Discussion Topics and Fix Proposals

Open Issues in XCCDF to be covered that the Security Automation Developer Day Workshop on February 23, 2010

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 will be brought before the broader community. The authors wish to emphasize this: listing of an issue as a "fix proposal" in this document is not meant to restrict discussion. It simply indicates that the authors' 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.

Issues are listed in order in which they appear in the proposed agenda for the day's events. This list looks extremely long, even given 8 hours to address it. However, 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

Using CVSS/CCSS in Scoring

Currently, XCCDF defines four scoring models: default (weighted scores normalized within each Group), flat (sum of weights), flat unweighted (a count of passing rules), and absolute (pass-fail). None of these models explicitly makes use of the scoring systems of CVSS or CCSS. One could manually assign weights to each rule equal to the appropriate CVSS or CCSS score, but given that XCCDF Rules already have a field for CVSS vectors, automating this step would be a reasonable expectation. It has been proposed that a more effective/efficient utilization of CVSS/CCSS scores be applied to the XCCDF scoring model. This issue corresponds to issue XCCDF-8 in JIRA.

Issues

1. Is a new scoring model likely to be utilized?
2. In most cases, a Benchmark would only be published with base scores and not temporal or environmental scores. However, the latter scores could be of interest to users. How would this information best be accounted for?
    a. Would this be part of the document tailoring step?
    b. Would we want profiles to provide default environmental information to represent certain standard types of enclaves?
    c. Would it be better to not handle such capabilities within the XCCDF language itself and instead allow implementers to support additional CVSS/CCSS scoring components (or not) as they wish?
3. How would CVSS scores best be used in algorithms? Do we just sum the metric value of failed Rules or is some weighting appropriate.

Sample Proposal

The impact-metric field already contains a CVSS vector string. (Currently it only allows a CVSS base vector, but the next issue discusses allowing addition types of information, such as CCSS vectors.) As such, there is no need to add a new field to the schema to accommodate the new scoring model.

The specification would be updated with two new scoring models. The first, called "flat metric". In this model the final score is the sum of the CVSS/CCSS scores, as provided in the impact-metric field, of all scored Rules (excludes notapplicable, notchecked, notselected, and informational) that do not return a result equivalent to "pass". The second, called "percentage metric" is one minus the value of the "flat metric" divided by the total CVSS/CCSS scores of all scored Rules (i.e., 1 – (flat-metric/total-scores)). As such, it represents the percentage of the total available CVSS/CCSS score that the assessment achieved.

If the impact-metric field is absent from a Rule in this scoring model, treat that rule as having a score of 6.0, which corresponds to the vector AV:N/AC:M/Au:S/C:P/I:P/A:P in both CVSS and CCSS. It should be noted, however, that using a default value in this calculation could be misleading and tool implementations may wish to warn users if the default value must be employed. Note further that this "default" is not the default value of the impact-metric field in the XML schema, but merely the default value used in calculations of the flat metric and percentage metric scoring models. As such, the given default should not be filled-in in other uses of impact metric, such as when converting a document to a human-readable format.

It is recommended that Benchmarks that are to be broadly published include only the base CVSS/CCSS vectors in order to maximize their applicability. The XCCDF specification does not dictate if or how temporal or environmental vectors be added to the impact-metric if they not already present - this detail is left up to tool vendors.

Using CVSS temporal and environmental vectors and CCSS vectors in the impact-metric field

Currently the impact-metric field of XCCDF Rules only allows base CVSS vectors. This suggests two issues. First, in practice, most Benchmark Rules correspond to configuration recommendations rather than checks for the presence of known vulnerabilities. CVSS metrics are not appropriate for scoring configuration items and the nascent CCSS metric (http://csrc.nist.gov/publications/drafts/nistir-7502/Draft-NISTIR-7502.pdf) should be used instead.

Secondly, the base vector in CVSS (and CCSS) represents only a third of the overall metric vector – both CVSS and CCSS have environmental and temporal vectors as well. While the base metric is the most general of the three portions of a CVSS/CCSS vector, many environments may wish to include tailored environmental and temporal information in the impact-metrics of rules. Both of these issues correspond to item XCCDF-52 in JIRA.

Issues

1. Assumedly, the restriction to just the base vector was taken in order to ensure that published Benchmarks had the widest possibly applicability (since environmental and temporal vectors are not universal). If the environmental and temporal vectors are added, we raise the possibility of people using impact-metric values that are not appropriate for their environment. Moreover, tools would now need to process the full CVSS/CCSS vectors rather than just the base vectors.
2. While the equations of CVSS and CCSS are currently identical, the temporal and environmental vectors and equations as well as the base vectors are not. As such, especially if the environmental and temporal vectors were included, tools would need to have separate computing structures for the two metrics.
3. CVSS is currently in version 2.0 while CCSS currently exists as a draft publication. Further revisions of either metric could result in changes to the vector and the scoring model. Should the schema include a field that indicates the metric and its version within the document or should the specification dictate the version of the two metrics?
4. Currently, impact-metric is of type xsd:string. Do we want to constrain the value of the impact-metric field so that it can only conform to the values and ordering in the vector specifications of the respective metrics?

Sample Proposal

In this proposal, the specification is updated to allow impact-metric to contain either CVSS or CCSS vectors. The specification continues to require that only the base vector be included in the field. Moreover, the schema is updated to require that the content of impact-metric matches the exact structures of the vectors as described in the CVSS or CCSS specifications. While organizations may wish to include temporal or environmental factors in the CVSS/CCSS information, such information will, by definition, only be relevant at a specific time and place. As such, tool vendors wishing to support temporal and environmental vectors may develop their own proprietary means for collecting and storing this information as broad dissemination of such information, and the need for a standardized way to encapsulate it, is unlikely to occur.

The XCCDF specification will require use of CVSS 2.x or CCSS 1.x. No flags will be added to the schema to indicate the specific metric or version.

XML

```
<xsd:element name="impact-metric" type="xsd:string"
        minOccurs="0" maxOccurs="1">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern
value="AV:[LAN]/AC:[HML]/Au:[MSN]/C:[NPC]/I:[NPC]/A:[NPC](/PL:[RUA]|(ND)/EM:[
AP])?"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

Update CPE Version

Currently XCCDF requires the use of CPE 2.0 for platform identification. However, the current version of CPE is 2.2 and, as such, this requirement is out of date. While the simplest solution is simply to require 2.2 instead of 2.0, this means that the release of CPE 2.3 or 3.0 will require additional changes to the XCCDF spec and schema. This proposal explores some alternative constructions. This corresponds to issue XCCDF-9 in JIRA.

Issues

1. How flexible do we want the specification to be?
    a. Do we want to require a specific version of a specific platform language (i.e. CPE 2.2) with possible support for a few deprecated formats? (This is akin to the current design.)
    b. Do we want permit a specific range of versions but still focus on one format (i.e. CPE 2.x)?
    c. Do we more flexible platform specification?

Sample Proposal #1 – Focus on CPE 2.x

Below is one possible solution that requires CPE but allows any version thereof within major version 2. (I.e. CPE 2.x)

Description

The specification will be updated to require use of CPE version 2.x. Other formats currently provided as deprecated formats will continue to be available as deprecated formats. Since minor revisions of CPE should be backwards compatible, we can have confidence that revisions of CPE 2.x will not deprecate existing XCCDF content. The only challenge could come in the form of a lack of forward compatibility where a tool is able to understand CPE 2.2 but becomes confused by later XCCDF documents written using CPE 2.3. This creates the added burden that tool authors must keep their XCCDF tools up to date with CPE. If they fall behind in CPE support, an document could be in a supported version of XCCDF (e.g 1.2), but because that document's CPE references have moved ahead of the tool's ability to support CPE, the tool might not be able to fully support the document.

XML

XCCDF imports the CPE language. Currently, while the CPE namespace only reflects the major version of CPE, the file referenced in the href includes major and minor information. In order to support the latest version of CPE, the CPE team would need to create a link that automatically pointed to the latest version of the CPE schema.

