

ISM Schematron Rules

The following documentation is informative. The actual Schematron files are the normative record. This documentation is generated from the Schematron files via XSLT it may be missing some file or pieces of a file but whatever is here other than titles came from the original file.

It is envisioned that this will be a useful to search and read resource but for questions and debates the source files should be consulted. This document is laid out by starting with rules that are restricted such as FOUO or above, followed by supporting files that are not actually rules but form the basis for the rules and finally all of the unclassified rules.

Rules are all of the format ISM-ID-XXXXXX any other heading is a supporting file that may strongly influence a rule but is not actually a numbered rule.

FileName:./ISM_XML.sch

Code Description:

This is the root file for the ISM Schematron ruleset. It loads all of the required CVEs declares some variables and includes all of the Rule .sch files.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- UNCLASSIFIED --><!-- WARNING:
Once compiled into an XSLT the result will
be the aggregate classification of all the CVEs
and included .sch files
-->
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron"
xmlns:cve="urn:us:gov:ic:cve:v1"
queryBinding="xslt2">
<sch:ns uri="urn:us:gov:ic:ism" prefix="ism"/>
<sch:ns uri="urn:us:gov:ic:cve:v1" prefix="cve"/>

<!-- (U) Resources -->
<sch:let name="countriesList"
value="document(' ./_CVE/CVEnumISMOwnerProducer.xml ' )//cve:CVE/cve:Enumeration/cve:Term/
cve:Value"/>
<sch:let name="classificationAllList"
value="document(' ./_CVE/CVEnumISMClassificationAll.xml ' )//cve:CVE/cve:Enumeration/
cve:Term/cve:Value"/>
<sch:let name="classificationNonUSList"
value="document(' ./_CVE/CVEnumISMClassificationNonUS.xml ' )//cve:CVE/cve:Enumeration/
cve:Term/cve:Value"/>
<sch:let name="classificationUSList"
value="document(' ./_CVE/CVEnumISMClassificationUS.xml ' )//cve:CVE/cve:Enumeration/cve:Term/
cve:Value"/>
<sch:let name="ownerProducerList"
value="document(' ./_CVE/CVEnumISMOwnerProducer.xml ' )//cve:CVE/cve:Enumeration/cve:Term/
cve:Value"/>
<sch:let name="declassExceptionList"
value="document(' ./_CVE/CVEnumISM25X.xml ' )//cve:CVE/cve:Enumeration/cve:Term/cve:Value"/>
<sch:let name="FGISourceOpenList"
value="document(' ./_CVE/CVEnumISMFGIOpen.xml ' )//cve:CVE/cve:Enumeration/cve:Term/
cve:Value"/>
<sch:let name="FGISourceProtectedList"
value="document(' ./_CVE/CVEnumISMFGIProtected.xml ' )//cve:CVE/cve:Enumeration/cve:Term/
cve:Value"/>
<sch:let name="nonICmarkingsList"
value="document(' ./_CVE/CVEnumISMNonIC.xml ' )//cve:CVE/cve:Enumeration/cve:Term/cve:Value"/>
>
<sch:let name="releasableToList"
```

```

value="document('._CVE/CVEnumISMRelTo.xml')//cve:CVE/cve:Enumeration/cve:Term/cve:Value"/
>
<sch:let name="SCIcontrolsList"
value="document('._CVE/CVEnumISMSCIcontrols.xml')//cve:CVE/cve:Enumeration/cve:Term/
cve:Value"/>
<sch:let name="SARIdentifierList"
value="document('._CVE/CVEnumISM SAR.xml')//cve:CVE/cve:Enumeration/cve:Term/cve:Value"/>
<sch:let name="validAttributeList"
value="document('._CVE/CVEnumISMAttributes.xml')//cve:CVE/cve:Enumeration/cve:Term/
cve:Value"/>
<sch:let name="noticeList"
value="document('._CVE/CVEnumISMNotice.xml')//cve:CVE/cve:Enumeration/cve:Term/
cve:Value"/>
<sch:let name="nonUSControlsList"
value="document('._CVE/CVEnumISMNonUSControls.xml')//cve:CVE/cve:Enumeration/cve:Term/
cve:Value"/>
<sch:let name="compliesWithList"
value="document('._CVE/CVEnumISMCompliesWith.xml')//cve:CVE/cve:Enumeration/cve:Term/
cve:Value"/>
<sch:let name="atomicEnergyMarkingsList"
value="document('._CVE/CVEnumISMAtomicEnergyMarkings.xml')//cve:CVE/cve:Enumeration/
cve:Term/cve:Value"/>
<sch:let name="displayOnlyToList"
value="document('._CVE/CVEnumISMRelTo.xml')//cve:CVE/cve:Enumeration/cve:Term/cve:Value"/
>

<!-- (U) Resources that may include FOUO values -->
<sch:let name="disseminationControlsList"
value="document('._CVE/CVEnumISMDissem.xml')//cve:CVE/cve:Enumeration/cve:Term/
cve:Value"/>

<!-- (U) Import Properites -->
<sch:include href="._Properties/resourceElement.sch"/>
<sch:include href="._Properties/createDate.sch"/>
<sch:include href="._Properties/partTags.sch"/>

<!------->
<!-- (U) Universal Lets -->
<!------->

<!-- (U) determine position of resource element -->
<sch:let name="resourceElementPosition"
value=" index-of((for $each in (/*) return if($each/@ism:resourceElement=true()) then 1
else 0),1)[1] "/>

<!-- (U) Integer value of the ISM-RESOURCE-CREATE-DATE, the ism:createDate attribute on
the
resource element. -->

```

```

<sch:let name="ISM_RESOURCE_CREATE_DATE_INT"
value="number(concat(substring($ISM_RESOURCE_CREATE_DATE,1,4),
concat(substring($ISM_RESOURCE_CREATE_DATE,6,2),
substring($ISM_RESOURCE_CREATE_DATE,9,2))))"/>

<!-- (U) Is this a CAPCO applicable resource -->
<sch:let name="ISM_CAPCO_RESOURCE"
value=" if(contains($ISM_RESOURCE_ELEMENT/@ism:ownerProducer, 'USA')) then true() else
false()"/>

<!-- (U) Is this a ICD-710 applicable resource -->
<sch:let name="ISM_ICD_710_APPLIES"
value="if(contains($ISM_RESOURCE_ELEMENT/@ism:compliesWith, 'ICD-710')) then true() else
false()"/>

<!-- (U) Is this a DoD5230.24 applicable resource -->
<sch:let name="ISM_DOD5230_24_APPLIES"
value="if(contains($ISM_RESOURCE_ELEMENT/@ism:compliesWith, 'DoD5230.24')) then true()
else false()"/>

<!-- (U) Does the current Classified National Security Information Executive Order apply
to this resource -->
<sch:let name="ISM_NSI_EO_APPLIES"
value=" if(not($ISM_CAPCO_RESOURCE) or ($ISM_RESOURCE_ELEMENT/@ism:classification='U') or
index-of(//*[not(@ism:excludeFromRollup=true())]/@ism:atomicEnergyMarkings,'FRD')>0 or
index-of(//*[not(@ism:excludeFromRollup=true())]/@ism:atomicEnergyMarkings,'RD')>0 or
($ISM_RESOURCE_CREATE_DATE_INT < 19960414)) then false() else true()"/>

<!-- (U) Get Banner Attributes -->
<sch:let name="bannerClassification"
value="normalize-space($ISM_RESOURCE_ELEMENT/@ism:classification)"/>
<sch:let name="bannerDisseminationControls"
value="normalize-space($ISM_RESOURCE_ELEMENT/@ism:disseminationControls)"/>
<sch:let name="bannerDisplayOnlyTo"
value="normalize-space($ISM_RESOURCE_ELEMENT/@ism:displayOnlyTo)"/>
<sch:let name="bannerNonICmarkings"
value="normalize-space($ISM_RESOURCE_ELEMENT/@ism:nonICmarkings)"/>
<sch:let name="bannerFGISourceOpen"
value="normalize-space($ISM_RESOURCE_ELEMENT/@ism:FGISourceOpen)"/>
<sch:let name="bannerFGISourceProtected"
value="normalize-space($ISM_RESOURCE_ELEMENT/@ism:FGISourceProtected)"/>
<sch:let name="bannerSCIconcontrols"
value="normalize-space($ISM_RESOURCE_ELEMENT/@ism:SCIconcontrols)"/>
<sch:let name="bannerNotice" value="normalize-space($ISM_RESOURCE_ELEMENT/@ism:notice)"/>
<sch:let name="bannerAtomicEnergyMarkings"
value="normalize-space($ISM_RESOURCE_ELEMENT/@ism:atomicEnergyMarkings)"/>

```

```

<!-- (U) Tokenize Banner Attributes -->
<sch:let name="bannerDisseminationControls_tok"
value="tokenize($bannerDisseminationControls, ' ')" />
<sch:let name="bannerDisplayOnlyTo_tok" value="tokenize($bannerDisplayOnlyTo, ' ')" />
<sch:let name="bannerNonICmarkings_tok" value="tokenize($bannerNonICmarkings, ' ')" />
<sch:let name="bannerFGISourceOpen_tok" value="tokenize($bannerFGISourceOpen, ' ')" />
<sch:let name="bannerFGISourceProtected_tok"
value="tokenize($bannerFGISourceProtected, ' ')" />
<sch:let name="bannerSCIcontrols_tok" value="tokenize($bannerSCIcontrols, ' ')" />
<sch:let name="bannerNotice_tok" value="tokenize($bannerNotice, ' ')" />
<sch:let name="bannerAtomicEnergyMarkings_tok"
value="tokenize($bannerAtomicEnergyMarkings, ' ')" />

<!-- (U) Get Part Attributes -->
<sch:let name="partClassification"
value=" for $token in $partTags/@ism:classification return tokenize(normalize-
space($token), ' ')" />
<sch:let name="partDisseminationControls"
value=" for $token in $partTags/@ism:disseminationControls return tokenize(normalize-
space($token), ' ')" />
<sch:let name="partDisplayOnlyTo"
value=" for $token in $partTags/@ism:displayOnlyTo return tokenize(normalize-
space($token), ' ')" />
<sch:let name="partAtomicEnergyMarkings"
value=" for $token in $partTags/@ism:atomicEnergyMarkings return tokenize(normalize-
space($token), ' ')" />
<sch:let name="partNonICmarkings"
value=" for $token in $partTags/@ism:nonICmarkings return tokenize(normalize-
space($token), ' ')" />
<sch:let name="partFGISourceOpen"
value="for $token in $partTags/@ism:FGISourceOpen return tokenize($token, ' ')" />
<sch:let name="partFGISourceProtected"
value="for $token in $partTags/@ism:FGISourceProtected return tokenize($token, ' ')" />
<sch:let name="partSCIcontrols"
value=" for $token in $partTags return tokenize(normalize-space($token/@ism:SCIcontrols), '
 ')" />
<sch:let name="partNotice"
value=" for $token in $partTags/@ism:notice return tokenize(normalize-space($token), ' ')" /
>

