XCCDF: Discussion Topics and Proposals

Open issues in XCCDF to be covered at the Security Automation Developer Days Workshop, March22 & 23, 2011

This document covers a range of open issues in XCCDF that have been raised by community members. This document organizes these issues into two categories: discussion topics and fix proposals. Discussion topics are issues that, in the opinion of the moderator, are likely to produce multiple opinions as to if and how they should be implemented and therefore would benefit from broader community discussions. Discussion topics include a list of issues to discuss as well as one or more sample solutions that are intended as starting points for community discussion. Fix proposals outline issues that, in the moderator's opinion, are low impact and where the ways to address the issues are relatively straightforward. Fix proposals simply outline the issue and provide a detailed write-up of how the fix would be accomplished. By way of example, additions of new XCCDF functionality would be listed under discussion topics but fixes of obvious typos would appear as fix proposals, although admittedly there comes a point where the distinction gets rather fine. At this time, it is not anticipated that there will need to be significant discussion on fix proposal issues and they are included here as a means of informing the community. However, if community members raise concerns with these issues or the solution proposed then these concerns can be discussed with the broader community. The authors wish to emphasize that listing of an issue as a "fix proposal" in this document is not meant to restrict discussion - it simply indicates that the moderator's expectation that the community will have little to discuss on the topic. Community members should feel free to comment on all issues listed in this document regardless of their classification.

These issues and the proposals associated with them use the DRAFT XCCDF 1.2 specification as a baseline, even though the changes that are approved by the community at this event will become part of the final XCCDF 1.2 specification. The decision to use the draft release as the baseline reflects the community consensus that the changes introduced in XCCDF 1.2 are all acceptable – a consensus recently confirmed by the lack of any objections during the public comment period for the draft document. (There were comments proposing the inclusion of additional features, and those proposals appear below for discussion. There were not, however, any objections to any of the changes between XCCDF 1.1.4 and the draft XCCDF 1.2.)

Virtually all issues in this document appear in NIST's JIRA issue tracker. This list is publicly available at https://services.nvd.nist.gov/jira/browse/XCCDF.

Issues are listed in order in which they appear in the proposed agenda for the day's events. While the list is long, most of the discussion topics are comparatively minor issues and are expected to require little discussion and the fix proposals will likely require none at all. As such, while the workshop may not get through the entire list of topics, it is expected that we will be able to get through a majority of them.

Table of Contents

Discussion Topics

Fix Proposals

Linear vs. Branching Policies

Currently, the guidance encapsulated in XCCDF might be described as "linear" in that it is a list of recommendations. Profiles add some flexibility, but in the end, each Profile is itself a list of recommendations (chosen from the Benchmark as a whole). Any further variation to accommodate an enterprise's needs and configurations must be accomplished through manual tailoring.

Some community members have suggested that there might be a benefit to supporting a more tree-like policy structure that can naturally accommodate variation at the sub-policy level. One example could be a variation in policy for environments supporting IPv4 vs. IPv6 – each variant could have a different set of recommendations, and an environment would only need to employ one variant. Another example could be the configuration of a large, multi-purpose server, such as Microsoft Exchange. Microsoft Exchange can serve multiple distinct Internet protocols, but an environment is only likely to employ a subset of them. Enterprises would wish to follow recommendations that secured the protocols they wished to use while turning off the protocols they do not need. While one could create separate Profiles for each combination, as the number of options increases, the number of Profiles needed to describe all possible combinations grows exponentially and quickly becomes impractical. As such, community members have suggested that having the ability to compose a benchmark from multiple, alternative chunks (sub-Profiles) would be a useful way to handle the natural variability encountered in modern enterprises. This corresponds to issue XCCDF-59 in NIST's issue tracker.

It should be emphasized that this is not a revisiting of the previously dismissed proposal for conditional checking. That proposal revolved around the idea that the result of one automated check could guide the selection of subsequent checks, potentially resulting in multiple exchanges between a testing server and a target. The proposal outlined today is focused on the tailoring step and does not require any interactions with the target platform.

Issues

1. Right now, creating a Profile is straightforward in that, for each Profile, the final set of Rules and Values can be easily determined. If we allow multiple Profiles to be combined, this becomes significantly more complicated because each Profile (or sub-Profile) could be combined in multiple ways. Thus Profile authors become responsible for ensuring correct behavior through a multitude of possible selections. Is the added functionality worth the added complexity?
2. Right now, one has the option of selecting no Profile and using the default settings. However, if we have a set of alternative sub-Profiles, we may want to ensure that at least one is selected. To use a previous example, if we have a sub-Profile for securing IPv4 and a sub-Profile for securing IPv6, we would want to ensure that one of them was selected or the IP mechanism would not receive any hardening.
3. It is possible that not all combinations of sub-Profiles would meet the author's intents. How would one ensure that invalid combinations of Profiles were not selected?
4. Should the order in which sub-Profiles are selected matter? Right now, extending Profiles can override extended Profiles because the former's selectors appear after the latter's selectors. If, however, we are combining multiple Profiles together, how would the order

of selector application (which is ultimately necessary when applying the combined Profile to the Benchmark) be determined?

Sample Proposal

A new selector type, called "include" will be added to the base Profile structure. The include selector will have mandatory "profile-ref" attribute (list of NCName) along with the optional attributes "default" (NCName) and "required" (boolean). A Profile may use the include selector to insert the contents of another Profile at the given location. The list of Profiles in the profile-ref attribute indicate the possible Profiles that could appear. The default attribute indicates the default Profile to insert (which serves for noting a particular Profile in a user interface, not to identify a Profile in the absence of a user selection) and the required attribute notes whether one of the listed Profiles must be selected (value of true) or if the user can chose not to select any.

Under the behavior described above, Profile extension is equivalent to adding an include selector at the beginning of the extending Profile. Specifically, if Profile A extends Profile B, the same final Profile could be achieved by inserting <include profile-ref="B" required="true"> at the beginning of Profile A. Note that included Profiles can be abstract.

This proposal adds the flexibility for authors and users to utilize branching within Profiles while allowing authors to constrain the branching. Specifically, authors can ensure that only known Profiles get inserted and that they are only inserted at specific places. They can even note when it is permissible to select none of the Profiles at a particular junction vs. places where tailoring users are required to pick from a set of choices. This mechanism helps with re-use as well: blocks of selectors common to multiple Profiles can be pulled out into their own abstract Profile and included at the appropriate junctures where appropriate.

The following shows the XML modifications and additions to the XCCDF schema:

```
<xsd:complexType name="profileIncludeType">
  <xsd:sequence>
    <xsd:element name="remark" type="cdf:textType" minOccurs="0"
      maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="profile-list" use="required">
    <xsd:simpleType>
      <xsd:list itemType="xsd:NCName"/>
    </xsd:simpleType>
  </xsd:attribute>
  <xsd:attribute name="default" type="xsd:NCName" use="optional"/>
  <xsd:attribute name="required" type="xsd:boolean" default="false" use="optional"/>
</xsd:complexType>

<xsd:complexType name="profileType">
    <xsd:sequence>
      ...
      <xsd:choice minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="select" minOccurs="0" type="cdf:profileSelectType"/>
        <xsd:element name="set-complex-value" minOccurs="0"
```

```
            type="cdf:profileSetComplexValueType"/>
          <xsd:element name="set-value" minOccurs="0" type="cdf:profileSetValueType"/>
          <xsd:element name="refine-value" minOccurs="0"
            type="cdf:profileRefineValueType"/>
          <xsd:element name="refine-rule" minOccurs="0" type="cdf:profileRefineRuleType"/>
          <xsd:element name="include" minOccurs="0" type="cdf:profileIncludeType"/>
        </xsd:choice>
        ...
      </xsd:sequence>
      ...
</xsd:complexType>
```

External Profiles

External Profiles are structures that can exist outside a given XCCDF Benchmark document and which provide a list of tailoring selections for that Benchmark. The objective of an external Profile is to store custom tailoring instructions for a Benchmark in a standardized way without changing the source document itself. Changing the source document (rather than using an external Profile) would invalidate any signatures associated with the source document and could also lead to document management challenges as one dealt with multiple different copies of what was intended to be a single Benchmark.

