**helix:**	**itunes:**

urn:xccdf:fact:application:wordprocessor:

True if the target has a word processor application is installed on the target. The subtree below this Fact is organized by names of word processors, and each has several child Facts. The following word processors are pre-defined:

word:	**wordperfect:**	**writer:**
framemaker:	**pages:**	

urn:xccdf:fact:application:dbms:

True if the target has a database management system (DBMS) installed. The subtree below this Fact is organized by names of common DBMS products. The following names are pre-defined:

oracle:	**mssql:**	**sybase:**
db2:	**postgres:**	**mysql:**

A chain of Facts provide installed version information for applications under the Fact **urn:xccdf:fact:application:.** If the relevant version Fact is true, then it means that the target has the indicated version installed. Each part of the chain is a component of the version designation, major version first, followed by minor version, etc. Each part of the version designation must be a lexically valid component of a URI. Build number information, if present, is independent from version.

urn:xccdf:fact:application:*apptype*:*appname*:version:*major*:*minor*:*point*:...

For example, for the Mozilla web browser version 1.7.12, the following Facts would be true:

```
urn:xccdf:fact:application:browser:mozilla:version:1:
urn:xccdf:fact:application:browser:mozilla:version:1:7:
urn:xccdf:fact:application:browser:mozilla:version:1:7:12:
```

urn:xccdf:fact:application:*apptype*:*appname*:build:*number*

A fact about the build number or build designation of an installed application. For example, if build number 2158 of MSIE 6.0 were installed, the following fact would be true:

`urn:xccdf:fact:application:browser:msie:BuildNo:2158`

`urn:xccdf:fact:application:`*`apptype`*`:`*`appname`*`:edition:`*`edition_name`*

Any designation for the edition, such as "professional" or "deluxe". Such designations are common with some commercial applications, and typically provide information about supported features.

`urn:xccdf:fact:hardware`

Root for facts about hardware. Facts under this subtree describe the target's hardware Asset, including a complement of boolean Facts for the hardware's purpose or type. It is not intended to support a comprehensive description of the hardware and all its components, merely

`urn:xccdf:fact:hardware:vendor:`*`vendor_domain`*

Name of the company or organization that designed and/or sold the hardware, expressed as the company's DNS domain name. For example, for a Juniper M7 router, the following Fact would be true:

`urn:xccdf:fact:hardware:vendor:juniper.net`

`urn:xccdf:fact:hardware:model:`*`model_designation`*

Name of the hardware model, chassis, or edition. For example, for a Cisco 3725 router, the model designation is "C3725".

`urn:xccdf:fact:hardware:version:`*`major`*`:`*`minor`*`:`*`point`*`:`

This chain of fact names may be used to indicate the version of the hardware, if that is meaningful for the particular piece of hardware.

`urn:xccdf:fact:hardware:host:`

True if the target hardware is a host or general-purpose computer. For example, an HP desktop PC, an Apple Powerbook laptop, and a Sun SunFire server are all hosts.

`urn:xccdf:fact:hardware:peripheral:`

True if the target is a peripheral device or add-on device that is not intended to operate alone.

`urn:xccdf:fact:hardware:appliance:`

True if the target is a network appliance or gateway. Since there are many kinds of network appliances, there are several child Facts defined under this Fact. If a particular appliance can perform multiple functions, then multiple child Facts may be true. The following appliance subtypes are pre-defined:

`router:`	**`switch:`**	**`firewall:`**
`vpn:`	**`printer:`**	**`ids:`**
`nas:`	**`phone:`**	**`sbc:`**
`san:`		

`urn:xccdf:fact:OS:`

boolean – root for facts about the target's operating system, always true. The subtree under this Fact is somewhat complicated. First, there are string and numeric child Facts for vendor and version information. Second, there are boolean child Facts for each major operating system family (listed below). Some of the specific operating systems may have OS-specific child Facts, if the OS needs more specific description than the version and vendor information. The following OS family Facts are pre-defined:

`Windows:`	**`Linux:`**	**`Solaris:`**
`HPUX:`	**`AIX:`**	**`BSD:`**
`IOS:`	**`PalmOS:`**	**`MacOS:`**
`Netware:`	**`VxWorks:`**	**`Irix:`**

`urn:xccdf:fact:OS:vendor:`*`vendor_domain`*

Name of the company or organization that designed and/or sold the hardware, expressed as the company's DNS domain name. For example, for MacOS X, the following Fact would be true.

```
urn:xccdf:fact:OS:vendor:apple.com
```

`urn:xccdf:fact:OS:edition:`*`edition_name`*

Edition of the operating system, if that is relevant. Some operating systems are sold with several different feature sets, designed with edition names that are independent of the version number. For example, for a given target running the Microsoft Windows XP OS, one of the following Facts would be true.

```
urn:xccdf:fact:OS:edition:home:
urn:xccdf:fact:OS:edition:professional:
urn:xccdf:fact:OS:edition:embedded:
```

`urn:xccdf:fact:OS:language:`*`lang_id`*`:`*`locale_id`*`:`

This chain of Facts can be used to designate the ISO language and locale identifier for the OS, if applicable. The language must be given as one of the official ISO language digraphs, such as "en". The locale may also be given as a nation digraph, such as "uk".