```
<xsd:import namespace="http://cpe.mitre.org/language/2.0"
    schemaLocation="cpe-language_2.latest.xsd">
    ...
</xsd:import>
```

Sample Proposal #2 – Support named platform identifiers

Below is a possible solution that allows benchmark authors to explicitly identify the platform identifier and version they are using, thus permanently freeing XCCDF from falling out of date in its platform identification.

Description

We will create a new platform identification element titled target-platform. This element would be made available wherever the platform element appears now. (E.g. Benchmark, Profiles, Rules, etc.) For backward compatibility, all other platform identification structures would remain in order to preserve backward compatibility, but would be targeted for removal whenever XCCDF undergoes a major revision.

The target-platform element would have a @system attribute that would identify the type of platform identifier. For example, to use CPE 2.x, the @system attribute would be set to http://cpe.mitre.org/language/2.0. Other platform identification structures would use other @system strings. The body of the target-platform element would be "any" XML and used to encapsulate the identifier that used the given system. To handle the case where a platform identifier might be expressed as a literal instead of in an XML structure, the target-platform element would also have an optional @value attribute that could hold a string. It would be an error for both the body of the element and the @value attribute to contain data – this would require a Schematron rule to enforce.

This design is similar to conventions in the check-content-ref elements and would effectively allow XCCDF to become agnostic with regards to revisions to platform specification changes.

XML

A new targetPlatformType would need to be created. This type would be instantiated wherever the platform element appears now.

```
<xsd:complexType name="targetPlatformType">
  <xsd:sequence>
    <xsd:any minOccurs="0" maxOccurs="unbounded" processContents="skip"/>
  </xsd:sequence>
  <xsd:attribute name="system" type="xsd:anyURI" use="required"/>
  <xsd:attribute name="value" type="xsd:string" use="optional"/>
</xsd:complexType>
```

Explicit Mapping of Check Results to XCCDF Results

Currently, NIST 800-126 describes a mapping from OVAL check results to XCCDF results. However, this mapping may not always encapsulate the desired behavior of a benchmark author. Moreover, XCCDF documents could reference checking systems other than OVAL for which a canonical mapping has not been defined, potentially leading to different behaviors across tools even if they both support the given checking language. It has been suggested that XCCDF checks add capabilities that would allow authors to control the mapping of checking system results to XCCDF results. This corresponds to issue XCCDF-4 in JIRA

Issues:

1. There are 9 different XCCDF results. Do we need the ability to explicitly map results to each and any of those result types, or is it only necessary to map to pass or fail?
2. Most checking languages will at least have the concept of passing or failing. As such, for most cases, users may really only need to control a mapping from the checking system's pass-fail to XCCDF's pass-fail. An excellent example of this is seen in NIST IR 800-126 (the SCAP specification) where if XCCDF references an OVAL compliance definition, a return value of "true" results in an XCCDF value of pass, but a "true" result from a vulnerability definition results in an XCCDF vale of fail. Would it be sufficient to provide a means to negate the default mapping of check result to XCCDF result.
3. Checking languages may have many result types. Do we want shorthand methods that allow many checking language results to map to a single XCCDF result?

Sample Proposal

A new element will be added to the checkType. This optional element, called check-result-map, will have a required attribute called "result" that must contain an XCCDF result. The body of this element is a string and would correspond to a return result from the checking system. Any number of check-result-map elements may appear with the only restriction being that no two check-result-map elements may have the same body value (enforced with an XML "unique" descriptor in Rule/check. If the result returned by the checking system matches the body of one of the check-result-map elements, the corresponding XCCDF result becomes the result of that Rule. If there is no match, a default mapping is used, starting with the mapping provided in the SCAP specification (NIST IR 800-126) followed by a mapping defined in the specification of the checking language itself. If no such default mapping is available, an XCCDF result of Unknown should be given to the Rule. Check-result-map elements would be inherited under the "replace" processing model. This means that any check-result-map element in an extending Rule would remove all check-result-map elements in the parent.

```
<xsd:complexType name="checkType">
    ...
  <xsd:sequence>
    <xsd:element name="check-result-map" minOccurs="0" maxOccurs="unbounded">
      <xsd:complexType>
        <xsd:simpleContent>
          <xsd:extension base="xsd:string">
            <xsd:attribute name="result" type="cdf:resultEnumType"
                use="required"/>
          </xsd:extension>
```

```
            </xsd:simpleContent>
          </xsd:complexType>
        </xsd:element>
        ...
      </xsd:sequence>
        ...
    </xsd:complexType>

<xsd:complexType name="ruleType">
        ...
      <xsd:complexContent>
        <xsd:extension base="cdf:selectableItemType">
          <xsd:sequence>
              ...
            <xsd:choice>
              <xsd:element name="check" type="cdf:checkType"
                      minOccurs="0" maxOccurs="unbounded">
                <xsd:unique name="resultMapUnique">
                  <xsd:selector xpath="./check-result-map"/>
                  <xsd:field xpath="."/>
                </xsd:unique>
              </xsd:element>
              <xsd:element name="complex-check" minOccurs="0"
                    type="cdf:complexCheckType" maxOccurs="1"/>
            </xsd:choice>
              ..
          </xsd:sequence>
              ...
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
```

A similar uniqueness constraint would be put in the check element of complexCheckType. (Given this modification, it might make more sense to make check a global element and use references to a common element in both locations.)

Segregated or Mixed Extensions to Value

In a previous discussion, the community suggested expanding the structures of XCCDF Values to allow lists and externally defined types. A proposal has been circulated that creates new structures in Value elements that parallel the value, default, and choice elements but hold either lists or external types. Similar structures were added to Profiles to allow tailoring of lists or external types. These structures exist in addition to the original value, default, etc. elements, thus preserving backwards compatibility.

One specific question has been raised in reference to this proposal, however. This is the question of whether a single Value should be restricted to only the original singleton types or only the new lists or external types, or if a single Value should be allowed to be tailored to either type without restriction. Arguments for separating center on a concern that forcing tools to be able to switch between the two types would be burdensome to implementers. Arguments for allowing mixing of singletons and new, complex types argue that tools will need to handle all types anyway and that authors may wish to mix basic and external types as options in the same Value. (For example, '1', '2', '3' using singletons, and 'infinity' using an external type.) This corresponds to issue XCCDF-5 in JIRA.

Proposals

A Value that allowed authors to mix simple and complex values might appear as follows:

```
<xsd:sequence>
  <xsd:choice minOccurs="1" maxOccurs="unbounded">
    <xsd:element name="value" type="cdf:selStringType" minOccurs="1"
      maxOccurs="1"/>
    <xsd:element name="complex-value" type="cdf:selComplexValueType"
      minOccurs="1" maxOccurs="1"/>
  </xsd:choice>
  <xsd:choice minOccurs="0" maxOccurs="unbounded">
    <xsd:element name="default" type="cdf:selStringType" minOccurs="1"
      maxOccurs="1"/>
    <xsd:element name="complex-default" type="cdf:selComplexValueType"
      minOccurs="1" maxOccurs="1"/>
  </xsd:choice>
    ...
```

A Value that restricted authors to only simple or only complex values might appear as follows:

```
<xsd:sequence>
  <xsd:choice minOccurs="1" maxOccurs="1">
    <xsd:sequence>
      <xsd:element name="value" type="cdf:selStringType" minOccurs="1"
        maxOccurs="unbounded"/>
      <xsd:element name="default" type="cdf:selStringType" minOccurs="0"
        maxOccurs="unbounded"/>
      <xsd:group ref="cdf:valueConstraints"/>
      <xsd:element name="choices" type="cdf:selChoicesType"
        minOccurs="0" maxOccurs="unbounded"/>
```

```
        </xsd:sequence>
        <xsd:sequence>
          <xsd:element name="complex-value" type="cdf:selComplexValueType"
            minOccurs="1" maxOccurs="unbounded "/>
          <xsd:element name="complex-default" type="cdf:selComplexValueType"
            minOccurs="0" maxOccurs="unbounded "/>
          <xsd:group ref="cdf:valueConstraints"/>
          <xsd:element name="complex-choices" type="cdf:selComplexChoicesType"
              minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:choice>
        ...
```

In both cases, selComplexValueType is defined as follows:

```
  <xsd:complexType name="complexValueType">
    <xsd:annotation>
      <xsd:documentation> The type that a Value may encapsulate. This can be a list or some
          external XML representation of a data structure.</xsd:documentation>
    </xsd:annotation>
    <xsd:choice>
      <xsd:element name="item" minOccurs="1" maxOccurs="unbounded"
                type="xsd:string"/> <!-- For lists -->
      <xsd:element name="external-type"> <!-- For external structures -->
        <xsd:complexType>
          <xsd:sequence>
            <xsd:any minOccurs="1" maxOccurs="unbounded" processContents="strict"
                namespace="##other"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:choice>
  </xsd:complexType>

  <xsd:complexType name="selComplexValueType">
    <xsd:annotation>
      <xsd:documentation> Use a selector to identify a complex value for later reference.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="cdf:complexValueType">
        <xsd:attribute name="selector" default="" type="xsd:string" use="optional"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
```

Content Categorization

Currently Rules and Groups can be organized by placing them within Groups. However, this means that any given Rule/Group can only belong to a single category. Sometimes, authors would like to be able to track many-to-many relationships. For example, mapping Rules to high-level security controls often involves situation where a single control maps to many Rules and a single Rule maps to many controls. Currently, the "requires" element has been co-opted for tracking this type of relationship, but others have pointed out that this is an inappropriate use of this field. This corresponds to issue XCCDF-12 in JIRA.

Issues:

1) There is some conceptual overlap between categorization and with cluster-ids. However, cluster-ids only allow Items one additional categorization (each Item can only have one cluster-id value) and, like Groups, there are uses of cluster-ids beyond simple categorization (e.g. for tailoring) and authors may need to utilize cluster-ids in that capacity rather than for annotation. This said, if cluster-ids were changed to support lists of values (a minor change) they could support unlimited categorization with the added capability of supporting tailoring actions. Is this preferable to the creation of a dedicated category element that does not have any link to tailoring actions?

Sample Proposal #1

This proposal is to add a new element, called "category", to Items. This optional element could be used to assign any number of categories to an Item. Each instance of "category" would represent a separate categorization, so, for example, membership in three categories would require three instances of the category element. Category names could be any string, including multiple-word phrases. The new category element would be inherited under the "append" processing model. The category field would be used by tools for searching or otherwise organizing Benchmark Rules, but plays no role in tailoring or assessment.

```
<xsd:complexType name="itemType" abstract="1">
            ...
  <xsd:sequence>
            ...
    <xsd:element name="metadata" type="cdf:metadataType"
            minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="category" type="xsd:string"
            minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
            ...
</xsd:complexType>
```

Sample Proposal #2

This proposal expands the use of the cluster-id attribute already in Items to allow it to hold a list of NCName values rather than a single value. This not only allows for many-to-many groupings, but also allows tailoring actions to affect Items through their group associations. Because cluster-ids are not inherited, extending Items would not inherit their parent's group associations.

It should be noted that the order in which a Profile applies tailoring actions becomes more important in the scenario because of the possibility of a single Item belonging to multiple clusters and, as a result, being affected by multiple selectors. This is already a possibility in the current version of XCCDF: a Rule could be affected by a selector that used its cluster-id and then modified again by a selector that names the Rule explicitly. However, the expansion of the number of handles by which an Item could be tailored will make such repeated tailorings more likely and tools will need to be able to support this.

```
<xsd:attribute name="cluster-id" use="optional">
  <xsd:simpleType>
    <xsd:list itemType="xsd:NCName"/>
  </xsd:simpleType>
</xsd:attribute>
```

Open the metadata field to additional types of metadata

Currently the XCCDF schema limits the contents of the metadata field to Dublin Core and NIST Checklist (SCCF) formats. There have been requests to use additional metadata information, such as included with STIGs. This corresponds to issue XCCDF-40 in JIRA.

Issues

1. Currently, the XCCDF schema can explicitly dictate the format of the contents of the metadata elements through references to the Dublin Core and SCCF schemas, respectively. If strict processing is no longer required, this not only allows for the introduction of data that follows other formats, but also opens the possibility for data that fails to follow any recognized format or which incorrectly follows a known schema. Tools will need to be able to handle such instances.
2. Alternatively, if we allow any schema but force strict processing, this prevents organizations from inserting ad-hoc metadata. While ad-hoc metadata would not be understandable outside the given organization, there is no reason such lack of understanding would be a significant problem. Moreover, if strict formatting is required, all recipients would need copies of all the schemas the authors used to create their metadata.

Sample Proposal

In this proposal we remove the requirement to use only the Dublin Core or SCCF schemas and instead allow any namespace other than the XCCDF target namespace. Lax processing is required so that authors can use schemas to ensure correct content formatting, but also to allow ad-hoc or private metadata constructs.