This issue was discussed in 2010 and, at that time, the community decided not to create any special structures in the XCCDF language to support this capability. At the time it was felt that existing methods outside of XCCDF, such as XSLT stylesheets, could perform equivalent activities without adding more complexity to XCCDF itself. Since that time, multiple parties have requested that this decision be re-examined. In addition, some vendors have adopted custom structures that, in effect, duplicate the intended purpose of an external Profile. Given the multiple calls as well as real-world examples of cases where external Profiles would be useful, it was felt that review of this issue was justified. This corresponds to issue XCCDF-3 in NIST's issue tracker. (Note that this appears under "reopened" issues rather than "open" issues, where all the other issues can be found.)

The original discussion of external Profiles included some fairly powerful capabilities. Subsequent discussions have indicated that most of these capabilities (e.g., having a single external Profile pull from multiple Benchmark documents or defining its own additional Rules and Values) do not appear to be needed at this time. While nothing is considered off the table, the discussion here will begin by focusing on a very basic external Profile that points to a single Benchmark and which includes no capabilities beyond what a Benchmarks own internal Profiles could perform.

Issues

1. Previous discussions suggested that XSLT (or some similar mechanism) could be used to address the underlying needs an external Profile would meet. Would it be sufficient for the community to agree on such a mechanism, or are changes to XCCDF truly necessary to ensure interoperability.
2. Should external Profiles be able to extend Profiles in the source document, or should they simply select/refine all the Rules and Values of interest in the source Benchmark?
    a. If we allow references to Profiles and branching Profiles are supported (see the issue titled "Linear vs. Branching Policies") how would selection of the include statements in an extended Profile be handled? Should the external Profile be allowed to have its own include statements that could point to multiple Profile elements in the source (create its own tree through the source document) or should it simply extend an existing Profile (simply indicate a path through one of the source-document's pre-defined trees).
3. Should external Profiles explicitly identify the Benchmark they modify?

Sample Proposal #1

This proposal is a relatively minimal solution. It can point to another Profile (using the id attribute), which effectively allows the external Profile to extend the named Profile. Because this

external Profile is meant to preclude the need for further user tailoring, the refine-value selectors are not included and only explicit setting of Values is supported. The root Tailoring structure includes a "description" element where comments can be included.

```
<xsd:element name="Tailoring">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      The Tailoring tag is the top level element representing a
      customization of a security checklist, including descriptive text.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="description" type="tlr:descriptionType"
        minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="profile" type="tlr:profileType" minOccurs="0"
          maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:complexType name="profileType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      The Profile element contains all the customizations for the specified profile.
      The id must match the id of a profile in the Benchmark for which this tailoring
      applies; it may be empty to indicate that the tailoring is to apply when an audit is
      performed with no profile selected.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element name="select" type="tlr:selectType" minOccurs="0"
          maxOccurs="unbounded"/>
    <xsd:element name="set-value" type="tlr:setValueType" minOccurs="0"
          maxOccurs="unbounded"/>
    <xsd:element name="set-complex-value" type="tlr:setComplexValueType"
          minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="refine-rule" type="tlr:refineRuleType" minOccurs="0"
          maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="id" type="xsd:Name" use="required"/>
</xsd:complexType>

<xsd:complexType name="descriptionType">
  <xsd:annotation>
```

```
      <xsd:documentation xml:lang="en">
        The Description element contains text given by the author as to why this tailoring is
        in place.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute ref="xml:lang"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
```

Sample Proposal #2

This proposal is a more powerful solution that re-uses the Profile structure already present in XCCDF. All fields, including the "extends" attribute, are available for use. Note that, because we are actually extending the profileType, subsequent changes to the schema supporting internal Profiles would also appear in external Profiles. (For example, addition of the include element in internal Profiles would add this ability to external Profiles, although exceptions could always be made.)

The only new property is an optional "benchmark" element. This element duplicates the structure of the benchmark element in TestResults with the addition of an optional version field. The intention of this is to provide a pointer to the source Benchmark that the external Profile tailored. (If this is acceptable, the community might consider the creation of a new type that was referenced by both the TestResults and external Profile structures to standardize structure.)

```
  <xsd:element name="ExternalProfile">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="cdf:profileType">
          <xsd:sequence>
            <xsd:element name="benchmark" minOccurs="0">
              <xsd:complexType>
                <xsd:attribute name="href" type="xsd:anyURI" use="required"/>
                <xsd:attribute name="id" type="xsd:NCName" use="optional"/>
                <xsd:attribute name="version" type="xsd:string" use="optional"/>
              </xsd:complexType>
            </xsd:element>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>
```

Automated Profile Selection

At the 2010 IT Security Automation Conference in Baltimore, a suggestion was made that it could be useful to support logic to allow for automated selection of Profiles. It was noted that this could further reduce the necessity for human involvement in cases where automated determinations could be made. The discussion proposed that the actual logic for Profile selection would not necessarily need to be defined within XCCDF and that instead existing SCAP languages (such as OVAL or OCIL) or a new Applicability Language could be referenced by XCCDF in a manner similar to the way that an XCCDF Rule's check structure can reference checking languages. This corresponds to issue XCCDF-63 in NIST's issue tracker.

XCCDF already has the ability to use the platform element in Benchmarks, Profiles, Rules, and Groups to automatically determine that the associated structure is inappropriate for a detected platform. In the proposal for the automated Profile selection, a few significant differences were noted:

1. The platform element only currently affects automated processing for Groups and Rules, causing them to behave as if they were deselected. Currently, the platform element is only informational in Benchmarks and Profiles.
2. In Groups and Rules, the platform element currently can only be used to de-select components. In the proposal, we are considering a structure that would select a Profile, which is not precisely an equivalent behavior.
3. Currently, CPEs identify software and platforms, but the specific examples given during the initial discussion of this proposal center on roles in an environment. Specific examples included the roles of Domain Controller in a Windows enterprise or external web server. These roles might not always be distinguishable solely by identifying the present software. As such, the proposal considered using logic beyond simple software inventory descriptions, which is what is used in the case of the platform element.

Issues

1. Is there a need for this behavior? It seems that this feature would be primarily useful where a single XCCDF document contained recommendations for multiple machines in different roles. Currently, virtually all published content is role-agnostic and traditionally splits content within a single document by security level for a single platform (although there has been some XCCDF content where a single document contained recommendations for multiple, closely related platforms). If multi-role content does not get produced, this seems to be of limited utility.
2. Should a new structure be added or should the platform structure, which as noted has the strongest alignment with this proposal, be repurposed?
    a. If the platform structure is re-purposed, should all platform structures be addressed or only the platform structures in a Profile?
3. Do existing check languages have sufficient capability to express the necessary conditions for this proposal? If not, what capability is missing?
4. Should automated Profile selections be capable of indicating "no Profile" (that is, use the default behaviors of the Benchmark's content without applying a Profile)? Should automated Profile selection support selecting multiple Profiles, thus serving to winnow the initial list of Profiles before ultimately relying on the user to make a final determination?

5. If we allow branching profiles (see the issue titled "Linear vs. Branching Policies") how would these be handled during automated selection.

Sample Proposal

A new, optional element named profile-selectors would be added to the Benchmark element. This, in turn, would contain one or more profile-select elements. Each profile-select element would contain a single check element using the new profileCheckType data type. The profileCheckType would be a reduced version of the existing checkType that is used in Rules, but with the check-import, check-export, and selector properties removed. (These are removed because meaningful use of the check-export and selector properties can only happen after tailoring and the processing of the new structures must happen before tailoring.) In addition to the check element, each profile-select element would contain an attribute named profiles containing a list of one or more Profile ids. The default value of the profiles attribute would be an underscore (_), which represents no selected profile.