<!-- (U) Tokenize portion Attributes -->
<sch:let name="partClassification_tok"
value=" for $token in $partClassification return tokenize($token, ' ')" />
<sch:let name="partDisseminationControls_tok"
value=" for $token in $partDisseminationControls return tokenize($token, ' ')" />
<sch:let name="partDisplayOnlyTo_tok"
value=" for $token in $partDisplayOnlyTo return tokenize($token, ' ')" />
<sch:let name="partAtomicEnergyMarkings_tok"

```

```

value=" for $token in $partAtomicEnergyMarkings return tokenize($token, ' ')" />
<sch:let name="partNonICmarkings_tok"
value=" for $token in $partNonICmarkings return tokenize($token, ' ')" />
<sch:let name="partSCIcontrols_tok"
value=" for $token in $partSCIcontrols return tokenize($token, ' ')" />
<sch:let name="partNotice_tok"
value=" for $token in $partNotice return tokenize($token, ' ')" />

<!-- (U) Check rollup of attributes in portions to the banner -->
<sch:let name="dcTags"
value="for $piece in $disseminationControlsList return $piece/text()" />
<sch:let name="dcTagsFound"
value="for $token in $dcTags return if ( index-of($partDisseminationControls_tok,$token)
&gt; 0 and (not(index-of($bannerDisseminationControls_tok,$token) &gt; 0)) ) then $token
else null" />
<sch:let name="aeaTags"
value="for $piece in $atomicEnergyMarkingsList return $piece/text()" />
<sch:let name="aeaTagsFound"
value=" for $token in $aeaTags return if ( index-of($partAtomicEnergyMarkings_tok,$token)
&gt; 0 and (not(index-of($bannerAtomicEnergyMarkings_tok,$token) &gt; 0)) ) then $token
else null" />

<!-- (U) Abstract Classes -->
<sch:include href="._Lib/ValidateValidAttributeValuesAgainstList.sch"/>
<sch:include href="._Lib/ValidateSortedAttributeValuesAgainstList.sch"/>
<sch:include href="._Lib/deprecatedError.sch"/>
<sch:include href="._Lib/deprecatedWarning.sch"/>

<!--*****-->
<!-- (U) ISM ID Rules -->
<!--*****-->

<!--(U) atomicEnergyMarkings-->
<sch:include href="._Rules/atomicEnergyMarkings/ISM_ID_00173.sch"/>
<sch:include href="._Rules/atomicEnergyMarkings/ISM_ID_00174.sch"/>
<sch:include href="._Rules/atomicEnergyMarkings/ISM_ID_00175.sch"/>
<sch:include href="._Rules/atomicEnergyMarkings/ISM_ID_00176.sch"/>
<sch:include href="._Rules/atomicEnergyMarkings/ISM_ID_00178.sch"/>
<sch:include href="._Rules/atomicEnergyMarkings/ISM_ID_00181.sch"/>
<sch:include href="._Rules/atomicEnergyMarkings/ISM_ID_00182.sch"/>
<sch:include href="._Rules/atomicEnergyMarkings/ISM_ID_00183.sch"/>
<sch:include href="._Rules/atomicEnergyMarkings/ISM_ID_00184.sch"/>
<sch:include href="._Rules/atomicEnergyMarkings/ISM_ID_00185.sch"/>

<!--(U) classification-->
<sch:include href="._Rules/classification/ISM_ID_00015.sch"/>
<sch:include href="._Rules/classification/ISM_ID_00016.sch"/>
<sch:include href="._Rules/classification/ISM_ID_00040.sch"/>

```

```
<sch:include href="./Rules/classification/ISM_ID_00142.sch"/>

<!--(U) classifiedBy-->
<sch:include href="./Rules/classifiedBy/ISM_ID_00017.sch"/>

<!--(U) declassException-->
<sch:include href="./Rules/declassException/ISM_ID_00133.sch"/>

<!--(U) derivativelyClassifiedBy-->
<sch:include href="./Rules/derivativelyClassifiedBy/ISM_ID_00143.sch"/>

<!--(U) displayOnlyTo-->
<sch:include href="./Rules/displayOnlyTo/ISM_ID_00167.sch"/>
<sch:include href="./Rules/displayOnlyTo/ISM_ID_00168.sch"/>

<!--(U) disseminationControls-->
<sch:include href="./Rules/disseminationControls/ISM_ID_00026.sch"/>
<sch:include href="./Rules/disseminationControls/ISM_ID_00028.sch"/>
<sch:include href="./Rules/disseminationControls/ISM_ID_00030.sch"/>
<sch:include href="./Rules/disseminationControls/ISM_ID_00031.sch"/>
<sch:include href="./Rules/disseminationControls/ISM_ID_00033.sch"/>
<sch:include href="./Rules/disseminationControls/ISM_ID_00034.sch"/>
<sch:include href="./Rules/disseminationControls/ISM_ID_00094.sch"/>
<sch:include href="./Rules/disseminationControls/ISM_ID_00107.sch"/>
<sch:include href="./Rules/disseminationControls/ISM_ID_00124.sch"/>
<sch:include href="./Rules/disseminationControls/ISM_ID_00140.sch"/>
<sch:include href="./Rules/disseminationControls/ISM_ID_00164.sch"/>
<sch:include href="./Rules/disseminationControls/ISM_ID_00169.sch"/>
<sch:include href="./Rules/disseminationControls/ISM_ID_00213.sch"/>
<sch:include href="./Rules/disseminationControls/ISM_ID_00215.sch"/>
<sch:include href="./Rules/disseminationControls/ISM_ID_10001.sch"/>
<sch:include href="./Rules/disseminationControls/ISM_ID_10003.sch"/>

<!--(U) FGISourceOpen-->
<sch:include href="./Rules/FGISourceOpen/ISM_ID_00095.sch"/>

<!--(U) FGISourceProtected-->
<sch:include href="./Rules/FGISourceProtected/ISM_ID_00096.sch"/>
<sch:include href="./Rules/FGISourceProtected/ISM_ID_00097.sch"/>

<!--(U) generalConstraints-->
<sch:include href="./Rules/generalConstraints/ISM_ID_00002.sch"/>
<sch:include href="./Rules/generalConstraints/ISM_ID_00012.sch"/>
<sch:include href="./Rules/generalConstraints/ISM_ID_00102.sch"/>
<sch:include href="./Rules/generalConstraints/ISM_ID_00103.sch"/>
<sch:include href="./Rules/generalConstraints/ISM_ID_00119.sch"/>
<sch:include href="./Rules/generalConstraints/ISM_ID_00125.sch"/>
<sch:include href="./Rules/generalConstraints/ISM_ID_00126.sch"/>
```



```
<sch:include href="./Rules/generalConstraints/ISM_ID_00166.sch"/>
<sch:include href="./Rules/generalConstraints/ISM_ID_00170.sch"/>
<sch:include href="./Rules/generalConstraints/ISM_ID_00179.sch"/>
<sch:include href="./Rules/generalConstraints/ISM_ID_00180.sch"/>
<sch:include href="./Rules/generalConstraints/ISM_ID_00188.sch"/>
<sch:include href="./Rules/generalConstraints/ISM_ID_00189.sch"/>
<sch:include href="./Rules/generalConstraints/ISM_ID_00190.sch"/>
<sch:include href="./Rules/generalConstraints/ISM_ID_00191.sch"/>
<sch:include href="./Rules/generalConstraints/ISM_ID_00192.sch"/>
<sch:include href="./Rules/generalConstraints/ISM_ID_00193.sch"/>
<sch:include href="./Rules/generalConstraints/ISM_ID_00194.sch"/>
<sch:include href="./Rules/generalConstraints/ISM_ID_00195.sch"/>
<sch:include href="./Rules/generalConstraints/ISM_ID_00196.sch"/>
<sch:include href="./Rules/generalConstraints/ISM_ID_00197.sch"/>
<sch:include href="./Rules/generalConstraints/ISM_ID_00198.sch"/>
<sch:include href="./Rules/generalConstraints/ISM_ID_00199.sch"/>
<sch:include href="./Rules/generalConstraints/ISM_ID_00200.sch"/>
<sch:include href="./Rules/generalConstraints/ISM_ID_00201.sch"/>
<sch:include href="./Rules/generalConstraints/ISM_ID_00202.sch"/>
<sch:include href="./Rules/generalConstraints/ISM_ID_00203.sch"/>
<sch:include href="./Rules/generalConstraints/ISM_ID_00204.sch"/>
<sch:include href="./Rules/generalConstraints/ISM_ID_00205.sch"/>
<sch:include href="./Rules/generalConstraints/ISM_ID_00206.sch"/>
<sch:include href="./Rules/generalConstraints/ISM_ID_00207.sch"/>
<sch:include href="./Rules/generalConstraints/ISM_ID_00208.sch"/>
<sch:include href="./Rules/generalConstraints/ISM_ID_00209.sch"/>
<sch:include href="./Rules/generalConstraints/ISM_ID_00210.sch"/>
<sch:include href="./Rules/generalConstraints/ISM_ID_00211.sch"/>
<sch:include href="./Rules/generalConstraints/ISM_ID_00212.sch"/>

<!--(U) nonICmarkings-->
<sch:include href="./Rules/nonICmarkings/ISM_ID_00035.sch"/>
<sch:include href="./Rules/nonICmarkings/ISM_ID_00036.sch"/>
<sch:include href="./Rules/nonICmarkings/ISM_ID_00037.sch"/>
<sch:include href="./Rules/nonICmarkings/ISM_ID_00038.sch"/>
<sch:include href="./Rules/nonICmarkings/ISM_ID_00148.sch"/>

<!--(U) nonUSControls-->
<sch:include href="./Rules/nonUSControls/ISM_ID_00163.sch"/>

<!--(U) notice-->
<sch:include href="./Rules/notice/ISM_ID_00127.sch"/>
<sch:include href="./Rules/notice/ISM_ID_00128.sch"/>
<sch:include href="./Rules/notice/ISM_ID_00129.sch"/>
<sch:include href="./Rules/notice/ISM_ID_00130.sch"/>
<sch:include href="./Rules/notice/ISM_ID_00134.sch"/>
<sch:include href="./Rules/notice/ISM_ID_00135.sch"/>
<sch:include href="./Rules/notice/ISM_ID_00136.sch"/>
```

```
<sch:include href="./Rules/notice/ISM_ID_00137.sch"/>
<sch:include href="./Rules/notice/ISM_ID_00138.sch"/>
<sch:include href="./Rules/notice/ISM_ID_00139.sch"/>
<sch:include href="./Rules/notice/ISM_ID_00150.sch"/>
<sch:include href="./Rules/notice/ISM_ID_00151.sch"/>
<sch:include href="./Rules/notice/ISM_ID_00152.sch"/>
<sch:include href="./Rules/notice/ISM_ID_00153.sch"/>
<sch:include href="./Rules/notice/ISM_ID_00156.sch"/>
<sch:include href="./Rules/notice/ISM_ID_00157.sch"/>
<sch:include href="./Rules/notice/ISM_ID_00158.sch"/>
<sch:include href="./Rules/notice/ISM_ID_00159.sch"/>
<sch:include href="./Rules/notice/ISM_ID_00160.sch"/>
<sch:include href="./Rules/notice/ISM_ID_00161.sch"/>