```
<xsd:complexType name="metadataType">
  <xsd:annotation>
    ...
  </xsd:annotation>
  <xsd:sequence>
    <xsd:choice minOccurs="1" maxOccurs="1">
      <xsd:any namespace="http://purl.org/dc/elements/1.1/"
               minOccurs="1" maxOccurs="unbounded"/>
      <xsd:any namespace="http://checklists.nist.gov/sccf/0.1"
               processContents="skip"
               minOccurs="1" maxOccurs="unbounded"/>
    </xsd:choice>
    <xsd:any minOccurs="1" maxOccurs="unbounded" processContents="lax"
             namespace="##other"/>
  </xsd:sequence>
</xsd:complexType>
```

Clarify check-import Behavior

XCCDF 1.1.4 added the check-import element "to allow a benchmark author to specify values to retrieve from the checking system." However, apart from some comments about the raw XML structure, the only reference to check-import comes in its dictionary item, and even this is a very limited description. As such, multiple members of the community have felt this feature to be significantly under-defined. This discussion will focus on check-import: what capabilities it needs to support and how it can be made to support these capabilities. This corresponds to issue XCCDF-6 in JIRA.

Issues

1. What should be the purpose of check-import? Is it trying to populate a Value that will be used in a check-export statement or is seeking just to document a finding in the rule-result? Is there an actual need for importing?
2. If we are seeking to import data for subsequent export in a check-export, how would we get around issues of Value dependency? How would we support batch-checking systems?
    a. Option 1: Limit the scope of the import. If an import is only in scope in its containing Rule then dependency is easy to control.
    b. Option 2: Allow global scope of imports but force all Rules that import into Values to be top-level (children of the Benchmark element), prohibit such Rules from utilizing imported Values of their own, and have these Rules processed first before all other Rules.
    c. Option 3: Make no requirements on the benchmark document but require interpreters as part of the document loading to trace dependencies of import/export and ensure that evaluation of these Rules occurs in a corresponding order. Detection of circular dependencies would lead to an error.

Sample Proposal #1 – check-import as an archiving tool

We will assume that the purpose of check-import is to document information discovered by the checking system. (I.e. it collects system artifacts/evidence.) The @import-name attribute identifies some structure in the checking language from which to pull information. Legal checking-language structures that could be referenced by the @import-name attribute and the nature of the information returned from the checking system would be defined by higher-level documents such as NIST 800-126. The returned information would appear in the body of a check-import statement in the appropriate rule-result in the XCCDF TestResults.

Sample Proposal #2 – check-import for Value population

We will assume that the purpose of check-import is to populate Values from the checking system in order to feed those values back into the checking system via a check-export statement. In order to eliminate dependency issues, Values "populated" by a check-import are only considered to hold those values within the enclosing check statement. For an example, consider a Value that is set to 5 in a Benchmark Profile. Further assume that in one check that Value is named in a check-import statement and the checking system provides a value of 1. Within that particular check only, use of that Value in a check-export statement will export a value of 1. However, in all other checks which name that Value in a check-export statement the value exported will be 5, as set in the Profile. This will occur regardless of the order in which the various checks appear or are processed. In other words, the check-import allows for a temporary override of the value of a Value within a very narrow scope.

In fact, it might be easier to conceptualize the use of check-import as a local short-circuit that connects the check system information identified in the check-import with a specific check system variable named in the check-export. In this regard, one might say the value of the Value that connects the check-import with the check-export is never really modified and that the name of the Value is simply used as a bridge to connect two pieces of information in the checking system. With this understanding, one could see how batch-checkers would be able to handle check-import since the functional logic of the above scenario could be entirely contained in the checking engine without resorting to multiple transactions with the XCCDF interpreter. (Note that some sort of information needs to be passed back to the interpreter so that the check-import statement in the rule-result can properly annotate the value that was sent in the check-export directive, but this can be done in the return from the batch process.)

In order to facilitate this use, a new optional attribute will be added to check-import to identify the XCCDF Value used to connect the import with the export. The new attribute in check-import will be named @value-id and it is this attribute that will name an XCCDF Value, just as with check-export. In fact, the dictionary entry for check-import already references a @value-id attribute that would be used for this purpose, but this attribute was absent in the schema. If @value-id is absent in a check-import statement, the check-import statement will simply provide data for archiving in the appropriate rule-result as described in proposal 1.

```
<xsd:complexType name="checkImportType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
    Data type for the check-import element, which specifies a
    value that the benchmark author wishes to retrieve from the
    the checking system.  The import-name attribute gives the
    name or id of the value in the checking system. When the
    check-import element appears in the context of a rule-result,
    then the element's body is the collected value.  When the
    check-import element appears in the context of a Rule, then
    the body should be empty and any content must be ignored.
    The optional value-id attribute may be used to identify a
    Value that will temporarily (within the scope of the enclosing
    check element) take on the imported value.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="value-id" type="xsd:NCName" use="optional"/>
      <xsd:attribute name="import-name" type="xsd:string" use="required"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```

Adding Dublin Core to Status Entries

A user has requested that status elements within items be expanded to allow Dublin Core content. It was felt that doing this would expand the ability of authors to annotate Benchmarks. This corresponds to item XCCDF-48 in JIRA.

Issues

1. Should only Dublin Core be allowed or should it be open to other metadata formats.
2. Is the status element the correct place for this information or should it be added to a different element or be associated with a new element. Currently, statusType is a simple type. As far as the author is aware, the only way to add Dublin Core data to the status element in a way that does not deprecate existing content would be to make the status element a mixed content type, but that would eliminate the ability to restrict the limit the status text to the prescribed enumeration. As such, doing this would do away with many of the constraints currently placed on documents.

Sample Proposal

In this proposal, a metadata field with the metadataType type is added to Items. This allows Dublin Core, as well as other metadata formats, to be applied directly to Items.

```
<xsd:complexType name="itemType" abstract="1">
    <xsd:annotation>
      ...
    </xsd:annotation>
    <xsd:sequence>
      ...
      <xsd:element name="reference" type="cdf:referenceType"
            minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="metadata" type="cdf:metadataType"
            minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
      ...
  </xsd:complexType>
```

Clarify use of selected vs. role="unchecked" vs. UNCHECKED rule result

Users have expressed confusion as to the relationship between various role settings and other pieces of XCCDF functionality. For example, it was noted that a role of "unchecked" gives an "Unchecked" XCCDF result, but it was noted that "Unchecked" can also arise because of situations in a referenced checking engine and, as such, the appearance of an "Unchecked" results does not clearly indicate what caused that result. On a separate note, the processing instructions do not address the settings of the role property, meaning that, according to a strict reading of the XCCDF processing directions, the role property does not affect whether a rule is run or scored. At the very least, these instructions need to be updated. This corresponds to issue XCCDF-14 in JIRA.

Sample Proposal

The role property of rules will be deprecated on the grounds that the same functionality can be achieved through other mechanisms. Specifically, the unchecked capability (don't run checks but do include in output) could be accomplished by not selecting any check within the body of the Rule. For example, in a Profile, use the refine-rule element with a selector value of "NotACheckSelectorInTargetRule".

Similarly, a role of "unscored" could be simulated by setting the Rule's weight to 0, with the added tweak in the "Flat Unweighted" scoring model that rules with weight 0 retain their weights rather than being set to 1.0.