When a Benchmark is processed, between the Benchmark.Front and Benchmark.Profile sub-steps, the profile-selectors element would be processed. If the profile-selectors element was present, a tool would go through each profile-select element in order. For each profile-select element, the tool would initiate a check based on the referenced checking language. Processing of the check would be identical to processing of a Rule's check, including cases where multiple check-content-refs are present in a single check. If a given check returns false, the tool would proceed to the next profile-select element and repeat the process. (Note that there is no reason that all checks could not be sent at once as long as the tool processed the results of these checks in order.) This would continue until one of the profile-select checks returned true, or all of the profile-select checks had returned false. In the former case, the list of effective Profiles would be set equal to list of profiles given in the profiles attribute and no further profile-select elements would be considered. If this list contained a single Profile id, that Profile could be automatically applied. If this list contained multiple Profile ids, the user would be prompted to select from that list of ids. An underscore (either explicit, implicit from the default value of the profiles attribute, or as part of a list of Profile id values) would indicate the inclusion of "no Profile" (that is, Benchmark.Profile is skipped during Benchmark processing) in any list of effective Profiles. If every profile-select check returns false, or if there is no profile-selectors element (or if a given tool does not support profile-selectors) then no automation is performed and the user must manually select a Profile, as is the case today.

For record keeping, a modification is also made to the profile element of the TestResult structure. Specifically, a new attribute named profile-select-ref is added. If during profile-selectors processing, a profile-select's check returns true resulting in its setting of a list of effective Profiles, then the profile-select-ref attribute would be set to the value of the corresponding check's id attribute. The profile-select-ref would be absent in cases where the profile-selectors structure did not provide any automation. (Note, the profile-select-ref would still be present in cases where automated Profile selection only reduced the list of applicable Profiles, but where the user still needed to make a final determination.) To completely align with the conventions described above, if "no Profile" were selected through automated means, the profile element would still be present in the TestResult element with its idref attribute set to underscore (_). To ensure that reporting in TestResults was useful, constraints would be added on the id attributes of all check elements in profile-select elements to ensure they were mutually

unique. For a similar reason, the id field in profile-select checks would be required, rather than the optional attribute of Rule checks.

Below is a summarized version of the proposed Benchmark.ProfileSelection processing step (inserted between Benchmark.Front and Benchmark.Profile):

- If profile-selectors is present and the tool supports profile-selectors processing
  - For each profile-select element
    - If check evaluates to true
      - Set list of effective Profiles equal to profile-select's profiles attribute
        - If there is more than one effective Profile present to the user for final Profile determination
      - In the TestResult, set the profile-select-ref equal to the value of the profile-select's check element's id attribute
      - End profile-selectors processing and proceed to the Benchmark.Profile processing step using the single selected Profile.
- Have the user select from the complete list of Profiles

The new XML structures would appear as follows:

Create type for holding automated Profile selection logic

```
<xsd:complexType name="profileAutoSelectType">
  <xsd:sequence>
    <xsd:element name="profile-select" minOccurs="1" maxOccurs="unbounded">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="check">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="check-content-ref" minOccurs="0"
                  maxOccurs="unbounded" type="cdf:checkContentRefType"/>
                <xsd:element name="check-content" minOccurs="0" maxOccurs="1"
                  type="cdf:checkContentType"/>
              </xsd:sequence>
              <xsd:attribute name="system" type="xsd:anyURI" use="required"/>
              <xsd:attribute name="negate" type="xsd:boolean" use="optional"
                default="false"/>
              <xsd:attribute name="id" type="xsd:NCName" use="required"/>
              <xsd:attribute ref="xml:base"/>
            </xsd:complexType>
          </xsd:element>
        </xsd:sequence>
        <xsd:attribute name="profiles" default="_" use="optional">
          <xsd:simpleType>
            <xsd:list itemType="xsd:NCName"/>
```

```
            </xsd:simpleType>
          </xsd:attribute>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
```

Add structure to Benchmark to record automated Profile selection structures

```
  <xsd:element name="Benchmark">
      ...
    <xsd:complexType>
      <xsd:sequence>
      ...
        <xsd:element ref="cdf:model" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="profile-selectors" minOccurs="0" maxOccurs="1"
            type="cdf:profileAutoSelectType"/>
        <xsd:element ref="cdf:Profile" minOccurs="0" maxOccurs="unbounded"/>
      ...
      </xsd:sequence>
      ...
    </xsd:complexType>
      ...
    <xsd:unique name="profileAutoSelectCheckKey">
      <xsd:selector xpath="./cdf:profile-selectors/cdf:profile-select/cdf:check"/>
      <xsd:field xpath="@id"/>
    </xsd:unique>
  </xsd:element>
```

Add structure to TestResults to record automated Profile selection

```
  <xsd:complexType name="testResultType">
      ...
      <xsd:element name="profile" type="cdf:idrefType" minOccurs="0"
        maxOccurs="1">
        <xsd:complexType>
          <xsd:complexContent>
            <xsd:extension base="cdf:idrefType">
              <xsd:attribute name="profile-select-ref" use="optional"/>
            </xsd:extension>
          </xsd:complexContent>
        </xsd:complexType>
      </xsd:element>
      ...
  </xsd:complexType>
```

Value Units

It has been proposed that XCCDF Values be augmented to include an indicator of the units to display to the user. Many tailoring selections may use unintuitive units (for example, providing a time in milliseconds when the UI control only allows whole minutes) and providing this information could give hints to the user that help them understand an otherwise confusing value. This corresponds to issue XCCDF-60 in NIST's issue tracker.

The solution to this issue can range in complexity. At the simplest, a units annotation simply tells the user what units are used with the Value's values. (For example, an annotation of "milliseconds" would inform the user that the potential values were millisecond counts.) At a more powerful level, XCCDF could perform mathematical conversions to present a more natural value to the user. (For example, a tailoring tool could present a time in minutes or seconds even though the actual value passed to a checking system would need to present a time in milliseconds.)

Issues

1. Selectors are strings and, as such, can provide meaningful information to users. Would it be sufficient to establish a convention that states that selectors serve as the display text for the selection of values?
    a. There are some negatives to this approach. First, only a single selector string can be given, which would prevent the creation of different language texts for a given selection. Second, selectors adjust multiple fields in a Value in addition to the value itself, so it is possible that two selectors might identify equivalent values but different auxiliary fields. The author would need two different strings to express the same value. Finally, selectors provide no benefit if the user is manually providing a value.
2. Should units come from a controlled vocabulary (separate from XCCDF, possibly in collaboration with CCE) or should any text be permitted?

Sample Proposal

An optional "units" element may appear in each Value. Within the units element may be one or more "unit" elements, the value of which is a string that would be displayed to a user during tailoring. If more than one unit elements appear, they must all have different xml:lang attributes.

In addition, a units element may have an optional "convert" attribute. The value of the convert attribute consists of an operator (+, -, *, /) followed by a digit. When values are presented to the user, the given operation occurs over the values for display. For example, if the convert attribute's value was "/60000", the value of each value and default entry in a Value would be divided by 60000 before it was displayed to the user. In this way, one could have a unit value of "minutes" and display values in minutes even though the underlying data passed to the checking system was given in milliseconds. If a user provides a manual value, the inverse of the operation would be applied before the Value was set. (In our example, the user's input would be multiplied by 60000.) Note that the convert attribute is only applicable for numeric values and is ignored in all other cases. Note also that the convert attribute applies to all the unit elements in a units structure.

```
<xsd:complexType name="valueType">
    <xsd:complexContent>
      <xsd:extension base="cdf:itemType">
        <xsd:sequence>
            ...
          <xsd:element name="choices" type="cdf:selChoicesType"
            minOccurs="0" maxOccurs="unbounded"/>
          <xsd:element name="units" minOccurs="0" maxOccurs="1">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="unit" minOccurs="1" maxOccurs="unbounded" >
                  <xsd:complexType>
                    <xsd:simpleContent>
                      <xsd:extension base="xsd:string">
                        <xsd:attribute ref="xml:lang" use="required"/>
                      </xsd:extension>
                    </xsd:simpleContent>
                  </xsd:complexType>
                </xsd:element>
              </xsd:sequence>
              <xsd:attribute name="convert" use="optional">
                <xsd:simpleType>
                  <xsd:restriction base="xsd:string">
                    <xsd:pattern value="[\-\+\*/][0-9]+"></xsd:pattern>
                  </xsd:restriction>
                </xsd:simpleType>
              </xsd:attribute>
            </xsd:complexType>
            <xsd:unique name="uniqueLangRef">
              <xsd:selector xpath="unit"/>
              <xsd:field xpath="@xml:lang"/>
            </xsd:unique>
          </xsd:element>
          <xsd:element name="source" type="cdf:uriRefType"
            minOccurs="0" maxOccurs="unbounded"/>
            ...
        </xsd:sequence>
            ...
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
```

Unique Benchmark Identifiers

Currently, the id attribute of a Benchmark is of type NCName with no further restrictions on its form. The id field is intended to uniquely identify a Benchmark document. Within XCCDF, the Benchmark id attribute is used within a TestResult entity to link a given result to a given source Benchmark, specifically in cases when the two reside in different files.