<!--(U) ownerProducer-->
<sch:include href="./Rules/ownerProducer/ISM_ID_00001.sch"/>
<sch:include href="./Rules/ownerProducer/ISM_ID_00099.sch"/>
<sch:include href="./Rules/ownerProducer/ISM_ID_00100.sch"/>

<!--(U) releasableTo-->
<sch:include href="./Rules/releasableTo/ISM_ID_00032.sch"/>
<sch:include href="./Rules/releasableTo/ISM_ID_00041.sch"/>
<sch:include href="./Rules/releasableTo/ISM_ID_00214.sch"/>

<!--(U) resourceElement-->
<sch:include href="./Rules/resourceElement/ISM_ID_00013.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00014.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00056.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00057.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00058.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00059.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00060.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00061.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00062.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00063.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00064.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00065.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00066.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00067.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00068.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00070.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00071.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00072.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00073.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00074.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00075.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00077.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00078.sch"/>
```

```
<sch:include href="./Rules/resourceElement/ISM_ID_00079.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00080.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00081.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00082.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00083.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00084.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00085.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00086.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00087.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00088.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00090.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00104.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00105.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00108.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00109.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00110.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00111.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00112.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00113.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00116.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00118.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00132.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00141.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00145.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00146.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00147.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00149.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00154.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00155.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00162.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00165.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00171.sch"/>
<sch:include href="./Rules/resourceElement/ISM_ID_00172.sch"/>

<!--(U) SARIdentifier-->
<sch:include href="./Rules/SARIdentifier/ISM_ID_00121.sch"/>

<!--(U) SCIcontrols-->
<sch:include href="./Rules/SCIcontrols/ISM_ID_00042.sch"/>
<sch:include href="./Rules/SCIcontrols/ISM_ID_00043.sch"/>
<sch:include href="./Rules/SCIcontrols/ISM_ID_00044.sch"/>
<sch:include href="./Rules/SCIcontrols/ISM_ID_00045.sch"/>
<sch:include href="./Rules/SCIcontrols/ISM_ID_00046.sch"/>
<sch:include href="./Rules/SCIcontrols/ISM_ID_00047.sch"/>
<sch:include href="./Rules/SCIcontrols/ISM_ID_00048.sch"/>
<sch:include href="./Rules/SCIcontrols/ISM_ID_00049.sch"/>
<sch:include href="./Rules/SCIcontrols/ISM_ID_00122.sch"/>
<sch:include href="./Rules/SCIcontrols/ISM_ID_00123.sch"/>
```

```
<sch:include href="./Rules/SCIcontrols/ISM_ID_00177.sch"/>  
<sch:include href="./Rules/SCIcontrols/ISM_ID_00186.sch"/>  
<sch:include href="./Rules/SCIcontrols/ISM_ID_00187.sch"/>  
</sch:schema>  
<!-- UNCLASSIFIED -->
```

FileName:._Properties/capcoResource.sch

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:let xmlns:sch="http://purl.oclc.org/dsdl/schematron" name="ISM_CAPCO_RESOURCE"
value="if(contains($ISM_RESOURCE_ELEMENT/@ism:ownerProducer, 'USA')) then true() else
false()"/>
<!-- Is this a CAPCO applicable resource with the ownerProducer of the resourceElement
being [USA] -->
```

FileName:./_Properties/createDate.sch

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:let xmlns:sch="http://purl.oclc.org/dsdl/schematron" name="ISM_RESOURCE_CREATE_DATE"
value="$ISM_RESOURCE_ELEMENT/@ism:createDate"/>
<!-- Resource Creation Date -->
```

FileName: //_Properties/ismDoD5230_24applies.sch

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:let xmlns:sch="http://purl.oclc.org/dsdl/schematron" name="ISM_DOD5230_24_APPLIES"
value="if(contains($ISM_RESOURCE_ELEMENT/@ism:compliesWith, 'DoD5230.24')) then true()
else false()"/>
<!-- Is this a DoD5230.24 applicable resource -->
```

FileName:._Properties/ismIcd_710applies.sch

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:let xmlns:sch="http://purl.oclc.org/dsdl/schematron" name="ISM_ICD_710_APPLIES"
value="if(contains($ISM_RESOURCE_ELEMENT/@ism:compliesWith, 'ICD-710')) then true() else
false()"/>
<!-- Is this a ICD-710 applicable resource -->
```


FileName:./_Properties/myPosition.sch

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:let xmlns:sch="http://purl.oclc.org/dsdl/schematron" name="myPosition"
value="count(self::node()/preceding::*|self::node()/ancestor::*|self::node()/self::*)" />
<!-- The Position Number of the element that has focus -->
```

FileName: //_Properties/partTags.sch

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:let xmlns:sch="http://purl.oclc.org/dsdl/schematron" name="partTags"
value="//*[(@ism:*) and (not(@ism:excludeFromRollup=true())) and
not(@ism:resourceElement=true())]"/>
<!-- Is this a CAPCO applicable resource that meets ISM-CONTRIBUTES such that it is not
excluded from rollup and is not the resource elemnt. -->
```

FileName: //_Properties/resourceElement.sch

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This is the first ISM-RESOURCE-ELEMENT in the document -->
<sch:let xmlns:sch="http://purl.oclc.org/dsdl/schematron" name="ISM_RESOURCE_ELEMENT"
value="(for $each in (//*) return if($each/@ism:resourceElement=true()) then $each else
null)[1]"/>
<!-- codeDesc
Traverse each element and return a list of all elements containing the attribute
resourceElement with a value of [true]. Then take the first one.
-->
```

Rule: ValidateSortedAttributeValuesAgainstList

FileName:../_Lib/ValidateSortedAttributeValuesAgainstList.sch

Rule Description:

\$rule

Code Description:

\$code

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron"
id="ValidateSortedAttributeValuesAgainstList"
abstract="true">

<sch:rule context="$context">
<!-- Define variables -->
<sch:let name="dataFileElems" value="$curElems"/>
<sch:let name="attrValues" value="$curAttr"/>
<sch:let name="attrValueTokens" value="tokenize($attrValues,' ')/>

<!-- Get the position of each client node relative to its position in the master list. If
the node is not found, return a -1 -->
<sch:let name="orderNums"
value=" for $token in $attrValueTokens return if
($dataFileElems[matches($token,concat('^',text(),'$'))]) then
count(($dataFileElems[matches($token,concat('^',text(),'$'))]/preceding::*) + 1 else -1"/>

<!-- Create a sequence that returns a 0 if the previous sibling has a higher order number,
else return a 1 -->
<sch:let name="sortedOrderNums"
value=" for $peice in distinct-values(for $token in $orderNums return index-of($orderNums,
$token)) return if($peice>1) then if($orderNums[$peice] = $orderNums[$peice - 1]) then
if(compare($attrValueTokens[$peice - 1],$attrValueTokens[$peice]) = 1) then 0 else 2 else
if($orderNums[$peice] > $orderNums[$peice - 1]) then 1 else 0 else 1 "/>
<sch:let name="hasUnsorted" value="count(index-of($sortedOrderNums,0)) > 0"/>
<sch:let name="unsortedValues"
value=" if ($hasUnsorted) then distinct-values( for $token in index-of($sortedOrderNums,0)
return $attrValueTokens[$token] ) else null "/>

<sch:assert test="not($hasUnsorted)" flag="$errFlag_AlphabeticalOrder">
<sch:value-of select="$errMsg_AlphabeticalOrder"/>
The following values are out of order [<sch:value-of select="$unsortedValues"/>] for
[<sch:value-of select="$attrValueTokens"/>] </sch:assert>
</sch:rule>
```

</sch:pattern>

Rule: ValidateValidAttributeValuesAgainstList

FileName:../_Lib/ValidateValidAttributeValuesAgainstList.sch

Rule Description:

\$rule

Code Description:

\$code

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron"
id="ValidateValidAttributeValuesAgainstList"
abstract="true">

<sch:rule context="$context">
<!-- Define variables -->
<sch:let name="dataFileElems" value="$curElems"/>
<sch:let name="attrValues" value="$curAttr"/>
<sch:let name="attrValueTokens" value="tokenize($attrValues,' ')/>
<sch:let name="capco" value="$capcoRestriction"/>

<!-- Get the position of each client node relative to its position in the master list. If
the node is not found, return a -1 -->
<sch:let name="orderNums"
value=" for $token in $attrValueTokens return if
($dataFileElems[matches($token,concat('^',text(),'$'))]) then
count(($dataFileElems[matches($token,concat('^',text(),'$'))])/preceding::*) + 1 else -1"/>

<!-- Determine if the list has invalid values. If and only if it does, figure out which
ones are invalids -->
<sch:let name="hasInvalids" value="count(index-of($orderNums,-1)) > 0"/>
<sch:let name="invalidValues"
value=" if ($hasInvalids) then distinct-values( for $token in index-of($orderNums,-1)
return $attrValueTokens[$token] ) else null "/>

<!-- Determine if the list has duplicate values. If and only if it does, figure out which
ones are duplicates -->
<sch:let name="hasDups"
value="count(distinct-values($attrValueTokens)) != count($attrValueTokens)/>
<sch:let name="dupValues"
value=" if ($hasDups) then distinct-values( for $token in $attrValueTokens return
if (count(index-of($attrValueTokens,$token)) > 1) then $attrValueTokens[index-
of($attrValueTokens,$token)[1]] else null ) else null "/>
```

```
<!-- Execute tests -->
<sch:assert test="if(not($ISM_CAPCO_RESOURCE) and $capco) then 1 else not($hasInvalids)"
flag="$errFlag_ValueNotFound">
<sch:value-of select="$errMsg_ValueNotFound"/>
Invalid value of [<sch:value-of select="$invalidValues"/>]</sch:assert>
<sch:assert test="not($hasDups)" flag="undefined">Duplicate values found [<sch:value-of
select="$dupValues"/>] for [<sch:value-of select="$attrValueTokens"/>] </sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: deprecatedError

FileName:../_Lib/deprecatedError.sch

Rule Description:

\$rule

Code Description:

\$code

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="deprecatedError"
abstract="true">

<!-- Depends on params:
- $rule (rule text)
- $code (english breakdown of the code)
- $context (context for the rule)
- $attrib (name of the attribute being tested)
- $depValues (depreceated values to test for)
- $depDates (related deprecation dates for the depValues)
-->
<sch:rule id="deprecateErr" context="$context">
<sch:let name="attrName" value="'$attrib'"/>
<sch:let name="curDate" value="translate($ISM_RESOURCE_ELEMENT/@ism:createDate,'-','')"/>
<sch:let name="values"
value=" for $each in $depValues return if(index-of(tokenize(./@ism:$attrib,' '),
$each)>0) then $each else null"/>
<sch:let name="reportErr"
value=" for $each in $values return if($curDate >=
translate(subsequence($depDates,index-of($depValues,$each),1),'-',''))
then concat('[',string($each),'] is not authorized for use after ',
string(subsequence($depDates,index-of($depValues,$each),1))) else null "/>

<sch:assert test="count($reportErr)=0" flag="error">
[ISM-ID-<sch:value-of select="'$ismID'"/>][Error] For attribute <sch:value-of
select="$attrName"/>, value(s) <sch:value-of select="$reportErr"/>
</sch:assert>
</sch:rule>
</sch:pattern>
```