Deprecating "role" removes one contributor of complexity and removes some current ambiguity as to some results while removing no capability from XCCDF. In order to avoid a major change, role would not actually be removed until the next major release of XCCDF, but its use would be discouraged.

Clarify the processing model for group selection and requires capabilities

It was noted that under the current processing model, the interactions of Group and Rule selection can cause Rules to be selected even through some of their requirements are not met. For example:

-- (From Brent Arias)
Benchmark

  GroupA (selected="false")
    RuleX (selected="true")

  RuleY (selected="true", requires="RuleX")

According to the "Item Processing Algorithm", RuleX will never be processed or visited, because its parent "GroupA" was not selected. But as a child of the Benchmark, when RuleY is visited, its "requires" condition will be fulfilled - even though (intuitively) it should indirectly have been cause for "de-selection" due to GroupA.
--
The result of the above scenario would be the execution of RuleY without the execution of RuleX, which is likely a violation of the purpose of the requires statement.

Alternatively, consider three Rules, A, B, and C where A requires B and B requires C. After the application of a Profile but before Item processing, A and B are selected and C is de-selected. When A is processed the interpreter would see that B is selected, meeting the requires constraint, and Rule A would be run. When the interpreter reaches B, it would see that C is not selected, violating the requires constraint, and B would be de-selected. Thus, A would run even though B was not run.

This corresponds to issue XCCDF-17 in JIRA.

Issues

1. What is the purpose of the requires/conflicts statements? Is it to ensure that Rules are or are not run together, or is it just a tool for tailoring selection? If the latter, then it might be argued that the "requires" statement in RuleY in the above example is intended to detect only explicit de-selection of RuleX, rather than the implicit de-selection caused by the parent Group's de-selection.
2. While it sometimes results in unintuitive results as seen above, the current processing model is simple and unambiguous and is also robust against circular references. If we add multi-hop searches, circular dependencies become problematic and the algorithm becomes significantly more complicated. Is the simplicity of the current model worth the somewhat counter-intuitive behaviors described above?

Sample Proposal

Change the instructions for Item.Process so that "...if any required Items or their parent Groups are unselected, or any conflicting Items are selected with all their parent Groups also selected,

then set [selected to false]." This would ensure that Rules and Groups that are implicitly deselected before Item processing begins would be treated as deselected for the purpose of resolving requires and conflicts statements. This would address the unintuitive behavior noted in the above example. Processing of Items would take place in the order in which they exist within the XML document and the selected property of Items could not be changed more than once, as is dictated in the current version of XCCDF.

This proposal does not eliminate the second unintuitive behavior caused by the transitivity of requirements. Solving this so that a Rule is only run if the Rules that it requires are guaranteed to run and the Rules that it conflicts with are guaranteed not to run would require multiple passes and is deemed too costly at this time.

Allow XCCDF check-content-ref statements to refer to other XCCDF documents

Sometimes a recommendation may be to follow the policy listed in another document. Currently, this can only be handled by checking instructions that explicitly ask the user if this has been accomplished. However, if the second policy was also expressed in XCCDF, it is not unreasonable to expect that a checking tool could actually run an assessment based on that document's policy. This corresponds to issue XCCDF-53 in JIRA.

Issues

1. Referencing an XCCDF document is more complicated than referencing a check system because of XCCDF tailoring. Check references would need to pass additional information in order to identify a Profile and possibly to perform other tailoring activities. This not only requires the existence of additional fields, but standardization of a format to express this information.
2. Having XCCDF documents reference XCCDF documents adds another level of complexity to document processing. Is there sufficient payoff to make this worthwhile?

Sample Proposal

The check element would be given a new optional element called "check-control" consisting of one or more "any" element. This element would hold a XML structure that would be passed directly to the interpreter. Each language that could be called from XCCDF would be responsible for defining a schema for the transmission of additional information. Strict processing would ensure that the checking language interpreter would be able to understand the instructions. If the referenced language had no use for additional information, then there would be no need for it to define such a schema. This field could be used to convey such information as interpreter command-line instructions, details on what metadata the checking engine should return, and pre-processing instructions.

```
<xsd:complexType name="checkType">
    <xsd:annotation>
      ...
    </xsd:annotation>
    <xsd:sequence>
      ...
      <xsd:element name="check-export" type="cdf:checkExportType"
              minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="check-control" minOccurs="0" maxOccurs="1">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:any minOccurs="1" maxOccurs="unbounded"
                  processContents="strict"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="check-content-ref"
              minOccurs="0" maxOccurs="unbounded"
```

```
                type="cdf:checkContentRefType"/>
            ...
        </xsd:sequence>
            ...
    </xsd:complexType>
```

In the case of XCCDF, the following schema would be employed:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:cdf-ctl="http://checklists.nist.gov/xccdf/1.1-control"
    xmlns:cdf="http://checklists.nist.gov/xccdf/1.1"
    targetNamespace="http://checklists.nist.gov/xccdf/1.1-control">

    <xsd:import namespace="http://checklists.nist.gov/xccdf/1.1"
        schemaLocation="xccdf-1.1.4.xsd">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
                Import the XCCDF schema to get the profile selectors.
            </xsd:documentation>
        </xsd:annotation>
    </xsd:import>

    <xsd:element name="BenchmarkControl">
        <xsd:complexType>
            <xsd:choice minOccurs="0" maxOccurs="unbounded">
                <xsd:element name="select" minOccurs="0"
                    type="cdf:profileSelectType"/>
                <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:choice>
            <xsd:attribute name="profile" type="xsd:NCName"/>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

Using the above schema, an author could identify a Profile using the "profile" attribute (or indicate that no Profile should be used if the attribute was absent). Optionally, after any Profile was applied, the four different types of selectors, pulled from profileType, could be used to provide further customization of the target.

Check-refs without names and the "multiple" property

The name attribute of check-content-ref is optional, allowing external references to name only a checking language file. However, the specification does not comment on the behavior if the name attribute is missing and the schema simply notes that, in the absence of a name, the "...reference is to the entire other document." Users have requested that the specification provide more guidance as to how to handle this particular case. This corresponds to issue XCCDF-22 in JIRA.

In a closely related issue, users have expressed confusion over the "multiple" property of XCCDF Rules. A paragraph description of this property appears on page 22, but users still remain confused as to the purpose and use of the capability. This corresponds to issue XCCDF-35 in JIRA.

Issues

1. Important documents have already been published that include check-content-ref statements that only include the href and not the name attributes. As such, it is important for any description to be consistent with their behavior.
2. As noted, there are multiple ways to interpret the current description of "multiple". One is to claim that it refers to references that are, in fact, multiple checks and a value of "true" means that each of these checks should be reported separately, as opposed to being combined into a single result. Another is to claim that it refers to individual checks that are applicable to multiple system artifacts (e.g. recommendations that apply to multiple files or multiple registry keys) and that, if "multiple" is true, the compliance of each artifact should be reported individually.
3. The document does not specify how component checks should be combined if the "multiple" property is false. Should these component results be ANDed or ORed together?