Unfortunately, the requirement that id attributes be unique has proven an elusive goal. Content producers have not always been diligent in giving different ids to different versions of a given Benchmark, and different content producers have, on occasion, assigned the same id to their respective content. This not only causes a problem for linking TestResults reliably back to their source Benchmark, but many tools have been written with the assumption that the id field is, in fact, unique and can become confused when this turns out not to be the case.

It has been suggested that conventions be adopted that will better ensure the global uniqueness of Benchmark id attributes. In previous discussions, the community suggested that using a simple Globally Unique Identifier (GUID) algorithm would address this challenge in a relatively straightforward manner. This corresponds to issue XCCDF-58 in NIST's issue tracker.

Issues

1. If a GUID is used, should this be in addition to the current id field (e.g., add a separate GUID attribute to the Benchmark element) or should it be applied to the id field. (Note that using a new field means that existing references to the Benchmark id, such as in the TestResults element, would not benefit from this change without additional changes.)
2. Should conventions on the id attribute be enforced by the schema, or should they simply be noted in the XCCDF specification? (Note that modifying the schema to enforce constraints that would reduce the chance of id collision will, by definition, deprecate at least some content given that we have examples of id collisions today. Such a change would need to take place in a major release.)

Sample Proposal

The specification will recommend that the Benchmark id field contain a 32-character hexadecimal GUID. The specific algorithm as well as the seed used to create the GUID is not standardized in XCCDF and individual authors can use whichever GUID procedures they wish. No formatting restrictions are added to the schema, allowing non-GUID values to appear in the id field, although this would be intended for backward compatibility and future documents would be discouraged from using ids other than GUIDs.

Specifically, the description of the id Property in the Benchmark properties table (page 15) would read, "Benchmark identifier; authors SHOULD use a GUID", where SHOULD is interpreted as described in the IETF's RFC 2119.

Record Platform Testing Artifacts

Currently, Benchmarks and individual Groups and Rules may utilize <platform> elements to identify the systems for which they are applicable. During XCCDF processing the XCCDF interpreter should determine the type of platform that is being assessed and de-select Rules if they are marked as being applicable only to a different type of platform. While, in practice, most tools use OVAL inventory definitions, in theory any mechanism could be utilized as there is no method prescribed by XCCDF. This could result in disagreement between tools as to which Rules are applicable on a given platform. Moreover, while TestResults contain a list of the identified platforms used to make platform-based applicability determinations, there is no record of how this list was arrived at.

It has been suggested that TestResults should contain links to files containing artifacts that resulted in a particular platform determination. For example, in the case of OVAL, such a link would identify an OVAL results file. This would provide a means for reviewers to see what specific tests were performed in the process of making a platform determination. This corresponds to issue XCCDF-71 in NIST's issue tracker.

Issues

1. Not all tools may make platform determinations using mechanisms that produce artifacts that can be referenced. We want to make sure that the recording of artifacts does not implicitly limit the ways the platform computation can be made.
2. Currently, records of how checks arrive at their results can be recorded within XCCDF but very few vendors do this. Would new structures to record records of platform computations even be used?
3. If there is interest in recording artifacts to record how a platform determination was made, is there interest in providing explicit references to checking information to make that determination in the first place? This is to say, should a Benchmark explicitly identify the checking instructions used to perform platform applicability instead of (or in addition to) naming the appropriate platform.

Sample Proposal

This extends the platform element in the TestResult to include a check element. The purpose of this check element would be to indicate the file and, optionally, the specific artifact corresponding to a particular platform discovery. (See the section titled "Conventions for use of Check Fields in rule-results" for more about this type of reference.) Tools could use this field to indicate how platform determinations were made and what specific results led to a given conclusion. The check structure could appear under each platform element allowing the specific results for each platform determination to be pinpointed.

```
<xsd:complexType name="testResultType">
    <xsd:sequence>
        ...
        <xsd:element name="platform" type="cdf:CPE2idrefType" minOccurs="0"
                maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:complexContent>
              <xsd:extension base="cdf:CPE2idrefType">
```

```
                <xsd:sequence>
                  <xsd:element name="check" type="cdf:checkType"/>
                </xsd:sequence>
              </xsd:extension>
            </xsd:complexContent>
          </xsd:complexType>
        </xsd:element>
        ...
      </xsd:sequence>
      ...
    </xsd:complexType>
```

Conventions for use of Check Fields in rule-results

Individual rule-result elements may contain check elements. It has been noted that there are multiple interpretations as to how these structures could be used. Specifically, these structures could be made to duplicate the structures in the corresponding Rule, thus identifying which of potentially several checks was actually utilized to create a given result. Alternatively, it has been suggested that check structures in rule-results could reference checking result structures to record the findings that led to a given result. To give examples using OVAL, the former usage would have rule-result check structures pointing to OVAL definitions, while the latter usage of the same structure would point to OVAL system-characteristics or results structures.

Currently the check element in rule-results is not widely used, but as the community becomes more interested in both tracking the source and findings of a given check, this field is of obvious interest and a divergence of practices regarding its use could lead to compatibility challenges. It has been suggested that XCCDF establish conventions for the use of this field to pre-empt such issues. This corresponds to issue XCCDF-72 in NIST's issue tracker.

Issues

1. Is the XCCDF specification the appropriate place to establish this kind of convention, or should it be raised to a higher level specification (e.g., SCAP)?
2. Are both uses worth preserving such that we wish to add a new field so both the source and the findings can simultaneously be recorded?
    a. Even if we wish to report both the source and the findings, is the check field the appropriate way to do so for either – in other words, would a different structure for each purpose be a good idea.
3. Rules contain either a check or complex-check element. Currently, rule-results may only contain an optional check element. Should rule-results be changed to allow complex-checks as well?

Sample Proposal

This proposal comes in two separable parts. (I.e., each part could be adopted independently of the adoption of the other.)

Part 1

The specification will recommend that, if it is present, the check field in rule-results should be used to point to the check-system result structures that led to the given Rule result. For example, in the case of OVAL, the check in a rule-result would point to the corresponding OVAL definition element within an OVAL results file. Specifically, the rule-result's check element could contain a check-content-ref with a href attribute set equal to the name of the OVAL results file and a name attribute set equal to the definition_id attribute of the definition element. Alternately, the check element could contain a check-content element that contained the actual OVAL definition element from the result file. In either case, the system attribute of the check element itself would correspond to the XML target namespaces of the document to which the reference was made. In the case of OVAL 5, this would be "http://oval.mitre.org/XMLSchema/oval-results-5". (It should be noted that, while OVAL uses a different XML target namespace for its source and result files, other languages, such as OCIL, might not.)

Part 2

In addition to the above change, the schema will be modified to add the possibility of including complex-check elements in rule-results. This will allow rule-results to document the bases of a result regardless of whether the source Rule uses a check or complex-check structure. Each component check within a complex-check element would follow the conventions outlined for individual check elements in rule-results.

```
<xsd:complexType name="ruleResultType">
    ...
  <xsd:sequence>
    ...
    <xsd:element name="fix" type="cdf:fixType" minOccurs="0"
      maxOccurs="unbounded"/>
    <xsd:element name="check" type="cdf:checkType" 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:sequence>
    ...
</xsd:complexType>
```

Use AI Spec in Test Results

Currently, the identity of an assessed system is recorded in the TestResult structure using structures like target, target-address, and target-facts. There is a desire to use a more standards-based format for identifying assets. Recently, a new NIST IR has emerged detailing a specification for detailing asset information: Asset Identification (AI, NIST IR 7693, http://csrc.nist.gov/publications/drafts/ir7693/draft-NISTIR-7693-AI_20101204.pdf). It may be beneficial to include references to target information encoded in AI, as an alternative to the XCCDF-specific methods currently employed. This corresponds to issue XCCDF-70 in NIST's issue tracker.