Rule: deprecatedWarning

FileName:../_Lib/deprecatedWarning.sch

Rule Description:

\$rule

Code Description:

\$code

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="deprecatedWarning"
abstract="true">

<!-- Depends on params:
- $rule (rule text)
- $code (english breakdown of the code)
- $context (context for the rule)
- $attrib (name of the attribute being tested)
- $depValues (depreceated values to test for)
- $depDates (related deprecation dates for the depValues)
-->
<sch:rule id="deprecateWarn" context="$context">
<sch:let name="attrName" value="'$attrib'"/>
<sch:let name="curDate" value="translate($ISM_RESOURCE_ELEMENT/@ism:createDate,'-','')"/>
<sch:let name="values"
value=" for $each in $depValues return if(index-of(tokenize(./@ism:$attrib,' '),
$each)>0) then $each else null"/>
<sch:let name="reportWarn"
value=" for $each in $values return if($curDate <
translate(subsequence($depDates,index-of($depValues,$each),1),'-','')) then
concat('[',string($each),' ] has been deprecated and is not authorized for used after ',
string(subsequence($depDates,index-of($depValues,$each),1))) else null "/>

<sch:assert test="count($reportWarn)=0" flag="warning">
[ISM-ID-<sch:value-of select="'$ismID'"/>][Warning] For attribute <sch:value-of
select="$attrName"/>, value(s) <sch:value-of select="$reportWarn"/>
</sch:assert>

</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00001

FileName:../Rules/ownerProducer/ISM_ID_00001.sch

Rule Description:

[ISM-ID-00001][Error] The attribute ownerProducer, when it exists, must have a non-null value.

Code Description:

This code makes sure that if ownerProducer is specified that it contains content that is a non-whitespace value.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00001">

<sch:rule context="//*[@ism:ownerProducer]">
<sch:assert id="ism00001" test="normalize-space(./@ism:ownerProducer)!=''" flag="error">
[ISM-ID-00001][Error] The attribute ownerProducer, when it exists, must have
a non-null value.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00002

FileName:../Rules/generalConstraints/ISM_ID_00002.sch

Rule Description:

[ISM-ID-00002][Error] For every optional attribute that is used in a document a non-null value must be present.

Code Description:

This code checks that if an attribute is present and has no value present (string-length = 0) then we return false since all attributes must have a value specified.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00002">

<sch:rule context="//*[@ism:*='' ]">
<sch:assert id="ism00002" test="false()" flag="error">
[ISM-ID-00002][Error] For every optional attribute that is used in a document a non-null
value must be present.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00012

FileName:../Rules/generalConstraints/ISM_ID_00012.sch

Rule Description:

[ISM-ID-00012][Error] If any of the attributes defined in this DES other than DESVersion are specified for an element, then attributes classification and ownerProducer must be specified for the element.

Code Description:

This code triggers on elements that have an ISM attribute whose name is not 'DESVersion' it ensures that both the ownerProducer and classification attributes are present on the element.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00012">

<sch:rule context="//*[@ism:*[not(local-name()='DESVersion')]]">
<sch:assert id="ism00012"
test=" (./@ism:ownerProducer and ./@ism:classification) "
flag="error">
[ISM-ID-00012][Error] If any of the attributes defined in
this DES other than DESVersion are specified for an element, then attributes
classification
and ownerProducer must be specified for the element.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00013

FileName:./Rules/resourceElement/ISM_ID_00013.sch

Rule Description:

[ISM-ID-00013][Error] If ISM-NSI-EO-APPLIES then either attribute classifiedBy or derivedFrom must be specified on the ISM-RESOURCE-ELEMENT. Human Readable: Documents under E.O. 13526 must have classification authority block information.

Code Description:

If the current Classified National Security Information Executive Order does not apply to the document then the rule does not apply and we return true. Otherwise, we make sure that the resourceElement has attribute classifiedBy or derivedFrom specified.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00013">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00013"
test=" if(not($ISM_NSI_EO_APPLIES)) then true() else if($ISM_RESOURCE_ELEMENT/
@ism:classifiedBy or $ISM_RESOURCE_ELEMENT/@ism:derivedFrom) then true() else false() "
flag="error">
[ISM-ID-00013][Error] Documents under E.O. 13526 must have classification authority block
information.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00014

FileName:./Rules/resourceElement/ISM_ID_00014.sch

Rule Description:

[ISM-ID-00014][Error] If ISM-NSI-EO-APPLIES then one or more of the following attributes: declassDate, declassEvent, or declassException must be specified on the ISM-RESOURCE-ELEMENT. Human Readable: Documents under E.O. 13526 must have declassification instructions included in the classification authority block information.

Code Description:

If the current Classified National Security Information Executive Order does not apply to the document then the rule does not apply and we return true. Otherwise, we make sure that the resourceElement has attribute declassDate, declassEvent, or declassException specified.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00014">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00014"
test=" if(not($ISM_NSI_EO_APPLIES)) then true() else if(($ISM_RESOURCE_ELEMENT/
@ism:declassDate or $ISM_RESOURCE_ELEMENT/@ism:declassEvent or $ISM_RESOURCE_ELEMENT/
@ism:declassException )) then true() else false() "
flag="error">
[ISM-ID-00014][Error] If ISM-NSI-EO-APPLIES then one or more of the following
attributes: declassDate, declassEvent, or declassException must be specified on the ISM-
RESOURCE-ELEMENT.

Human Readable: Documents under E.O. 13526 must have declassification instructions
included in the
classification authority block information.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00015

FileName:./Rules/classification/ISM_ID_00015.sch

Rule Description:

[ISM-ID-00015][Error] If ISM-CAPCO-RESOURCE and attribute classification has a value of [U], then attributes releasableTo, SARIdentifier, and SCIcontrols must not be specified.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If it is a CAPCO resource then we check that if we have an element having attribute classification with a value of [U] then we do not have attributes releasealbeTo, SARIdentifier, or SCIcontrols on the same element.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00015">

<sch:rule context="//*[@ism:classification]">
<sch:assert id="ism00015"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(./@ism:classification='U' and (./
@ism:releasableTo or ./@ism:SARIdentifier or ./@ism:SCIcontrols)) then false() else true()
"
flag="error">
[ISM-ID-00015][Error] If ISM-CAPCO-RESOURCE and attribute
classification has a value of [U], then attributes releasableTo, SARIdentifier, and
SCIcontrols
must not be specified.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00016

FileName:./Rules/classification/ISM_ID_00016.sch

Rule Description:

[ISM-ID-00016][Error] If ISM-CAPCO-RESOURCE and attribute classification has a value of [U], then attributes classificationReason, classifiedBy, derivativelyClassifiedBy, declassDate, declassEvent, declassException and derivedFrom must not be specified.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If it is a CAPCO resource then we check that if we have an element having attribute classification with a value of [U] then we do not have attributes classifiedBy, declassDate, declassEvent, declassException, derivativelyClassifiedBy, or derivedFrom on the same element.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00016">