Sample Proposal

The explanation of the "multiple" property will explicitly note, in the case of a check-content-ref which references a file with multiple checks, that each of these checks should be reported as if it was pointed to by a separate Rule with the same properties as the actual Rule. The results from these implicit rules would appear in their own rule-results entry in the TestResults. These rule-results would all have the same idref attribute and would instead be distinguished by other fields, such as a check-content-ref statement that named their source check within their check elements. If the "multiple" property is false, all component checks are ANDed together.

Clarify the order of operation of Profile selectors

In a previous discussion it was decided to change the inheritance model of Profile selectors from "append" to "replace" with the caveat that selectors with differing idref attributes are considered distinct. This was done to address prohibitions in the text that prevent two copies of a single selector from holding the same idref, thus preventing extending Profiles from overriding existing selectors in the extended Profile.

Since that time it has been noted that some edge cases are not addressed by this change. Specifically, consider the following Profile:

```
<Profile id ="Profile-A">
        ...
        <select idref="rule-x" selected="true">
        ...
        <select idref="cluster-y" selected="true">
        ...
</Profile>
```

Assume that rule-x is part of the cluster with the name cluster -y. In Profile-A, rule-x is selected first in its own selector and then selected again via its cluster-id association. In this case, this represents redundant behavior, but, especially as Benchmarks evolve and include legacy content, this scenario is not unlikely and is not prohibited by the XCCDF specification.

Consider now a second profile, Profile-B, which seeks to extend Profile-A. In particular, Profile-B wishes to de-select rule-x. However, if it includes <select idref="rule-x" selected="false"> this will replace the corresponding selector in Profile-A, but this will then get reversed by the selector for cluster-y. The result is, despite Profile-B's explicit de-selecting of rule-x, rule-x remains selected due to its cluster-id association. In fact, the only way to disable rule-x would be for Profile-B to override both rule-x and cluster-y, but this then forces Profile-B to manually select all other members of cluster-y in order to have rule-x be an explicit exception to the general behavior of the cluster. One can imagine how this complexity could turn nightmarish if cluster-y was followed by other customizations that affected its other members.

The problem is caused by the fact that Profile selectors are applied in the order in which they appear, but adopting a replace model given extending Profiles no control over where overriding selectors will appear in the order. This corresponds to issue XCCDF-13 in JIRA.

Issues

1. Do we want Profile extension to allow extending Profiles to override the selectors of the extended Profiles?
2. Do we wish to prohibit multiple copies of the same selector with the same idref? (If we remove this restriction, overriding can be accomplished through the original "append" model.)
3. Do we wish extending Profiles to be able to control whether their selectors appear before or after those of their parent?

Sample Proposals

There are three immediately apparent options:

1) The current XCCDF 1.1.4 spec: Selectors are appended during extension but no overlapping selector-idref pairs are permitted. Extending Profiles cannot override explicit selection actions in the source.
2) The proposal from the January 19 discussion: Selectors replace in the case of overlapping selector-idref pairs and append otherwise. This allows extending Profiles to override their parents with the caveat noted above.
3) Allowing of duplicates: Selectors are appended during extension and there is no prohibition about overlapping selector-idref pairs. Extending Profiles can override parents because extending Profile selectors would appear after extended Profile selectors, thus overriding their behavior.

Clarify the concept of "default values" in Values

Values can contain multiple value entries in support of tailoring actions. However, it is unclear which value should be in force in the absence of tailoring. While there is a "default" element, the text implies that this is to present a default suggestion to the user during tailoring rather than in any expectation that the Value will take on the default's value in the absence of tailoring. Moreover, like value elements, there can be many default elements making it unclear which should be used in the absence of tailoring. To address this issue, we must define a canonical way to identify a default configuration of Value structures in the absence of prior tailoring actions. This corresponds to issue XCCDF-41 in JIRA.

Issues

1. Currently selectors are available for all tailorable fields within a Value. In Rules, it is the selector that is used to determine the default (pre-tailoring) behavior: "When Profiles are not used, then all check elements with non-empty selectors are ignored." Do we want to follow this example and utilize the selector or should we create a separate attribute that denotes a default?
2. We cannot require the presence of any particular designator that a value be the default. However, all Values must have at least on value. How do we handle the case where there is no designated default value.
3. How do we want extension to affect defaults? Currently all tailorable fields within a Value use the "append" processing model except the choices field, which uses "prepend". XCCDF prohibits two instances of value (or default) from sharing the same selector (including the blank selector). This means that, if default behavior is controlled by the selector, extension cannot change default behavior. Do we want to change the processing model of these fields to something like the processing model for check, which is to replace values with duplicate selectors and append otherwise.
    a. The schema only prevents duplicate selectors in value or default. However, many other fields can be tailored by selectors and it is unclear what should happen if two instances of the same field shared a selector. Should all tailorable fields prohibit duplication of their selector values?

Sample Proposal

Following the lead of XCCDF Rule checks, prior to tailoring, all fields with non-empty selector attributes will be ignored. (Fields without a selector attribute are treated as having an empty selector.) The exception is the value element: prior to tailoring a value with an empty or absent selector will be treated as the only value. However, if no such value field exists, the first value field in the Value's XML will be treated as the only value, thus ensuring that all Values will have a value element even before tailoring.

In addition, added constraints will be placed on the tailorable fields of Value to ensure that no selector will select multiple instances of the same tailorable field:

```
<xsd:element name="Value" type="cdf:valueType">
  <xsd:unique name="valueSelectorKey">
    <xsd:selector xpath="./cdf:value"/>
    <xsd:field xpath="@selector"/>
  </xsd:unique>
```

```
    <xsd:unique name="defaultSelectorKey">
      <xsd:selector xpath="./cdf:default"/>
      <xsd:field xpath="@selector"/>
    </xsd:unique>
    <xsd:unique name="matchSelectorKey">
      <xsd:selector xpath="./cdf:match"/>
      <xsd:field xpath="@selector"/>
    </xsd:unique>
    <xsd:unique name="lower-boundSelectorKey">
      <xsd:selector xpath="./cdf:lower-bound"/>
      <xsd:field xpath="@selector"/>
    </xsd:unique>
    <xsd:unique name="upper-boundSelectorKey">
      <xsd:selector xpath="./cdf:upper-bound"/>
      <xsd:field xpath="@selector"/>
    </xsd:unique>
    <xsd:unique name="choicesSelectorKey">
      <xsd:selector xpath="./cdf:choices"/>
      <xsd:field xpath="@selector"/>
    </xsd:unique>
  </xsd:element>
```

Finally, the extension model of all tailorable Value fields (value, default, match, lower-bound, upper-bound, and choices) will be changed to the "replace" model with the caveat that fields that have differing selectors are not treated as being the same. This follows the example of the check field in Rules. Fields of the same type but with different selectors will be treated as if they used the "append" processing model. (This could be significant in the case of values if neither the extended nor the extending Values had a value with a blank selector. In such a case, the effective default value of the extended Value remains the default value.)

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. While this does allow for validation in the absence of network connectivity, it has the disadvantages of forcing these schemas to be distributed with the XCCDF as a bundle and also raises the possibility that users might end up working with obsolete referenced schemas if one of those schemas is revised. It has been suggested that, instead of local references, imports reference the canonical sources of the named schemas.