Issues

1. AI is a relatively new standard and may undergo several revisions as adoption grows. Is this the appropriate time to start referencing the new specification or should this be delayed until the AI spec has achieved more maturity?
2. Would AI be an alternative to the current target fields in a TestResult (allowing users to utilize whichever structure they desired) or would it be intended to replace them (in which case the target fields in TestResult would be deprecated)?

Sample Proposal

The XCCDF IdentityType is modified so as to encapsulate the location of an Asset Identification instance document (in an "href" attribute) and a reference to a specific asset entry (a "name" attribute) contained within it. TestResult elements would no longer record target information inline, so target, target-address, and target-facts would be deprecated.

```
<xsd:complexType name="identityType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="href" type="xsd:anyURI" use="required"/>
      <xsd:attribute name="name" type="xsd:string"/>
      <xsd:attribute name="authenticated" type="xsd:boolean" use="required"/>
      <xsd:attribute name="privileged" type="xsd:boolean" use="required"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```

Allow Variations of the Fixed Result

Currently, XCCDF rule-results may be marked as "fixed" to indicate that a configuration that had previously resulted in a result of "fail" had been changed so that the Rule would now be regarded as passing. “Fixed” may be insufficiently granular for addressing failed configurations. Specifically, those viewing results may want to know more about why a particular configuration is being regarded as a “Pass” although it failed during evaluation. Adding more information about the nature of a “fixed” rule-result could lead to more helpful result documentation.

This discussion will focus on conveying more information about a “fixed” rule-result and the possible scoring impact of making changes to the result types. This corresponds to issue XCCDF-66 in NIST's issue tracker.

Issues

1. How should scores be affected by these new types? Currently a result can be equivalent to pass, fail, or "not count against the score". Would the new types be one of these, or would a fractional value (half a pass) be a concept to entertain.

Sample Proposal

Two new result types are defined: accepted and mitigated. A result of “accepted” is used when a failed configuration is acknowledged but will never be fixed. A result of “mitigated” is used where vulnerabilities posed by a failed configuration have been addressed through other means without necessarily bringing the configuration which caused the original failure into compliance with policy. While it is unlikely that a checking engine would return either of these results directly, they could appear in the "override" block of a rule-result to indicate an authority's acknowledgement of the issue.

When scoring, a Rule that has a result of "mitigated" is treated as if it had a result of Pass. A Rule that has a result of "accepted" does not contribute to the overall score, either positively or negatively. I.e., it has the same effective weight as an "informational" result.

```
<xsd:simpleType name="resultEnumType">
      <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:enumeration value="mitigated"/>
      <xsd:enumeration value="accepted"/>
</xsd:simpleType>
```

Allow Arbitrary External XML

Currently, XCCDF allows external XML structures of a different namespace to be used only within the context of a few objects. Specifically, the metadata fields in Benchmarks, Profiles, and Items can contain any type of content. It has been claimed that this remains overly restrictive and that foreign namespace inclusion be allowed anywhere within an XCCDF document. A namespace identifier could be used to distinguish foreign content from content based on the XCCDF schema and tools could use this to skip any foreign content that they did not know how to process. This corresponds to issue XCCDF-64 in NIST's issue tracker.

Issues

1. Currently, metadata fields are used to encapsulate externally namespaced structures. What need that would be met by allowing arbitrary XML in an XCCDF document would not be met by the current metadata fields?
2. Should limits be put on the use of external structures? For example, the metadata fields prohibit content that changes the standard processing of an XCCDF document. This was done to ensure that the openness of the metadata field did not introduce variability between XCCDF processing tools that were nominally performing the same task. Should similar restrictions be placed on external structures inserted in an XCCDF document?
3. Should some limits be placed on where external structures can appear? For example, should we only allow external elements to appear within elements that themselves can have child elements?

Sample Proposal

The schema is modified to include an "any" element in all complex types that can have child elements. These include:

checkType	noticeType	ruleResultType
complexCheckType	overrideType	selChoicesType
complexValueType	profileNoteType	signatureType
dc-statusType	profileRefineRuleType	targetFactsType
fixType	profileRefineValueType	testResultType
htmlTextType	profileSelectType	textWithSubType
htmlTextWithSubType	profileType	
itemType	referenceType	

To all of these types an any element will be added allowing use of another namespace. While this does not allow arbitrary XML to appear anywhere (since the external content will usually appear after the XML content) it does allow external content to appear in virtually every structure that can hold child elements. A simple example of the changes to one such type is shown below:

```
<xsd:complexType name="checkType">
    <xsd:sequence>
      <xsd:element name="check-import" type="cdf:checkImportType"
        minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="check-export" type="cdf:checkExportType"
        minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="check-content-ref" minOccurs="0"
```

```
        maxOccurs="unbounded" type="cdf:checkContentRefType"/>
      <xsd:element name="check-content" minOccurs="0" maxOccurs="1"
        type="cdf:checkContentType"/>
      <xsd:any namespace="##other" processContents="skip" minOccurs="0"
            maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
```

Allow Open Document Specification formatting

Currently XCCDF can embed HTML in the following fields;

Benchmark: description	Item: description	Rule::Item profile-note
Benchmark: notice	Item: warning	Rule::Item: fixtext
Benchmark: front-matter	Group::Item: rationale	Profile: description
Benchmark: rear-matter	Rule::Item: rationale	

While the current ability to embed HTML is useful, some community members have claimed that it does not provide authors with enough flexibility in terms of formatting and structuring documents. It has been proposed that XCCDF support including Open Document Specification Format (ODF) content as an alternative to provide authors the additional flexibility.

Like HTML, ODF is vendor neutral, non-proprietary, and platform independent, and provides an extensive range of formatting structures. ODF, however, has a more extensive library of formatting capabilities and, because it is used natively by a number of tools, allows content created in other tools to be expressed in ODF without significant changes to the document's appearance. For example, Microsoft Word can export to ODF with minimal loss of formatting in most cases. Allowing XCCDF to embed ODF would allow content to be directly pasted in from applications, such as Word, while preserving the source document formatting. This corresponds to issue XCCDF-65 in NIST's issue tracker.

Issues

1. Allowing ODF also imposes a burden on software developers who will need to support multiple formats for XCCDF fields. Are these trade-offs worth it?
2. Should new fields be created to support ODF or should we expand the capabilities of existing fields?
3. Should any conventions be established regarding the use of these new capabilities? For example, one might wish to disallow macros, scripts, and Dynamic Data Exchange (DDE) connections, all of which are supported by ODF. (This might be desirable from a security standpoint as well as reducing the burden on implementers.)

Sample Proposal

The htmlTextType and htmlTextWithSubType, which are the two existing types that allow HTML content, will be expanded to allow ODF in addition to their regular content. As such, HTML is still permitted, but ODF is now an additional option.