<sch:rule context="//*[@ism:classification]">
<sch:assert id="ism00016"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(matches(./@ism:classification,'^U
$') and (./@ism:classificationReason or ./@ism:classifiedBy or ./@ism:declassDate or ./
@ism:declassEvent or ./@ism:declassException or ./@ism:derivativelyClassifiedBy or ./
@ism:derivedFrom)) then false() else true() "
flag="error">
[ISM-ID-00016][Error] If ISM-CAPCO-RESOURCE and attribute
classification has a value of [U], then attributes classificationReason, classifiedBy,
derivativelyClassifiedBy, declassDate, declassEvent, declassException,
and derivedFrom must not be specified.
</sch:assert>
</sch:rule>
</sch:pattern>
```


Rule: ISM-ID-00017

FileName:./Rules/classifiedBy/ISM_ID_00017.sch

Rule Description:

[ISM-ID-00017][Error] If ISM-NSI-EO-APPLIES and attribute classifiedBy is specified, then attribute classificationReason must be specified. Human Readable: Originally Classified data requires a classification reason be identified.

Code Description:

If current Classified National Security Information Executive Order does not apply to this resource then the rule does not apply and we return true. Otherwise we ensure that any element with the attribute classifiedBy also has the attribute classificationReason.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00017">

<sch:rule context="//*[@ism:classifiedBy]">
<sch:assert id="ism00017"
test=" if(not($ISM_NSI_EO_APPLIES)) then true() else ./@ism:classificationReason "
flag="error">
[ISM-ID-00017][Error] If ISM-NSI-EO-APPLIES and attribute
classifiedBy is specified, then attribute classificationReason must be specified.

Human Readable: Originally Classified data requires a classification reason be identified.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00026

FileName:./Rules/disseminationControls/ISM_ID_00026.sch

Rule Description:

[ISM-ID-00026][Error] If ISM-CAPCO-RESOURCE and attribute disseminationControls is specified, then each of its values must be ordered in accordance with CVEnumISMDissem.xml.

Code Description:

This rule calls upon an abstract pattern to perform the logic work. It makes sure that every value in the disseminationControls attributes are sorted in the order found in the CVEnumISMDissem.xml file.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00026">

<sch:rule context="//*[@ism:disseminationControls]">
<!-- Define variables -->
<sch:let name="capcoRestriction" value="true()"/>
<sch:let name="errFlag_AlphabeticalOrder" value="error"/>
<sch:let name="errMsg_AlphabeticalOrder"
value=" '[ISM-ID-00026][Error] If ISM-CAPCO-RESOURCE and attribute disseminationControls
is specified, then each of its values must be ordered in accordance with
CVEnumISMDissem.xml.' "/>

<sch:let name="dataFileElems" value="$disseminationControlsList"/>
<sch:let name="attrValues" value="./@ism:disseminationControls"/>
<sch:let name="attrValueTokens" value="tokenize($attrValues,' ')/>

<!-- Get the position of each client node relative to its position in the master list. If
the node is not found, return a -1 -->
<sch:let name="orderNums"
value=" for $token in $attrValueTokens return if
($dataFileElems[matches($token,concat('^',text(),'$'))]) then
count(($dataFileElems[matches($token,concat('^',text(),'$'))]/preceding::*) + 1 else -1"/>

<!-- Create a sequence that returns a 0 if the previous sibling has a higher order number,
else return a 1 -->
<sch:let name="sortedOrderNums"
value=" for $peice in distinct-values(for $token in $orderNums return index-of($orderNums,
$token)) return if($peice>1) then if($orderNums[$peice] = $orderNums[$peice - 1]) then
if(compare($attrValueTokens[$peice - 1],$attrValueTokens[$peice]) = 1) then 0 else 2 else
if($orderNums[$peice] > $orderNums[$peice - 1]) then 1 else 0 else 1 "/>
<sch:let name="hasUnsorted" value="count(index-of($sortedOrderNums,0)) > 0"/>
```

```
<sch:let name="unsortedValues"
value=" if ($hasUnsorted) then distinct-values( for $token in index-of($sortedOrderNums,0)
return $attrValueTokens[$token] ) else null "/>

<sch:assert test="not($hasUnsorted)" flag="$errFlag_AlphabeticalOrder">
<sch:value-of select="$errMsg_AlphabeticalOrder"/>
The following values are out of order [<sch:value-of select="$unsortedValues"/>] for
[<sch:value-of select="$attrValueTokens"/>] </sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00028

FileName:../Rules/disseminationControls/ISM_ID_00028.sch

Rule Description:

[ISM-ID-00028][Error] If ISM-CAPCO-RESOURCE and attribute disseminationControls contains the name token [OC], [EYES], or [RELIDO], then attribute classification must have a value of [TS], [S], or [C].

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If it is a CAPCO resource then we check that if we have an element having attribute disseminationControls with a value of [OC], [EYES], or [RELIDO] then we also have the attribute classification specified with a value of [C], [S], or [TS] on the same element.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00028">

<sch:rule context="//*[@ism:disseminationControls]">
<sch:assert id="ism00028"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else count(for $token in tokenize(./
@ism:disseminationControls, ' ') return if(matches($token,'^(OC|EYES|RELIDO)$') and
not( matches(./@ism:classification,'^(TS|S|C)$')) then $token else null )=0 "
flag="error">
[ISM-ID-00028][Error] If ISM-CAPCO-RESOURCE and attribute
disseminationControls contains the name token [OC], [EYES], or [RELIDO],
then attribute classification must have a value of [TS], [S], or [C].
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00030

FileName:../Rules/disseminationControls/ISM_ID_00030.sch

Rule Description:

[ISM-ID-00030][Error] If ISM-CAPCO-RESOURCE and attribute disseminationControls contains the name token [FOUO], then attribute classification must have a value of [U].

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If it is a CAPCO resource then we check that if we have an element having attribute disseminationControls without a value of [FOUO] then we return true because the rule does not apply. Otherwise we make sure the attribute classification is specified with a value of [U] on the same element.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00030">

<sch:rule context="//*[@ism:disseminationControls]">
<sch:assert id="ism00030"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(not(index-of(tokenize(./
@ism:disseminationControls,' '), 'FOUO')>0)) then true() else ./@ism:classification='U'
"
flag="error">
[ISM-ID-00030][Error] If ISM-CAPCO-RESOURCE and attribute
disseminationControls contains the name token [FOUO], then attribute classification must
have
a value of [U].
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00031

FileName:./Rules/disseminationControls/ISM_ID_00031.sch

Rule Description:

[ISM-ID-00031][Error] If ISM-CAPCO-RESOURCE and attribute disseminationControls contains the name token [REL] or [EYES], then attribute releasableTo must be specified.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If it is a CAPCO resource then we check that if we have an element having attribute disseminationControls with a value of [REL] or [EYES] then the attribute releasableTo is specified and does not have an empty value set.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00031">

<sch:rule context="//*[@ism:disseminationControls]">
<sch:assert id="ism00031"
test=" count(for $token in tokenize(./@ism:disseminationControls, ' '))
return if(matches($token,'^(REL|EYES)$') and (not(./@ism:releasableTo) or ./
@ism:releasableTo='') ) then $token else null )=0 "
flag="error">
[ISM-ID-00031][Error] If ISM-CAPCO-RESOURCE and attribute
disseminationControls contains the name token [REL] or [EYES], then attribute releasableTo
must be specified.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00032

FileName:./Rules/releasableTo/ISM_ID_00032.sch

Rule Description:

[ISM-ID-00032][Error] If ISM-CAPCO-RESOURCE and attribute disseminationControls is not specified or is specified and does not contain the name token [REL] or [EYES], then attribute releasableTo must not be specified.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If the attribute releasableTo is specified, then we make sure that the attribute disseminationControls is specified with a value containing [EYES] or [REL].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00032">

<sch:rule context="//*[@ism:releasableTo]">

<sch:let name="dissemTok" value="tokenize(./@ism:disseminationControls,' ')" />
<sch:assert id="ism00032"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else index-of($dissemTok,'EYES')>0 or
index-of($dissemTok,'REL')>0 "
flag="error">
[ISM-ID-00032][Error] If ISM-CAPCO-RESOURCE and attribute
disseminationControls is not specified or is specified and does not contain the name token
[REL] or [EYES], then attribute releasableTo must not be specified.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00033

FileName:../Rules/disseminationControls/ISM_ID_00033.sch

Rule Description:

[ISM-ID-00033][Error] If ISM-CAPCO-RESOURCE, then Name tokens [REL], [EYES] and [NF] are mutually exclusive for attribute disseminationControls.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If it is a CAPCO resource then we check that if we have an element having attribute disseminationControls and counting 1 for each value of [REL], [NF] or [EYES] found. If the count is greater than one then the values are not being used exclusively with respect to each other

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00033">

<sch:rule context="//*[@ism:disseminationControls]">
<sch:let name="dissemTok" value="tokenize(./@ism:disseminationControls,' ')" />
<sch:assert id="ism00033"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else not( count(( if(index-
of($dissemTok,'REL')>0) then 1 else null, if(index-of($dissemTok,'NF')>0) then 1
else null, if(index-of($dissemTok,'EYES')>0) then 1 else null) ) > 1 ) "
flag="error">
[ISM-ID-00033][Error] If ISM-CAPCO-RESOURCE, then
tokens [REL], [EYES] and [NF] are mutually exclusive for attribute disseminationControls.
</sch:assert>
</sch:rule>
</sch:pattern>
```


Rule: ISM-ID-00034

FileName:../Rules/disseminationControls/ISM_ID_00034.sch

Rule Description:

[ISM-ID-00034][Error] If ISM-CAPCO-RESOURCE, then Name tokens "RELIDO" and "NF" are mutually exclusive for attribute disseminationControls.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If it is a CAPCO resource then we check that if we have an element having attribute disseminationControls that does not have values [RELIDO] and [NF] at the same time.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00034">

<sch:rule context="//*[@ism:disseminationControls]">
<sch:let name="dissemTok" value="tokenize(./@ism:disseminationControls,' ')" />
<sch:assert id="ism00035"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else not(index-
of($dissemTok,'RELIDO')>0 and index-of($dissemTok,'NF')>0) "
flag="error">
[ISM-ID-00034][Error] If ISM-CAPCO-RESOURCE, then Name
tokens "RELIDO" and "NF" are mutually exclusive for attribute disseminationControls.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00035

FileName:./Rules/nonICmarkings/ISM_ID_00035.sch

Rule Description:

[ISM-ID-00035][Error] If ISM-CAPCO-RESOURCE and attribute nonICmarkings is specified, then each of its values must be ordered in accordance with CVENumISMNonIC.xml.

Code Description:

This rule calls upon an abstract pattern to perform the logic work. It makes sure that every value in the disseminationControls attributes are sorted in the order found in the CVENumISMNonIC.xml file.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00035">

<sch:rule context="//*[@ism:nonICmarkings]">
<!-- Define variables -->
<sch:let name="errFlag_AlphabeticalOrder" value="error"/>
<sch:let name="errMsg_AlphabeticalOrder"
value="' [ISM-ID-00035][Error] If ISM-CAPCO-RESOURCE and attribute nonICmarkings is
specified, then each of its values must be ordered in accordance with CVENumISMNonIC.xml.
' "/>

<sch:let name="dataFileElems" value="$nonICmarkingsList"/>
<sch:let name="attrValues" value="./@ism:nonICmarkings"/>
<sch:let name="attrValueTokens" value="tokenize($attrValues, ' ' )"/>

<!-- Get the position of each client node relative to its position in the master list. If
the node is not found, return a -1 -->
<sch:let name="orderNums"
value=" for $token in $attrValueTokens return if
($dataFileElems[matches($token,concat('^',text(),'$'))]) then
count(($dataFileElems[matches($token,concat('^',text(),'$'))])/preceding::*) + 1 else -1"/>

<!-- Create a sequence that returns a 0 if the previous sibling has a higher order number,
else return a 1 -->
<sch:let name="sortedOrderNums"
value=" for $peice in distinct-values(for $token in $orderNums return index-of($orderNums,
$token)) return if($peice>1) then if($orderNums[$peice] = $orderNums[$peice - 1]) then
if(compare($attrValueTokens[$peice - 1],$attrValueTokens[$peice]) = 1) then 0 else 2 else
if($orderNums[$peice] > $orderNums[$peice - 1]) then 1 else 0 else 1 "/>
<sch:let name="hasUnsorted" value="count(index-of($sortedOrderNums,0)) > 0"/>
<sch:let name="unsortedValues"
```

```
value=" if ($hasUnsorted) then distinct-values( for $token in index-of($sortedOrderNums,0)
return $attrValueTokens[$token] ) else null "/>

<sch:assert test="not($hasUnsorted)" flag="$errFlag_AlphabeticalOrder">
<sch:value-of select="$errMsg_AlphabeticalOrder"/>
The following values are out of order [<sch:value-of select="$unsortedValues"/>] for
[<sch:value-of select="$attrValueTokens"/>] </sch:assert>
</sch:rule>