As noted, pulling schemas from canonical remote sources ensures that all users of the XCCDF schema are using the same referenced schemas. On the other hand, changes to a referenced schema could deprecate content that utilizes this schema. Using local copies of files fixes XCCDF's references to specific versions of files. It should be noted, however, that many canonical schema locations may offer remote links to specific versions of a schema, thus eliminated this concern. Moreover, remote references force XCCDF interpreters to add the ability to pull these remote schemas in order to perform validation.

On the other hand, as noted in the discussion of using the latest CPE version, sometimes we might wish to ensure that changes to a referenced schema are automatically reflected. In such an instance, there are benefits to pulling the references schema from a canonical remote site.

This corresponds to issue XCCDF-46 in JIRA.

Issues

1. Do we wish to make a blanket determination as to whether to import locally or remotely or should we do this on a case-by-case basis?

Sample Proposal

Self-explanatory

Add an Enumeration to Classify types of Notice Elements

Currently notice elements in Benchmarks are used for a range of purposes, including usage disclaimers, copyright notifications, acknowledgements, and similar information. XCCDF provides no way to distinguish between these different uses of this field, but tools and document conversion procedures might wish to present different types of notices using different structures and appearances. It has been suggested that a flag be added to the notice element that could be used to classify the content in order to enhance processing. This corresponds to issue XCCDF-49 in JIRA.

Issues

1. Even if a list of standard types of notices is created, we will not be standardizing if/how tools treat those different notice types. As such, given that the standardization will not standardize behavior, is this worth changing the spec?

Sample Proposal

We add an optional "type" attribute to the noticeType complex type. This new attribute has an enumerated list of possible values:

- copyright – indicates the notice holds copyright information or other ownership markings
- warning – indicates warnings or disclaimers addressed to users regarding use of the benchmark
- license – indicates the document's license agreement or other controls on the use or dissemination of the document
- general – indicates a notice that does not fall into the above categories. This is the default value of this attribute.

Tools may use this information to change the way notices are presented but are not required to do so. The way in which information is displayed is up to the tool implementer with the exception that tools may not remove or hide notice information based on this attribute.

```
<xsd:complexType name="noticeType" mixed="true">
  <xsd:annotation>
    ...
  </xsd:annotation>
    ...
  <xsd:attribute name="type" use="optional" default="general"
          type="cdf:noticeEnumType"/>
</xsd:complexType>

<xsd:simpleType name="noticeEnumType">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
      Lists some different types of notices so tools can used
      different display methods.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="copyright"/>
```

```
        <xsd:enumeration value="warning"/>
        <xsd:enumeration value="license"/>
        <xsd:enumeration value="general"/>
    </xsd:restriction>
</xsd:simpleType>
```

Structures for Stand-Alone TestResults

In a previous discussion it was noted that it was possible to create a single, stand-alone TestResult or a single file with multiple TestResults providing they were preceded by all the Rules and Values that these results referenced. A suggestion was made to support multiple TestResults in a single file without requiring the inclusion of referenced Rules and Values, but at the time it was unclear if there was significant need for such a change. Since that time, others have added to the call for this capability and some additional capabilities besides. This corresponds to issue XCCDF-47 in JIRA.

Issues

1. How do we best handle multiple TestResults?
    a. Remove restrictions in Benchmarks that force TestResult reference to resolve?
    b. Create a new root element that contains just TestResults
2. If large sets of TestResults will exist apart from the Benchmark that created them, what additional information, if any, should we include to ensure important information is retained? It has been suggested that a metadata field be added to TestResults to hold a copy of the Benchmark metadata element. Are there other pieces of information we would like to copy?

Sample Proposal

We create a new root element called BenchmarkResults. This element may contain any number of TestResult elements. It has no other elements. As such, a single BenchmarkResults could contain the TestResults from different assessments or even different Benchmarks. Moreover, we add an optional metadata element to testResultType. The intent of this element is to hold a copy of the Benchmark's metadata. As such, it is added as a child of the Benchmark element to indicate this association.

```
<xsd:element name="BenchmarkResults">
  <xsd:annotation>
    <xsd:documentation>
      Used to hold any number of TestResult objects without
      requiring copies of their source Benchmark.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="cdf:TestResult" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:complexType name="testResultType">
    ...
  <xsd:sequence>
    <xsd:element name="benchmark" minOccurs="0" maxOccurs="1">
      <xsd:complexType>
        <xsd:sequence>
```

```
            <xsd:element name="metadata" type="cdf:metadataType"
              minOccurs="0" maxOccurs="unbounded"/>
          </xsd:sequence>
          <xsd:attribute name="href" type="xsd:anyURI"
                  use="required"/>
        </xsd:complexType>
      </xsd:element>
      ...
    </xsd:sequence>
    ...
</xsd:complexType>
```

Update Truth Tables to Address all XCCDF Results

The truth tables on pages 53 and 54 of the XCCDF documentation outline how to treat combinations of differing results from the components of a complex check. However, these tables only cover 6 of the 9 possible XCCDF results. Moreover, the preceding paragraphs of the specification claim that the abbreviations used in the table can be found in the description of the 'result' element in the TestResult object on page 49. In fact, TestResult objects have no 'result' element and the abbreviations are defined in the description of the rule-result object on pages 31 & 32. The errors should be corrected and the table updated to include all 9 possible XCCDF results. This corresponds to issue XCCDF-1 in JIRA.

Note, in all the truth tables it is not necessary to provide details on results of "Fixed" because, as is explained in the preamble to the truth tables, this is treated as being equivalent to "Pass". For reference, the abbreviations for the various results are duplicated here:

- P = Pass
- F = Fail
- U = Unknown
- E = Error
- N = NotApplicable
- K = NotChecked
- S = NotSelected
- I = Informational
- X = Fixed (treated as P)

Change Details:

DOCUMENTATION: Page 53 & 54 – Correct preamble to truth tables

Truth tables for boolean operation in complex checks are given below; all the abbreviations in the truth tables come from the description of the ~~'result' element in the TestResult~~ TestResult/rule-result object (see ~~page 49~~ pages 31 & 32).

DOCUMENTATION: Page 53 & 54 – AND Truth Table

Replace the truth table with the following:

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

DOCUMENTATION: Page 54 – OR Truth Table

Replace the truth table with the following:

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

DOCUMENTATION: Page 54 – NOT Truth Table

Replace the truth table with the following:

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

Remove Reference to check-export from Last Paragraph of <check>

The last paragraph of the documentation for <check> reads: "When a check element is a child of a Rule object, check-import and check-export elements must be empty. When a check element is a child rule-result object, check-import elements contain the value retrieved from the checking system." This is misleading - check-export is always "empty" in that it never has a body. It should be removed from this reference. This corresponds to item XCCDF-23 in JIRA.

Change Details:

DOCUMENTATION: Page 51

Change from:
When a check element is a child of a Rule object, check-import and check-export elements must be empty.

To:
When a check element is a child of a Rule object, the check-import element must be empty.