```
<xsd:complexType name="htmlTextType" mixed="true">
    ...
  <xsd:sequence>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:any namespace="http://www.w3.org/1999/xhtml" minOccurs="0"
               maxOccurs="unbounded" processContents="skip"/>
      <xsd:any namespace="http://docs.oasis-open.org/office/v1.1/O"
               processContents="skip"/>
    </xsd:sequence>
    </xsd:choice>
</xsd:complexType>
```

```
<xsd:complexType name="htmlTextWithSubType" mixed="true">
      ...
    <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:any namespace="http://docs.oasis-open.org/office/v1.1/O"
            processContents="skip"/>
    </xsd:choice>
 </xsd:complexType>
```

Merging Value Capabilities into Rules

At the 2010 IT Security Automation Conference in Baltimore, a suggestion was made that the current structural separation between Values and Rules in XCCDF documents makes the documents more difficult to author and edit while adding little utility in practice. The original purpose of the separation of these two structures was to allow many-to-many relationships between Rules and Values. In a recent poll, however, virtually no content producers could identify cases where a Value was referenced by more than one Rule. Unless this behavior is likely to change, we are left with little practical need to keep Rules and Values separate. This corresponds to issue XCCDF-62 in NIST's issue tracker.

While the elimination of Value structures would require a major change to XCCDF (breaking backward compatibility with existing XCCDF content), new structures could be added to Rules that mirror the behavior of Values in a backward-compatible way. Values can currently be scoped globally (the entire Benchmark document) or within Groups, so this change would simply to refine the scope of Values one more level down to individual Rules. Indeed, even if there is a sense that at some point in the future globally scoped Values might still be necessary, there might still be interest in creating Rule-specific Values for the many cases where they can be used.

Issues

1. This change adds no new functionality to XCCDF – its benefit is entirely for authors and editors who are manually reading XCCDF content. Editing tools (such as the Recommendation Tracker and Benchmark Editor, as well as editors published by multiple vendors) can abstract away the underlying problem. That said, many authors use basic XML editing tools and will continue to do so. Given this, is the benefit worth the change?
2. How should Profile tailoring be handled if Values and Rules are merged? Currently, Rules use selectors to select individual checks via refine-rule statements in Profiles. Values use selectors to control multiple fields via the refine-value Profile elements. When a refine-value statement gives a selector, every selectable field in a Value gets affected even if the selector id doesn't match. If no fields of a given type have a matching selector id, no fields of that type would be used when the Value was actually processed. Given this behavior, should a refine-rule statement affect the check structures and the new Value structures equally, or should a distinction be made? How should cluster-ids be made to behave in this case? See the discussion of "Address Ambiguity in cluster-id Use" for more on the use of these selectors.
3. Does it make more sense to associate Values with Rules or with individual checks within a Rule? This becomes significant when considering how the check-export structure, which is associated with individual checks, gets used. If Values are associated with Rules, there is still a need for the check-export to reference the (local) Value because different checks might use different sets of Values. On the other hand, if Values are associated with checks, those Value structures could simply identify the checking system variable to which they were exported and remove a level of indirection. The down side to the latter approach is that, if multiple checks used identical Value structures, these structures would need to be duplicated in each check.

Sample Proposal

Values can now appear within Rule elements. These Value structures are identical in form and function to Values that appear within Groups and Benchmarks. Values within a Rule can only be referenced (via a check-export statement) by that Rule.

Regarding Profile tailoring, refine-rule and refine-value would be strictly compartmentalized in their behaviors. Specifically, the selector attribute a refine-rule statement would only affect selectable elements of a Rule (namely the check field). A Rule's contained Values would not be affected by selector values in refine-rule structures. This would be the case even if the Value had a cluster-id value that was matched in the refine-rule's idref attribute. Correspondingly, refine-value statements would only affect Values (regardless of location) and would have no affect on fields in encapsulating Rules. This behavior is an extension of current behavior, where refine-rule statements can affect Groups without affecting the encapsulated Values and Rules.

The revised XML schema appears below:

```
<xsd:complexType name="ruleType">
    ...
  <xsd:complexContent>
    <xsd:extension base="cdf:selectableItemType">
      <xsd:sequence>
          ...
        <xsd:element ref="cdf:Value" minOccurs="0"
          maxOccurs="unbounded"/>
        <xsd:element name="signature" type="cdf:signatureType"
          minOccurs="0" maxOccurs="1"/>
      </xsd:sequence>
    ...
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

Local vs. Remote Imports

Currently the import statements in the XCCDF schema assume the six imported schemas are local and in the same directory as the XCCDF schema. Some have suggested that this be changed so that these imports are from canonical, remote locations. This corresponds to issue XCCDF-46 in NIST's issue tracker.

Prior discussions on this topic resulted in deadlock within the community. It is clear that either solution can be made to work in a technical sense. It is also clear that either solution leads to its own set of challenges for tool users and developers. As such, "consensus" on this topic will need to involve a solution that everyone considers workable, if not necessarily ideal.

Advantages of local references:

- Tools can validate content without network connectivity. While unconnected validation can be supported through special mechanisms when using remote references, doing so obviates the advantages of remote references, at least in this particular case.
- Users and tools can have direct control over which imported schemas are used to validate content without modifying the XCCDF schema file.

Advantages of remote reference:

- Pointing to a canonical location means that tools can always be directed to the latest version of each imported schema.
- Using remote references prevents "branching" of the standards by pointing to canonical locations. If local files are used, different schema bundles could have different versions of each referenced schema.
- It was noted that local files can become lost or corrupted (in addition to becoming obsolete) but having canonical remote references provides a location that is always to a present and uncorrupted source.

With these trade-offs in mind, it may be useful to consider the following questions:

1) Who has the greatest need to dictate the specific copy of referenced schemas used when using XCCDF content:
    a. The XCCDF specification and schema?
    b. Content producers?
    c. Tool users?
    d. Tools?
    e. The authorities responsible for the referenced schemas?
2) Who should be responsible for ensuring that, by default, the correct referenced schemas are used?
    a. The XCCDF specification and schema?
    b. Content producers?
    c. Tool users?
    d. Tools?
    e. The authorities responsible for the referenced schemas?
3) Of the advantages listed above, which one provides the greatest benefit to users? To tool developers? To content producers?

Given the challenges posed by this topic in the past, it would be very useful if participants could provide hard examples, either at the discussion or ahead of time via the mailing list, of scenarios where one solution or the other is particularly advantageous.

Sample Proposal

Self-explanatory

Linking Results to Source Checks

The community has repeatedly asserted that there need to be methods to associate a given rule-result with the source check that produced it. In particular, community members have argued that, should the source check change between XCCDF assessments, there should be some way to at-least detect the difference in the rule-results, and at-best trace those results to the specific instances of the check that gave rise to the result. One example where this becomes important can be seen when a source OVAL Definition (or an OVAL structure that the source Definition points to) gets revised between two XCCDF assessments. The result would be that, despite the fact that both assessments point to the same file and name of OVAL Definition, the given Rule would be evaluated against different criteria, possibly with a different result. This corresponds to issue XCCDF-19 in NIST's issue tracker.

Prior solutions have been stymied by the fact that a Rule's check instruction is not necessarily an atomic unit, but frequently the composition of several structures that can come from multiple files. Changes to a check can be functional (e.g., changing the value against which a setting is compared) or non-functional (e.g., changing a description or even the whitespace of the XML), and discerning one from the other so as not to give false positives when detecting changes is very difficult. Finally, not all tools even use the same source formats. For example, some tools pull out individual checking instructions from a given file and storing them in an internal format. Doing this means that indicators based on the source file are no longer applicable.

Given this, it is perhaps beneficial to consider a solution to a simpler problem: When a Rule is evaluated there might be multiple potential checks to use. Selectors can narrow the list, but even then there can be multiple effective check structures, each using a different checking system. Furthermore, a single check may have multiple check-content-ref statements or even a check-content statement from which to select the actual checking procedure to employ. Multiple factors can affect which of a given list of checks or check-content-ref statements actually get used, including ephemeral factors such as Internet connectivity and server availability. This can produce a challenging audit situation where the utilized source of a check's content can vary between runs due to unpredictable influences. A great deal of clarity into the workings of an assessment could be gained if one could track which check and check-content/check-content-ref was used for each Rule.

This does not address the case where the content in a particular target file changes between runs, such as if a file is updated. However, this latter scenario could be minimized by good document handling procedures and is also likely to be a less common occurrence in most environments.

Issues

1. Is noting the check and check-content-ref utilized for each rule-result sufficient for auditing or is a higher degree of assurance (e.g., signatures) required?
2. Should this information be recorded in a rule-result's check structure, or should new structures be utilized? See the discussion of "Conventions for use of Check Fields in rule-results" for discussions on the utilization of these fields.