</sch:pattern>
```

Rule: ISM-ID-00036

FileName:./Rules/nonICmarkings/ISM_ID_00036.sch

Rule Description:

[ISM-ID-00036][Error] If ISM-CAPCO-RESOURCE and attribute nonICmarkings contains the name token [SC], then attribute classification must have a value of [TS], [S], or [C].

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If it is a CAPCO resource then we check if the attribute disseminationControls contains the value [SC] and if it does we check that the classification attribute has a value of [C], [S], or [TS].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00036">

<sch:rule context="//*[@ism:nonICmarkings]">
<sch:assert id="ism00036"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(index-of(tokenize(./
@ism:nonICmarkings, ' '), 'SC')>0) then matches(./@ism:classification, '^(TS|S|C)$') else
true() "
flag="error">
[ISM-ID-00036][Error] If ISM-CAPCO-RESOURCE and attribute nonICmarkings
contains the name token [SC], then attribute classification must have a
value of [TS], [S], or [C].
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00037

FileName:./Rules/nonICmarkings/ISM_ID_00037.sch

Rule Description:

[ISM-ID-00037][Error] If ISM-CAPCO-RESOURCE and attribute nonICmarkings contains the name token [SINFO], [SBU], or [SBU-NF], then attribute classification must have a value of [U].

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If it is a CAPCO resource then we check if the attribute nonICmarkings contains a value of [SINFO], [SBU], or [SBU-NF] then the classification attribute must have a value of [U].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00037">

<sch:rule context="//*[@ism:nonICmarkings]">
<sch:let name="nonICtok" value="tokenize(./@ism:nonICmarkings, ' ')" />
<sch:assert id="ism00037"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(index-of($nonICtok, 'SINFO')>0
or index-of($nonICtok, 'SBU')>0 or index-of($nonICtok, 'SBU-NF')>0) then ./
@ism:classification='U' else true() "
flag="error">
[ISM-ID-00037][Error] If ISM-CAPCO-RESOURCE and attribute nonICmarkings contains
the name token [SINFO], [SBU], or [SBU-NF], then attribute classification must have a
value of [U].
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00038

FileName:./Rules/nonICmarkings/ISM_ID_00038.sch

Rule Description:

[ISM-ID-00038][Error] If ISM-CAPCO-RESOURCE, then Name tokens [XD] and [ND] are mutually exclusive for attribute nonICmarkings.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If it is a CAPCO resource then we check that the attribute nonICmarkings does not contain both a value of [XD] and a value of [ND].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00038">

<sch:rule context="//*[@ism:nonICmarkings]">
<!-- get list of tokens in nonICmarkings attribute -->
<sch:let name="nicmTok" value="tokenize(./@ism:nonICmarkings,' ')" />

<sch:assert id="ism00038"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else not(index-of($nicmTok,'XD')>0 and
index-of($nicmTok,'ND')>0) "
flag="error">
[ISM-ID-00038][Error] If ISM-CAPCO-RESOURCE, then Name tokens [XD] and [ND] are mutually
exclusive for attribute nonICmarkings.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00040

FileName:./Rules/classification/ISM_ID_00040.sch

Rule Description:

[ISM-ID-00040][Error] If ISM-CAPCO-RESOURCE and attribute ownerProducer contains [USA] then attribute classification must have a value in CVEnumISMClassificationUS.xml.

Code Description:

This rule calls upon an abstract pattern to perform the logic work. It makes sure that every value in the classification attributes are a value found in the CVEnumISMClassificationUS.xml file

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00040">

<sch:rule context="//*[@ism:classification]">
<!-- Define variables -->
<sch:let name="errFlag_ValueNotFound" value="error"/>
<sch:let name="errMsg_ValueNotFound"
value="' [ISM-ID-00040][Error] If ISM-CAPCO-RESOURCE and attribute ownerProducer contains
[USA] then attribute classification must have a value in CVEnumISMClassificationUS.xml.'
"/>

<sch:let name="dataFileElems" value="$classificationUSList"/>
<sch:let name="attrValues" value="./@ism:classification"/>
<sch:let name="attrValueTokens" value="tokenize($attrValues,' ')/>
<sch:let name="capco"
value="if(contains(./@ism:ownerProducer, 'USA')) then true() else false()"/>

<!-- Get the position of each client node relative to its position in the master list. If
the node is not found, return a -1 -->
<sch:let name="orderNums"
value=" for $token in $attrValueTokens return if
($dataFileElems[matches($token,concat('^',text(),'$'))]) then
count(($dataFileElems[matches($token,concat('^',text(),'$'))])/preceding::* ) + 1 else -1"/>

<!-- Determine if the list has invalid values. If and only if it does, figure out which
ones are invalids -->
<sch:let name="hasInvalids" value="count(index-of($orderNums,-1)) > 0"/>
<sch:let name="invalidValues"
value=" if ($hasInvalids) then distinct-values( for $token in index-of($orderNums,-1)
return $attrValueTokens[$token] ) else null "/>
```

```

<!-- Determine if the list has duplicate values. If and only if it does, figure out which
ones are duplicates -->
<sch:let name="hasDups"
value="count(distinct-values($attrValueTokens)) != count($attrValueTokens)"/>
<sch:let name="dupValues"
value=" if ($hasDups) then distinct-values( for $token in $attrValueTokens return
if (count(index-of($attrValueTokens,$token)) > 1) then $attrValueTokens[index-
of($attrValueTokens,$token)[1]] else null ) else null "/>

<!-- Execute tests -->
<sch:assert test="if(($ISM_CAPCO_RESOURCE) and not($capco)) then true() else
not($hasInvalids)"
flag="$errFlag_ValueNotFound">
<sch:value-of select="$errMsg_ValueNotFound"/>
Invalid value of [<sch:value-of select="$invalidValues"/>]</sch:assert>
<sch:assert test="not($hasDups)" flag="undefined">Duplicate values found [<sch:value-of
select="$dupValues"/>] for [<sch:value-of select="$attrValueTokens"/>] </sch:assert>
</sch:rule>
</sch:pattern>

```


Rule: ISM-ID-00041

FileName:./Rules/releasableTo/ISM_ID_00041.sch

Rule Description:

[ISM-ID-00041][Error] If ISM-CAPCO-RESOURCE and attribute releasableTo is specified, then each of its values must be ordered in accordance with CVENumISMRelTo.xml."

Code Description:

This rule calls upon an abstract pattern to perform the logic work. It makes sure that every value in the releasableTo attributes are sorted in the order found in the CVENumISMRelTo.xml file.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00041">

<sch:rule context="//*[@ism:releasableTo]">
<!-- Define variables -->
<sch:let name="errFlag_AlphabeticalOrder" value="error"/>
<sch:let name="errMsg_AlphabeticalOrder"
value="' [ISM-ID-00041][Error] If ISM-CAPCO-RESOURCE and attribute releasableTo is
specified, then each of its values must be ordered in accordance with CVENumISMRelTo.xml.
' "/>

<sch:let name="dataFileElems" value="$releasableToList"/>
<sch:let name="attrValues" value="./@ism:releasableTo"/>
<sch:let name="attrValueTokens" value="tokenize($attrValues, ' ' )"/>

<!-- Get the position of each client node relative to its position in the master list. If
the node is not found, return a -1 -->
<sch:let name="orderNums"
value=" for $token in $attrValueTokens return if
($dataFileElems[matches($token,concat('^',text(),'$'))]) then
count(($dataFileElems[matches($token,concat('^',text(),'$'))])/preceding::*) + 1 else -1"/>

<!-- Create a sequence that returns a 0 if the previous sibling has a higher order number,
else return a 1 -->
<sch:let name="sortedOrderNums"
value=" for $peice in distinct-values(for $token in $orderNums return index-of($orderNums,
$token)) return if($peice>1) then if($orderNums[$peice] = $orderNums[$peice - 1]) then
if(compare($attrValueTokens[$peice - 1],$attrValueTokens[$peice]) = 1) then 0 else 2 else
if($orderNums[$peice] > $orderNums[$peice - 1]) then 1 else 0 else 1 "/>
<sch:let name="hasUnsorted" value="count(index-of($sortedOrderNums,0)) > 0"/>
<sch:let name="unsortedValues"
```

```
value=" if ($hasUnsorted) then distinct-values( for $token in index-of($sortedOrderNums,0)
return $attrValueTokens[$token] ) else null "/>

<sch:assert test="not($hasUnsorted)" flag="$errFlag_AlphabeticalOrder">
<sch:value-of select="$errMsg_AlphabeticalOrder"/>
The following values are out of order [<sch:value-of select="$unsortedValues"/>] for
[<sch:value-of select="$attrValueTokens"/>] </sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00042

FileName:./Rules/SCIcontrols/ISM_ID_00042.sch

Rule Description:

[ISM-ID-00042][Error] If ISM-CAPCO-RESOURCE and attribute SCIcontrols is specified, each of its values must be ordered in accordance with CVENumISMSCIControls.xml.

Code Description:

This rule calls upon an abstract pattern to perform the logic work. It makes sure that every value in the SCIcontrols attributes are sorted in the order found in the CVENumISMSCIControls.xml file.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00042">

<sch:rule context="//*[@ism:SCIcontrols]">
<!-- Define variables -->
<sch:let name="errFlag_AlphabeticalOrder" value="error"/>
<sch:let name="errMsg_AlphabeticalOrder"
value="' [ISM-ID-00042][Error] If ISM-CAPCO-RESOURCE and attribute SCIcontrols is
specified, each of its values must be ordered in accordance with CVENumISMSCIControls.xml.
' "/>

<sch:let name="dataFileElems" value="$SCIcontrolsList"/>
<sch:let name="attrValues" value="./@ism:SCIcontrols"/>
<sch:let name="attrValueTokens" value="tokenize($attrValues, ' ' )"/>

<!-- Get the position of each client node relative to its position in the master list. If
the node is not found, return a -1 -->
<sch:let name="orderNums"
value=" for $token in $attrValueTokens return if
($dataFileElems[matches($token,concat('^',text(),'$'))]) then
count(($dataFileElems[matches($token,concat('^',text(),'$'))]/preceding::*) + 1 else -1"/>

<!-- Create a sequence that returns a 0 if the previous sibling has a higher order number,
else return a 1 -->
<sch:let name="sortedOrderNums"
value=" for $peice in distinct-values(for $token in $orderNums return index-of($orderNums,
$token)) return if($peice>1) then if($orderNums[$peice] = $orderNums[$peice - 1]) then
if(compare($attrValueTokens[$peice - 1],$attrValueTokens[$peice]) = 1) then 0 else 2 else
if($orderNums[$peice] > $orderNums[$peice - 1]) then 1 else 0 else 1 "/>
<sch:let name="hasUnsorted" value="count(index-of($sortedOrderNums,0)) > 0"/>
<sch:let name="unsortedValues"
value=" if ($hasUnsorted) then distinct-values( for $token in index-of($sortedOrderNums,0)
return $attrValueTokens[$token] ) else null "/>
```

```
<sch:assert test="not($hasUnsorted)" flag="$errFlag_AlphabeticalOrder">
<sch:value-of select="$errMsg_AlphabeticalOrder"/>
The following values are out of order [<sch:value-of select="$unsortedValues"/>] for
[<sch:value-of select="$attrValueTokens"/>] </sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00043

FileName:./Rules/SCIcontrols/ISM_ID_00043.sch

Rule Description:

[ISM-ID-00043][Error] If ISM-CAPCO-RESOURCE and attribute SCIcontrols contains the name token [SI], then attribute classification must have a value of [TS], [S], or [C].

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If the current element has attribute SCIcontrols specified with a value containing [SI], then we make sure that attribute classification has a value of [TS], [S], or [C].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00043">

<sch:rule context="//*[@ism:SCIcontrols]">
<sch:assert id="ism00043"
test="if(not($ISM_CAPCO_RESOURCE)) then true() else if ( index-of(tokenize(./
@ism:SCIcontrols, ' '), 'SI')>0 and not( matches(./@ism:classification, '^([TS|S|C])$' ) )
then false() else true() "
flag="error">
[ISM-ID-00043][Error] If ISM-CAPCO-RESOURCE and attribute SCIcontrols contains the name
token [SI], then attribute
classification must have a value of [TS], [S], or [C].
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00044

FileName:./Rules/SCIcontrols/ISM_ID_00044.sch

Rule Description:

[ISM-ID-00044][Error] If ISM-CAPCO-RESOURCE and attribute SCIcontrols contains the name token [SI-G], then attribute classification must have a value of [TS].