`urn:xccdf:fact:OS:version:`*`major`*`:`*`minor`*`:`*`point`*`:`

This chain of Facts can be used to indicate the installed version of the OS.

`urn:xccdf:fact:OS:build:`*`build_number`*`:`

The OS image build number or build designation.

`urn:xccdf:fact:OS:Windows:`

True if the OS family is Microsoft Windows. Because it is conventional to refer to different editions of the Windows platform by name, this Fact has a child Fact for each of major release names:

`95:`	**`98:`**	**`ME:`**	**`NT:`**
`2000:`	**`XP:`**	**`2003:`**	**`Vista:`**

`urn:xccdf:fact:OS:Windows:ServicePack:`*`svc_pack_number`*`:`

This Fact will be true for every service

`urn:xccdf:fact:OS:Solaris:trusted:`

True if the target OS is a member of the Sun Solaris family and has the Mandatory Access Control (MAC) feature set common called 'Trusted Solaris'.

`urn:xccdf:fact:OS:Linux:kernel:version:`*`major`*`:`*`minor`*`:`*`point`*`:`

The version information for the Linux kernel, which is independent of the version Fact chain for the OS distribution release. For example, if a Linux target system used kernel release 2.6.4, then the following Fact would be true.

```
urn.xccdf:fact:OS:Linux:kernel:version:2:6:4:
```

`urn:xccdf:fact:service:`

Root for facts about services running and available on the target system, always true. In this context, services are local and network services like file sharing, web, directory, remote login, remote display, routing, chat, etc. The subtree below this Fact is organized by protocol name. The following service names are pre-defined:

ftp:	**ssh:**	**telnet:**	**tftp:**
lpr:	**dns:**	**ntp:**	**http:**
smtp:	**pop:**	**imap:**	**nntp:**
tacacs:	**radius:**	**ldap:**	**nis:**
smb:	**wins:**	**nds:**	**nfs:**
rdp:	**vnc:**	**ica:**	**x11:**
irc:	**xmpp:**	**iiop:**	**soap:**
rip:	**ospf:**	**isis:**	**bgp:**
ripng:	**ospfv3:**	**eigrp:**	**pim:**
syslog:	**socks:**	**ntdomain:**	**snmp:**
kerberos:	**ocsp:**	**certificateauthority:**	

`urn:xccdf:fact:service:`*`svcname`*`:version:`*`major`*`:`*`minor`*`:`*`point`*`:`

This Fact chain indicates the version of the software that provides the given service. For example, if the SSH service were provided by OpenSSH 3.9p1, then the following Fact would be true:

```
urn:xccdf:fact:service:ssh:version:3:9:
```

`urn:xccdf:..`

8. References

[1] Fallside, David C., *XML Schema Part 0: Primer*, W3C Recommendation, 2 May 2001. (http://www.w3.org/)

[2] Buttner, Andrew, "<Oval SQL='false'>", presentation, The MITRE Corporation, Oct 2003.

[3] Baker, Mark *et al*, *XHTML Basic*, W3C Recommendation, Dec 2000. (http://www.w3.org/)

[4] Bray, Tim *et al*, *Namespaces in XML*, W3C Recommendation, Jan 1999. (http://www.w3.org/)

[5] Waltermire, David "XCCDF Platform Specification", Center for Internet Security, Sep 2004.

[6] Ziring, N. and Wack, J. "Specification for the Extensible Configuration Checklist Description Format (XCCDF)", Version 1.0, NIST IR 7188, Jan 2005.

[7] Wojcik, M., Proulx, D., Baker, J. and Roberge, R. "Introduction to OVAL", The MITRE Corporation, Jul 2005.

[8] Davis, Mark "Unicode Regular Expressions", Unicode Technical Recommendation No. 18, version 9, January 2004.

[9] Ziring, N. and Grance, T. "Specificationf ro the Extensible Configuration Checklist Description Format (XCCDF)", Version 1.1, NISTIR 7275, December 2005.