Sample Proposal

A new structure is created for recording information about the source of a rule-result. Specifically, a new optional element named "source" is added. The source element can contain

either a single check or a single complex-check. In either case, the purpose is to use the check element's structures to note the specific check by mirroring its fields, as well as the appropriate check-content-ref or check-content structures that were utilized. In the case of a check-content-ref, this would be a duplicate of the check-content-ref statement in the corresponding Rule. In the case of a check-content, because a Rule can have at most one check-content, the body of the check-content in the rule-result can remain empty and simply serve as an indicator that the check-content body was utilized in the assessment.

```
<xsd:complexType name="ruleResultType">
        ...
      <xsd:sequence>
        ...
        <xsd:element name="fix" type="cdf:fixType" minOccurs="0"
          maxOccurs="unbounded"/>
        <xsd:element name="source" minOccurs="0" maxOccurs="1">
          <xsd:complexType>
            <xsd:choice minOccurs="1" maxOccurs="1">
              <xsd:element name="check" type="cdf:checkType"
                minOccurs="1" maxOccurs="1"/>
              <xsd:element name="complex-check" minOccurs="1"
                type="cdf:complexCheckType" maxOccurs="1"/>
            </xsd:choice>
          </xsd:complexType>
        </xsd:element>
        <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:sequence>
        ...
  </xsd:complexType>
```

Algorithmic to Functional Descriptions

It has been suggested that the current algorithmic depictions of XCCDF procedures are open to some degree of interpretation and that, to better codify the expected behavior, they should be replaced with functional specifications. The important distinction between the two methods is that, while an algorithmic description attempts to explain how to process the document, a functional specification would focus on precise descriptions of the state before and after a given step. Any algorithm that led to the correct final state would be acceptable, but implementers would be required to follow the given states explicitly. This corresponds to issue XCCDF-61 in NIST's issue tracker.

The full conversion of the XCCDF document to the use of functional descriptions would be a fairly large undertaking. As such, instead of providing a complete outline of all the changes we will simply provide a few examples of how this conversion might look and the community can evaluate the proposal based on these before a full conversion occurs.

Sample Change #1: 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. The processing type is Tailoring, and the selected property is false. 2. The processing type is Document Generation, and the hidden property is true. 3. The processing type is Compliance Checking, and the selected property is false. 4. 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.

This would change to:

Sub-Step	End State
Item.Process	If the item has a require statement where none of the indicated Items are selected, the current Item becomes deselected If the item has a conflicts statement where the indicated Item is selected, the current Item becomes deselected The Item.Select step is initiated.
Item.Select	If any of the following conditions holds, cease processing of this Item. 1. The processing type is Tailoring, and the selected property is false. 2. The processing type is Document Generation, and the hidden property is true. 3. The processing type is Compliance Checking, and the selected property is false.

	4. 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. Otherwise, the Item.Content step is initiated
Item.Content	If the Item is of type Group, the Group.Content step is run. If the Item is of type Rule, the Rule.Content step is run. If the Item is of type Value, the Value.Content step is run.
Group.Content	All Group properties relevant to the processing type are processed. Group.Content.Items is initiated.
Group.Content.Items	Each Item within the Group undergoes Item Processing in the order in which they appear within the Group.
Rule.Content	All Rule properties relevant to the processing type are processed.
Value.Content	All Value properties relevant to the processing type are processed.

Sample Change #2: Rule Check Processing

During the Rule.Content sub-step of the Item Processing Algorithm, the properties of a given Rule are processed. This includes processing of the Rule's check structure(s). This section describes how these structures are processed. Note that, if a Rule contains a complex-check, then the XCCDF interpreter must process this complex-check, ignoring any regular checks that are also contained by the Rule. However, within a given complex-check, processing of component checks follows the same procedures described below.

First, the appropriate set of check structures within the given Rule must be identified. This is done through the selector attribute of the check property. A Profile or manual tailoring action can associate a selector with individual Rules. If a selector name has been associated with a Rule then all of the Rule's checks whose selector attribute equals that selector name are added to the working set of check structures. If no selector name has been associated with the Rule or, if after this step, the working set remains empty, then all checks with no selector attribute or with a selector attribute that is the empty string are added to the working set of check structures. If the working set of check structures is still empty after this, then no check is processed for the Rule and the Rule will get a result of UNCHECKED. Note that this could happen even when the Rule defines check structures.

Once a working set of check structures has been created, the XCCDF interpreter must select the check to process. At this stage, all check structures in the working set must have different values in their system attributes. An XCCDF interpreter should select the check structure that identifies a check system it supports. If there are multiple checks with supported system attributes, XCCDF interpreters may use their own logic to determine which check to utilize. If the XCCDF interpreter does not support any of the identified checking systems, then no check is processed for this Rule and the Rule will get a result of UNCHECKED.

Now that a single check structure has been selected by the XCCDF interpreter, the interpreter begins processing the check structure itself. First, the check-export statements are processed. For each XCCDF Value named in a check-export statement, determine the appropriate value of this Value. The Value's value might have been set explicitly via a set-value selector in a Profile or from manual tailoring, or it might be identified via a selector name via a refine-value selector in

a Profile or also from manual tailoring. The XCCDF interpreter must identify a value for each exported Value.

Finally, a check's check-content-ref and/or check-content structures must be processed. These must be processed in the order in which they appear in the XML, so processing all check-content-ref structures will precede processing of any check-content structure. The XCCDF interpreter must inspect each check-content-ref structure, in order, and determine if content can be retrieved from the given location. If content can be retrieved from the location specified by a given check-content-ref then that content becomes associated with the check and no further check-content-ref or check-content statements are consulted. Only if none of the check-content-ref statements can be resolved to content (or the check contains no check-content-ref statements to begin with) would the body of a check-content statement become the content associated with a given check. If none of a check's check-content-ref statements can be resolved to content and the check has no check-content statement, then no check is associated with this Rule and the Rule will get a result of UNCHECKED. Note that the XCCDF interpreter must make no assertions as to the validity of the content associated with a check. Content that is successfully retrieved from a source might be invalid for a giving checking system, but it is the checking system interpreter's job to catch and report the content error back to the XCCDF interpreter. A check-content-ref would only be skipped if content could not be retrieved from the identified source (such as if a given file could not be found or in the presence of a network failure during a remote reference resolution). Retrieval of invalid content is not grounds for skipping a check-content-ref.

At no point should an XCCDF interpreter "backtrack" in the processing of these steps. For example, once a check with a preferred system is selected by the XCCDF interpreter the interpreter may not attempt to use a different check that uses a different system, even if none of the originally selected check's content can be resolved. Instead, the checking language interpreter's response (probably an error of some kind) must be mapped an appropriate XCCDF result (probably ERROR) and that becomes the result of this check.

This would change to:

Sub-Step	End State
Rule.Check.Selector	An "working set" consisting of all check elements whose selector attribute matches the active selector string for the Rule is created. If the working set is empty, this Rule's result is "unchecked" and Rule.Check processing stops for this Rule. Otherwise, the Rule.Check.System step is initiated.
Rule.Check.System	A single check from the working set is selected based on the value of its system attribute. (Specific criteria for this determination are at the discretion of implementers.) If this fails, this Rule's result is "unchecked" and Rule.Check processing stops for this Rule. Otherwise, the Rule.Check.Export step is initiated.
Rule.Check.Contents	The Check.Content.Verify step is applied to each check-content-ref and check-content statement in the order in which they appear.
Check.Content.Verify	If processing a check-content-ref statement, cease this step if the referenced content is unreachable. Otherwise, the Check.Content.Process step is initiated.

	If processing a check-content statement, the Check.Content.Process step is initiated.
Check.Content.Process	A set of checking instructions is identified. The result of this check will be the result of attempting to apply these checking instructions. (Note – the check instructions to not need to be applied at this time; tools may wait until all Rules have been processed and then apply all checking instructions simultaneously.) The Rule.Check.Complete step is initiated, terminating the Rule.Check.Contents loop.
Rule.Check.Complete	If Check.Content.Process was never reached, the result of this Rule is "unchecked".