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If the current element has attribute SCIcontrols specified with a value containing [SI-G], then we make sure that attribute classification has a value of [TS].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00044">

<sch:rule context="//*[@ism:SCIcontrols]">
<sch:assert id="ism00045"
test="if(not($ISM_CAPCO_RESOURCE)) then true() else if(index-of(tokenize(./
@ism:SCIcontrols,' '), 'SI-G')>0 and not(matches(./@ism:classification, '^TS$')) ) then
false() else true() "
flag="error">
[ISM-ID-00044][Error] If ISM-CAPCO-RESOURCE and attribute SCIcontrols contains the name
token [SI-G], then attribute
classification must have a value of [TS].
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00045

FileName:./Rules/SCIcontrols/ISM_ID_00045.sch

Rule Description:

[ISM-ID-00045][Error] If ISM-CAPCO-RESOURCE and attribute SCIcontrols contains the name token [SI-G], then attribute disseminationControls must contain the name token [OC].

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If the current element has attribute SCIcontrols specified with a value containing [SI-G], then we make sure that attribute disseminationControls contains the value [OC].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00045">

<sch:rule context="//*[@ism:SCIcontrols]">
<sch:assert id="ism00045"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(index-of(tokenize(./
@ism:SCIcontrols,' '), 'SI-G')>0) then index-of(tokenize(./@ism:disseminationControls, '
'), 'OC')>0 else true() "
flag="error">
[ISM-ID-00045][Error] If ISM-CAPCO-RESOURCE and attribute SCIcontrols contains the name
token [SI-G], then attribute
disseminationControls must contain the name token [OC].
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00046

FileName:./Rules/SCIcontrols/ISM_ID_00046.sch

Rule Description:

[ISM-ID-00046][Error] If ISM-CAPCO-RESOURCE and attribute SCIcontrols contains a name token starting with [SI-ECI], then attribute classification must have a value of [TS].

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If the current element has attribute classification specified with a value of [TS] then the rule does not apply and we return true. Otherwise, we make sure that attribute SCIcontrols does not contain the value [SI-ECI].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00046">

<sch:rule context="//*[@ism:SCIcontrols]">
<sch:assert id="ism00046"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(matches(./@ism:classification,'^TS
$')) then true() else count( for $token in tokenize(./@ism:SCIcontrols,' ') return
if(matches($token,'^SI-ECI')) then 1 else null )=0 "
flag="error">
[ISM-ID-00046][Error] If ISM-CAPCO-RESOURCE and attribute SCIcontrols contains
a name token starting with [SI-ECI], then attribute classification must have a
value of [TS].
</sch:assert>
</sch:rule>
</sch:pattern>
```


Rule: ISM-ID-00047

FileName:./Rules/SCIcontrols/ISM_ID_00047.sch

Rule Description:

[ISM-ID-00047][Error] If ISM-CAPCO-RESOURCE and attribute SCIcontrols contains the name token [TK], then attribute classification must have a value of [TS] or [S].

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If the current element has attribute SCIcontrols specified with a value containing [TK], then we make sure that attribute classification has a value of [TS], or [S].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00047">

<sch:rule context="//*[@ism:SCIcontrols]">
<sch:assert id="ism00047"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(index-of(tokenize(./
@ism:SCIcontrols, ' '), 'TK')>0 and not( matches(./@ism:classification, '^(TS|S)$')) )
then false() else true() "
flag="error">
[ISM-ID-00047][Error] If ISM-CAPCO-RESOURCE and attribute SCIcontrols contains
the name token [TK], then attribute classification must have a value of [TS] or [S].
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00048

FileName:./Rules/SCIcontrols/ISM_ID_00048.sch

Rule Description:

[ISM-ID-00048][Error] If ISM-CAPCO-RESOURCE and attribute SCIcontrols contains the name token [HCS], then attribute classification must have a value of [TS], [S], or [C].

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If the current element has attribute SCIcontrols specified with a value containing [HCS], then we make sure that attribute classification has a value of [TS], [S], or [C].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00048">

<sch:rule context="//*[@ism:SCIcontrols]">
<sch:assert id="ism00048"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if( index-of(tokenize(./
@ism:SCIcontrols, ' '), 'HCS')>0 and not(matches(./@ism:classification, '^(TS|S|C)$')) )
then false() else true() "
flag="error">
[ISM-ID-00048][Error] If ISM-CAPCO-RESOURCE and attribute SCIcontrols contains the name
token [HCS], then attribute
classification must have a value of [TS], [S], or [C].
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00049

FileName:./Rules/SCIcontrols/ISM_ID_00049.sch

Rule Description:

[ISM-ID-00049][Error] If ISM-CAPCO-RESOURCE and attribute SCIcontrols contains the name token [HCS], then attribute disseminationControls must contain the name token [NF].

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If the current element has attribute SCIcontrols specified with a value containing [HCS], then we make sure that attribute disseminationControls contains the value [NF].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00049">

<sch:rule context="//*[@ism:SCIcontrols]">
<sch:assert id="ism00048"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else not( index-
of(tokenize(./@ism:SCIcontrols, ' '), 'HCS')>0 and not(index-of(tokenize(./
@ism:disseminationControls, ' '), 'NF')) ) "
flag="error">
[ISM-ID-00049][Error] If ISM-CAPCO-RESOURCE and attribute SCIcontrols contains the name
token [HCS], then attribute
disseminationControls must contain the name token [NF].
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00056

FileName:../Rules/resourceElement/ISM_ID_00056.sch

Rule Description:

[ISM-ID-00056][Error] If ISM-CAPCO-RESOURCE and attribute classification of ISM-RESOURCE-ELEMENT has a value of [U] then no element meeting ISM-CONTRIBUTES-USA in the document may have a classification attribute of [C], [S] or [TS]. Human Readable: USA UNCLASSIFIED documents can't have TS, S, or C data.

Code Description:

If the current element is not the resourceElement then the rule does not apply and we return true. If CAPCO rules do not apply to the document then the rule does not apply and we return true. If the resourceElement has attribute classification specified with a value of [U], then we make sure that no portion with ownerProducer containing USA has attribute classification specified with a value of [C], [S], or [TS].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00056">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:include href="../../../_Properties/myPosition.sch"/>
<sch:assert id="ism00056"
test=" if(not($myPosition = $resourceElementPosition)) then true() else
if(not($ISM_CAPCO_RESOURCE)) then true() else if(not($ISM_RESOURCE_ELEMENT/
@ism:classification='U')) then true() else count( for $each in $partTags return
if(contains($each/@ism:ownerProducer, 'USA') and not($each/@ism:classification='U')) then
$each else null )=0 "
flag="error">
[ISM-ID-00056][Error] USA unclassified documents can't have TS, S, or C data.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00057

FileName:./Rules/resourceElement/ISM_ID_00057.sch

Rule Description:

[ISM-ID-00057][Error] If ISM-CAPCO-RESOURCE and attribute classification of ISM-RESOURCE-ELEMENT has a value of [U] then no element meeting ISM-CONTRIBUTES in the document may have a classification attribute of [R]. Human Readable: USA UNCLASSIFIED documents cannot have R data.

Code Description:

If the current element is not the resourceElement then the rule does not apply and we return true. If CAPCO rules do not apply to the document then the rule does not apply and we return true. If the resourceElement has attribute classification specified with a value of [U], then we make sure that no portion has attribute classification specified with a value of [R].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00057">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:include href="../../../_Properties/myPosition.sch"/>
<sch:assert id="ism00057"
test=" if(not($myPosition = $resourceElementPosition)) then true() else
if(not($ISM_CAPCO_RESOURCE)) then true() else if(not($ISM_RESOURCE_ELEMENT/
@ism:classification='U')) then true() else not(index-of($partClassification_tok,
'R')>0) "
flag="error">
[ISM-ID-00057][Error] USA unclassified documents cannot have R data.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00058

FileName:../Rules/resourceElement/ISM_ID_00058.sch

Rule Description:

[ISM-ID-00058][Error] If ISM-CAPCO-RESOURCE and attribute classification of ISM-RESOURCE-ELEMENT has a value of [C] then no element meeting ISM-CONTRIBUTES-USA in the document may have a classification attribute of [S] or [TS]. Human Readable: USA CONFIDENTIAL documents can't have TS or S data.

Code Description:

If the current element is not the resourceElement then the rule does not apply and we return true. If CAPCO rules do not apply to the document then the rule does not apply and we return true. If the resourceElement has attribute classification specified with a value of [C], then we make sure that no portion with ownerProducer containing USA has attribute classification specified with a value of [TS] or [S].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00058">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:include href="../../../_Properties/myPosition.sch"/>
<sch:assert id="ism00058"
test=" if(not($myPosition = $resourceElementPosition)) then true() else
if(not($ISM_CAPCO_RESOURCE)) then true() else if(not($ISM_RESOURCE_ELEMENT/
@ism:classification='C')) then true() else count( for $each in $partTags return
if(contains($each/@ism:ownerProducer, 'USA') and not($each/@ism:classification='U' or
$each/@ism:classification='C') ) then $each else null )=0 "
flag="error">
[ISM-ID-00058][Error] USA confidential documents can't have TS or S data.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00059

FileName:./Rules/resourceElement/ISM_ID_00059.sch

Rule Description:

[ISM-ID-00059][Error] If ISM-CAPCO-RESOURCE and attribute classification of ISM-RESOURCE-ELEMENT has a value of [S] then no element meeting ISM-CONTRIBUTES-USA in the document may have a classification attribute of [TS].

Human Readable: USA SECRET documents can't have TS data.

Code Description:

If the current element is not the resourceElement then the rule does not apply and we return true. If CAPCO rules do not apply to the document then the rule does not apply and we return true. If the resourceElement has attribute classification specified with a value of [S], then we make sure that no portion with ownerProducer containing USA has attribute classification specified with a value of [TS].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00059">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:include href="../../../_Properties/myPosition.sch"/>
<sch:assert id="ism00059"
test=" if(not($myPosition = $resourceElementPosition)) then true() else
if(not($ISM_CAPCO_RESOURCE)) then true() else if(not($ISM_RESOURCE_ELEMENT/
@ism:classification='S')) then true() else count( for $each in $partTags return
if(contains($each/@ism:ownerProducer, 'USA') and not($each/@ism:classification='U' or
$each/@ism:classification='C' or $each/@ism:classification='S')) then $each else null )=0
"
flag="error">
[ISM-ID-00059][Error] USA secret documents can't have TS data.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00060

FileName:./Rules/resourceElement/ISM_ID_00060.sch

Rule Description:

[ISM-ID-00060][Error] If ISM-CAPCO-RESOURCE and any element meeting ISM-CONTRIBUTES in the document have the attribute SCIcontrols containing a value of [SI] then the ISM-RESOURCE-ELEMENT element's SCIcontrols must contain [SI]. Human Readable: USA documents having SI must have it at the resource level.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If the current element has attribute SCIcontrols specified with a value containing [SI] and does not have attribute excludeFromRollup set to true, then we make sure that the resourceElement has attribute SCIcontrols specified with a value containing [SI].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00060">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00060"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(index-of($partSCIcontrols_tok,
'SI') > 0) then index-of(tokenize($ISM_RESOURCE_ELEMENT/@ism:SCIcontrols,' '), 'SI')
> 0 else true() "
flag="error">
[ISM-ID-00060][Error] USA documents having SI must have it at the resource level.
</sch:assert>
</sch:rule>
</sch:pattern>
```