Incorrect Assertion that <reference> has an xml:lang attribute

The XCCDF specification states that <reference> elements have xml:lang attributes. However, the schema does not include this attribute. Moreover, the body of the referenceType is supposed to contain either Dublin Core XML or string content corresponding to a Dublin Core 'title'. (See <reference> dictionary element on pages 68-69.) Given this, correct usage of this field should not need a lang descriptor - content should always come from the Dublin Core schema. This corresponds to issue XCCDF-27 in JIRA.

Change Details:

DOCUMENTATION Page 68:

<reference>

...

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

Incorrect claim regarding <description> attributes

The description of the <description> element states "It has no attributes." This is incorrect. Description has two attributes. The documentation should be corrected. This corresponds to issue XCCDF-28 in JIRA.

Change Details:

DOCUMENTATION: Page 55

<description>

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

Reference to non-existent "optional" element in Item processing

The Item.Select row of the Item Processing table on page 36 states:

--

If any of the following conditions holds, cease processing of this Item.

1. The processing type is Tailoring, and the optional property and selected property are both false.

...

--

However, there is no "optional" property in items. The reference should be removed. This corresponds to issue XCCDF-29 in JIRA.

Change Details:

DOCUMENTATION: Page 36 –

Sub-Step	Description
Item.Select	If any of the following conditions holds, cease processing of this Item. 1. The processing type is Tailoring, and the ~~optional property and~~ selected property ~~are both~~ 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.

Redundant Description of cluster-id in both Item and Group

The cluster-id attribute is listed as a child of Item (p 17) and a child of Group (p 18). However, since Group extends Item, it is implicitly a child of Group already. No other elements of Item are repeated on extensions of Item. For consistency, cluster-id should be removed from the description of Group. This corresponds to issue XCCDF-32 in JIRA.

Change Details:

DOCUMENTATION: Page 18 – Removing cluster-id from description of Group

Group :: Item

Property	Type	Count	Description
requires	identifier	0-n	The id of another Group or Rule in the Benchmark that must be selected for this Group to be applied and scored properly
...	...	...	...
platform	URI	0-n	Platforms to which this Group applies, CPE Names or CPE platform specification identifiers
~~cluster-id~~	~~identifier~~	~~0-1~~	~~An identifier to be used from Benchmark profiles to refer to multiple Groups and Rules, optional~~
extends	identifier	0-1	An id of a Group on which to base this Group

...

DOCUMENTATION: Page 45 – Removing cluster-id from Group dictionary entry

A Group element contains descriptive information about a portion of a Benchmark, as well as Rules, Values, and other Groups. A Group must have a unique id attribute to be referenced from other XCCDF documents or extended by other Groups. The id attribute must be a unique identifier. The 'extends' attribute, if present, must have a value equal to the id attribute of another Group. ~~The 'cluster-id' attribute is an id; it designates membership in a cluster of Items, which are used for controlling Items via Profiles.~~ The 'hidden' and 'allowChanges' attributes are of boolean type and default to false. The weight attribute is a positive real number.

Clarify Loading.Resolve.Items and Loading.Resolve.Profiles

In Loading.Resolve.Items and in Loading.Resolve.Profiles, the word prepend is used repeatedly in the description of how the property sequence should be processed. The Processing Models table indicates that the appropriate action is not always prepend but can be one of none, prepend, append, replace and override. In order to avoid confusion, some text should be amended to remove any implication of loss of generality. This corresponds to issue XCCDF 34 in JIRA.

In addition, in Loading.Resolve.Items, the specification states to remove duplicate properties, but the corresponding instructions in Loading.Resolve.Profiles states to remove all but the last instance of duplicate properties. Loading.Resolve.Items should be amended, following the example of Loading.Resolve.Profiles, to remove any ambiguity. This corresponds to issue XCCDF-33 in JIRA.

Change Details:

DOCUMENTATION: Page 34

Sub-Step	Description
Loading.Noticing	For each notice property of the Benchmark object, add the notice to the tool's set of legal notices. If a notice with an identical id value is already a member of the set, then replace it. If the Benchmark's resolved property is set, then Loading succeeds, otherwise go to the next step: Loading.Resolve.Items.
Loading.Resolve.Items	For each Item in the Benchmark that has an extends property, resolve it by using the following steps: (1) if the Item is Group, resolve all the enclosed Items, (2) resolve the extended Item, (3) ~~prepend the property sequence from the extended Item to the extending Item~~ insert the necessary property sequences from the extended Item into the appropriate locations in the extending Item (4) if the Item is a Group, assign values for the id properties of Items copied from the extended Group, (5) remove all but the last instance of duplicate properties and apply property overrides, and (6) remove the extends property. If any Item's extends property identifier does not match the identifier of a visible Item of the same type, then Loading fails. If the directed graph formed by the extends properties includes a loop, then Loading fails. Otherwise, go to the next step: 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) ~~prepend the property sequence from the extended Profile to that of the extending Profile~~ 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. 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.

Fix Statement of Ordering of Profiles, Rules, Groups, and Values

There is an error in the specification where it states that all Values, Groups, and Rules must precede all Profiles. The schema says exactly the opposite - Profiles come first, then Values, then Rules and Groups, and finally TestResults. This corresponds to issue XCCDF-38 in JIRA.

Change Details:

DOCUMENTATION: Page 16

Conceptually, a Benchmark contains Group, Rule, and Value objects, and it may also contain Profile and TestResult objects. For ease of reading and simplicity of scoping, all Value objects must precede all Groups and Rules, which must precede all Profiles, which must precede all TestResults. These objects may be directly embedded in the Benchmark, or incorporated via W3C standard XML Inclusion [10].

becomes:

Conceptually, a Benchmark contains Group, Rule, and Value objects, and it may also contain Profile and TestResult objects. For ease of reading and simplicity of scoping, all Profiles must precede all Groups, Rules, and Values. Groups can contain Values, Rules, and other Groups. Within any level of the Group hierarchy (including at the top level, within the Benchmark itself), Values must precede sibling Groups and Rules. All Values, Groups, and Rules must precede all TestResults. These objects may be directly embedded in the Benchmark, or incorporated via W3C standard XML Inclusion [10].

Document use of selector attribute in Profile/refine-rule

There is currently no description of the @select attribute capability in Profile/refine-rule despite explicit references to the other capabilities of refine-rule. We should expand the portions of the spec that deal with what a refine-rule statement can do to include selection of check statements. This corresponds to issue XCCDF-42 in JIRA.

Change Details:

DOCUMENTATION: Page 28

refine-rule – a Rule/Group selector. This selector allows the Profile author to select check statements, override the scoring weight, severity, and role of a Rule, Group, or cluster of Rules and Groups. Despite the name, this selector does apply for Groups, but only to their weight property.

Typo in Profile dictionary entry

The last sentence of the first paragraph of the Profile dictionary entry states that there are 3 selectors but, in fact, there are 4. This corresponds to issue XCCDF-45 in JIRA.

Change Details:

DOCUMENTATION: Page 48 – <Profile> Updating Profile dictionary to reflect four selectors

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