Address Ambiguity in cluster-id Use

The XCCDF specification currently contains ambiguous statements with regards to cluster-id use. The XCCDF specification states the following about cluster-id use by a refine-rule: “The idef attribute must match the id attribute of a Value or a cluster-id of one or more Items in the Benchmark”. The XCCDF specification also states the following about cluster-id use by a refine-value: “The 'idref' attribute must match the id attribute of a Value or a cluster-id of one or more Items in the Benchmark”.

The issue is that it is ambiguous as to whether a refine-rule selector should tailor Values that share the given cluster-id, or if a refine-value should tailor Rules that share the given cluster-id. Given that both Rules and Values utilize the selector field that is affected by the refine-X statements, any selector-based tailoring could affect both types of structures equally, although the other fields of a refine-rule statement (tailoring weight, role, and severity) and of refine-value (operator) would be inapplicable to the other element. This corresponds to item XCCDF-68 in NIST's issue tracker

The two options are:

1. refine-value only affects Values and refine-rule only affects Rules and Groups
2. refine-value and refine-rule statements affect any Item equally and inappropriate properties (e.g., a weight property in refine-rule when applied to a Value) are ignored.

Issues

1. If refine-value and refine-rule will both apply to all Items, is there any need to have distinct structures for the two? Would it be better to have a refine-item to make coverage explicit?
2. On the other hand, refine-rule already does double duty, tailoring Rules and Groups. Should there be a separate refine-group for clarity?

Sample Proposal #1

Descriptions of refine-value and refine-rule and their usage will be clarified to make it explicit that the former only applies to Values and the latter only applies to Rules and Groups. XCCDF interpreters will be expected to ignore Items of the wrong type, even if the referenced id or cluster-id would point to such an Item.

Sample Proposal #2

A new refine-item type is created to (eventually) replace existing refine-rule and refine-value objects. This new refine-item type would address selector-based tailoring when applied to all Item type objects: Rules, Values, and Groups. Processing engines would need to determine the applicability of individual properties of a refine-item instruction upon dereferencing an idref to a particular type of Item. The refine-rule and refine-values selectors would be deprecated.

```
<xsd:complexType name="profileRefineItemType”>
    <xsd:sequence>
        <xsd:element name="remark" type="cdf:textType" minOccurs="0
            maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="idref" type="xsd:NCName" use="required"/>
```

```
    <xsd:attribute name="weight" type="cdf:weightType" use="optional"/>
    <xsd:attribute name="selector" type="xsd:string" use="optional"/>
    <xsd:attribute name="severity" type="cdf:severityEnumType" use="optional"/>
    <xsd:attribute name="role" type="cdf:roleEnumType" use="optional"/>
    <xsd:attribute name="operator" type="cdf:valueOperatorType" use="optional"/>
</xsd:complexType>
```

Address Inconsistency in the Default Scoring Model

According to the text on page 34, XCCDF results of notapplicable, notchecked, notselected, and informational "do not contribute to the Benchmark score". However, in the default scoring model (page 51), the steps only distinguish between results of "pass" and all other results. In order to be consistent with the text on page 34, the default scoring algorithm must be modified to explain how to handle these result conditions. This corresponds to issue XCCDF-67 in NIST's issue tracker.

The algorithm for the flat scoring model (page 52) provides an example of how these exceptional conditions should be addressed. The updated wording in Score.Rule is designed to remain consistent with the Score.Group.Recurse and Score.Weight steps in that it ensures the values of the score and count variables are defined in all cases.

Change Details:

DOCUMENTATION: Page 51 – Update default scoring algorithm

Sub-Step	Description
Score.Rule	If the node is a Rule: - if the test result is a member of the set {notapplicable, not checked, informational, notselected} then set its score and count to 0 - otherwise 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.

Inconsistent Removal of Extension Property

The sub-step Loading.Resolve.Items includes four steps, the last being "remove the extends property". The sub-step Loading.Resolve.Profiles only has three steps, leaving off the "remove the extends property". Given that the resolution of either object follows the same steps for the same reasons, the two sub-steps should be made to match each other. This corresponds to issue XCCDF-69 in NIST's issue tracker.

Change Details:

DOCUMENTATION: Page 37 – Loading.Resolve.Profiles

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) insert the necessary property sequences from the extended Profile into the appropriate locations in the extending Profile, (3) remove all but the last instance of duplicate properties, and (4) remove the extends property. 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 the extends properties of Profiles includes a loop, then Loading fails. Otherwise, go to Loading.Resolve.Abstract.

Normalize Formatting

There are formatting inconsistencies in the core structure descriptions. Specifically, the manners in which default values are given and the manner in which mandatory vs. optional properties are indicated vary within the property tables. Formatting has been normalized to improve readability.

Change Details:

DOCUMENTATION: Page 15 – *Benchmark*

Property	Type	Count	Description
id	identifier	1	Benchmark identifier~~, mandatory~~

DOCUMENTATION: Page 19 – *Item (base)*

Property	Type	Count	Description
id	identifier	1	Unique object identifier~~, mandatory~~
status	string+date	0-n	Status of the Item and date at which it attained that status~~, optional~~
cluster-id	identifier	0-1	An identifier to be used from a Profile to refer to multiple Groups and Rules~~, optional~~
signature	special	0-1	Digital signature over this Item~~, optional~~

DOCUMENTATION: Page 20 – *Group : : Item*

Property	Type	Count	Description
selected	boolean	1	If true, this Group is selected to be processed as part of the Benchmark when it is applied to a target system; an unselected Group is not processed, and none of its contents are processed either (i.e., all descendants of an unselected group are implicitly unselected).~~Default is true~~ (default: true). Can be overridden by a Profile
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~~

DOCUMENTATION: Pages 21-23 – *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 checked and does not contribute to scoring).~~Default is true~~ (default: true). Can be overridden by a Profile
multiple	boolean	0-1	Whether this rule should be multiply instantiated. If false, then Benchmark tools should avoid multiply instantiating this Rule. If true, this Rule should produce a result for every instance of an assessed target. ~~The default is false.~~ (default: false).

multi-check	boolean	0-1	If true, this Rule should produce a result for every referenced component check. If false, the check should produce only a single result regardless of the number of component checks.~~The default is false.~~ (default: false).
role	string	0-1	Rule's role in scoring and reporting; one of the following: "full", "unscored", "unchecked".~~Default is "full".~~ (default: full). Can be overridden by a Profile. The use of this field is now deprecated.
severity	string	0-1	Severity level code, to be used for metrics and tracking. One of the following: "unknown", "info", "low", "medium", "high". ~~Default is "unknown".~~ (default: unknown). Can be overridden by a Profile
weight	float	0-1	The relative scoring weight of this Rule, for computing a compliance score. ~~Default is 1.0~~ (default: 1.0). Can be overridden by a Profile

DOCUMENTATION: Page 23

The ~~'multiple'~~ multiple property is applicable in cases where there are many instances of a target. For example, a Rule may provide a recommendation about the configuration of application user accounts, but an application may have many user accounts. By setting ~~'multiple'~~ multiple to true, the Rule's author is directing that separate instances of the target to which the Rule can apply should be tested separately and the results recorded separately. By setting ~~'multiple'~~ multiple to false, the author is directing that the test results of such instances be combined.

DOCUMENTATION: Page 26 – *Value : : Item*

Property	Type	Count	Description
default/ complex-default	string + id/*special*	0-n	Default value of this Value object~~, optional~~
type	string	0-1	The data type of the Value: "string", "number", or "boolean" (default: ~~"~~string~~"~~)
interactive	boolean	0-1	Tailoring for this Value should also be performed during Benchmark application~~, optional~~ (default~~: is~~ false)

DOCUMENTATION: Page 29 – *Profile*

Property	Type	Count	Description
signature	*special*	0-1	Digital signature over this Profile~~, optional~~

DOCUMENTATION: Page 30– *TestResult*

Property	Type	Count	Description
remark	string	0-n	A remark about the test, possibly supplied by the person administering the Benchmark

			checking run~~, optional~~

DOCUMENTATION: Page 33 – *TestResult/rule-result*

Property	Type	Count	Description
severity	string	0-1	The severity string code copied from the Rule~~;defaults to unknown.~~ (default: unknown).