Rule: ISM-ID-00061

FileName:./Rules/resourceElement/ISM_ID_00061.sch

Rule Description:

[ISM-ID-00061][Error] If ISM-CAPCO-RESOURCE and any element meeting ISM-CONTRIBUTES in the document have the attribute SClcontrols containing a value of [SI-G] then the ISM-RESOURCE-ELEMENT element's SClcontrols must contain [SI-G]. Human Readable: USA documents having SI-G must have it at the resource level.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If the current element has attribute SClcontrols specified with a value containing [SI-G] and does not have attribute excludeFromRollup set to true, then we make sure that the resourceElement has attribute SClcontrols specified with a value containing [SI-G].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00061">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00061"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(index-of($partSClcontrols_tok,
'SI-G') &gt; 0) then index-of($bannerSClcontrols_tok, 'SI-G') &gt; 0 else true() "
flag="error">
[ISM-ID-00061][Error] USA documents having SI-G must have it at the resource level.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00062

FileName:./Rules/resourceElement/ISM_ID_00062.sch

Rule Description:

[ISM-ID-00062][Error] If ISM-CAPCO-RESOURCE and any element meeting ISM-CONTRIBUTES in the document have the attribute SClcontrols containing a value of [TK] then the ISM-RESOURCE-ELEMENT element's SClcontrols must contain [TK]. Human Readable: USA documents having TK must have it at the resource level.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If the current element has attribute SClcontrols specified with a value containing [TK] and does not have attribute excludeFromRollup set to true, then we make sure that the resourceElement has attribute SClcontrols specified with a value containing [TK].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00062">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00062"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(index-of($partSClcontrols_tok,
'TK') > 0) then index-of($bannerSClcontrols_tok, 'TK') > 0 else true() "
flag="error">
[ISM-ID-00062][Error] USA documents having TK must have it at the resource level.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00063

FileName:./Rules/resourceElement/ISM_ID_00063.sch

Rule Description:

[ISM-ID-00063][Error] If ISM-CAPCO-RESOURCE and any element meeting ISM-CONTRIBUTES in the document have the attribute SCIconcontrols containing a value of [HCS] then the ISM-RESOURCE-ELEMENT element's SCIconcontrols must contain [HCS]. Human Readable: USA documents having HCS must have it at the resource level.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If the current element has attribute SCIconcontrols specified with a value containing [HCS] and does not have attribute excludeFromRollup set to true, then we make sure that the resourceElement has attribute SCIconcontrols specified with a value containing [HCS].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00063">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00063"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(index-of($partSCIconcontrols_tok,
'HCS') > 0) then index-of($bannerSCIconcontrols_tok, 'HCS') > 0 else true() "
flag="error">
[ISM-ID-00063][Error] USA documents having HCS must have it at the resource level.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00064

FileName:../Rules/resourceElement/ISM_ID_00064.sch

Rule Description:

[ISM-ID-00064][Error] If ISM-CAPCO-RESOURCE and any element meeting ISM-CONTRIBUTES in the document have the attribute FGIsorceOpen containing any value then the ISM-RESOURCE-ELEMENT must have either FGIsorceOpen or FGIsorceProtected with a value. Human Readable: USA documents having FGI Open data must have FGI Open or FGI Protected at the resource level.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If the current element has attribute FGIsorceOpen specified and does not have attribute excludeFromRollup set to true, then we make sure that the resourceElement has one of the following attributes specified: FGIsorceOpen or FGIsorceProtected.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00064">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00064"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(not(empty($partFGIsorceOpen)))
then ($bannerFGIsorceOpen or $bannerFGIsorceProtected) else true() "
flag="error">
[ISM-ID-00064][Error] USA documents having FGI Open data must have FGI Open or FGI
Protected at the resource level.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00065

FileName:./Rules/resourceElement/ISM_ID_00065.sch

Rule Description:

[ISM-ID-00065][Error] If ISM-CAPCO-RESOURCE and any element meeting ISM-CONTRIBUTES in the document have the attribute FGIsorceProtected containing any value then the ISM-RESOURCE-ELEMENT must have FGIsorceProtected with a value. Human Readable: USA documents having FGI Protected data must have FGI Protected at the resource level.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If any element has attribute FGIsorceProtected specified with a non-empty value and does not have attribute excludeFromRollup set to true, then we make sure that the banner has attribute FGIsorceProtected specified with a non-empty value.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00065">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00065"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else
if(not(empty($partFGIsorceProtected))) then $bannerFGIsorceProtected else true() "
flag="error">
[ISM-ID-00065][Error] USA documents having FGI Protected data must have FGI Protected at
the resource level.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00066

FileName:./Rules/resourceElement/ISM_ID_00066.sch

Rule Description:

[ISM-ID-00066][Error] If ISM-CAPCO-RESOURCE and: 1. Any element meeting ISM-CONTRIBUTES in the document has the attribute disseminationControls containing [FOUO] AND 2. ISM-RESOURCE-ELEMENT has the attribute classification [U] AND 3. No element meeting ISM-CONTRIBUTES in the document has nonICmarkings containing [SBU], [SBU-NF], [LES], [LES-NF] Then the ISM-RESOURCE-ELEMENT must have disseminationControls containing [FOUO].
Human Readable: USA Unclassified documents having FOUO data and not having SBU, SBU-NF, LES, or LES-NF must have FOUO at the resource level.

Code Description:

If CAPCO rules do not apply to the document, then the rule does not apply and we return true. If the resourceElement has attribute classification specified with a value other than [U], then the rule does not apply and we return true. If the banner has attribute disseminationControls specified with a value containing [FOUO], or no element has attribute disseminationControls specified with value containing [FOUO], then the rule does not apply and we return true. Otherwise, we make sure that an element has attribute nonICmarkings specified with a value of [SBU], [SBU-NF], [LES], or [LES-NF].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00066">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00066"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(not($bannerClassification='U'))
then true() else if(not(index-of($dcTagsFound,'FOUO')>0)) then true() else index-
of($partNonICmarkings_tok,'SBU')>0 or index-of($partNonICmarkings_tok,'SBU-NF')>0
or index-of($partNonICmarkings_tok,'LES')>0 or index-of($partNonICmarkings_tok,'LES-
NF')>0 "
flag="error">
[ISM-ID-00066][Error] USA Unclassified documents having FOUO data and not having SBU, SBU-
NF, LES, or LES-NF must have
FOUO at the resource level.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00067

FileName:./Rules/resourceElement/ISM_ID_00067.sch

Rule Description:

[ISM-ID-00067][Error] If ISM-CAPCO-RESOURCE and any element meeting ISM-CONTRIBUTES in the document have the attribute disseminationControls containing [OC] then the ISM-RESOURCE-ELEMENT must have disseminationControls containing [OC]. Human Readable: USA documents having ORCON data must have ORCON at the resource level.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. We make sure that no element does not have attribute excludeFromRollup set to true has attribute disseminationControls specified with a value containing [OC] unless the resourceElement also has attribute disseminationControls specified with a value containing [OC].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00067">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00067"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else not(index-of($dcTagsFound,'OC') >
0) "
flag="error">
[ISM-ID-00067][Error] USA documents having ORCON data must have ORCON at the resource
level.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00068

FileName:./Rules/resourceElement/ISM_ID_00068.sch

Rule Description:

[ISM-ID-00068][Error] If ISM-CAPCO-RESOURCE and any element meeting ISM-CONTRIBUTES in the document have the attribute disseminationControls containing [IMC] then the ISM-RESOURCE-ELEMENT must have disseminationControls containing [IMC]. Human Readable: USA documents having IMCON data must have IMCON at the resource level.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. We make sure that no element that does not have attribute excludeFromRollup set to true has attribute disseminationControls specified with a value containing [IMC] unless the resourceElement also has attribute disseminationControls specified with a value containing [IMC].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00068">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00068"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else not(index-of($dcTagsFound,'IMC') >
0) "
flag="error">
[ISM-ID-00068][Error] USA documents having IMCON data must have IMCON at the resource
level.
</sch:assert>
</sch:rule>
</sch:pattern>
```


Rule: ISM-ID-00070

FileName:./Rules/resourceElement/ISM_ID_00070.sch

Rule Description:

[ISM-ID-00070][Error] If ISM-CAPCO-RESOURCE and any element meeting ISM-CONTRIBUTES in the document have the attribute disseminationControls containing [NF] then the ISM-RESOURCE-ELEMENT must have disseminationControls containing [NF]. Human Readable: USA documents having NF data must have NF at the resource level.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. We make sure that no element that does not have attribute excludeFromRollup set to true has attribute disseminationControls specified with a value containing [NF] unless the resourceElement also has attribute disseminationControls specified with a value containing [NF].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00070">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00070"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else not(index-of($dcTagsFound,'NF') >
0) "
flag="error">
[ISM-ID-00070][Error] USA documents having NF data must have NF at the resource level.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00071

FileName:./Rules/resourceElement/ISM_ID_00071.sch

Rule Description:

[ISM-ID-00071][Error] If ISM-CAPCO-RESOURCE and any element meeting ISM-CONTRIBUTES in the document have the attribute disseminationControls containing [PR] then the ISM-RESOURCE-ELEMENT must have disseminationControls containing [PR]. Human Readable: USA documents having PROPIN data must have PROPIN at the resource level.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. We make sure that no element that does not have attribute excludeFromRollup set to true has attribute disseminationControls specified with a value containing [PR] unless the resourceElement also has attribute disseminationControls specified with a value containing [PR].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00071">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00071"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else not(index-of($dcTagsFound,'PR') >
0) "
flag="error">
[ISM-ID-00071][Error] USA documents having PROPIN data must have PROPIN at the resource
level.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00072

FileName:./Rules/resourceElement/ISM_ID_00072.sch

Rule Description:

[ISM-ID-00072][Error] If ISM-CAPCO-RESOURCE and any element meeting ISM-CONTRIBUTES in the document have the attribute atomicEnergyMarkings containing [RD] then the ISM-RESOURCE-ELEMENT must have atomicEnergyMarkings containing [RD]. Human Readable: USA documents having Restricted Data must have Restricted Data at the resource level.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. We make sure that no element that does not have attribute excludeFromRollup set to true has attribute atomicEnergyMarkings specified with a value containing [RD] unless the resourceElement also has attribute atomicEnergyMarkings specified with a value containing [RD].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00072">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00072"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else not(index-of($aeaTagsFound,'RD') >
0) "
flag="error">
[ISM-ID-00072][Error] USA documents having Restricted Data must have Restricted Data at
the resource level.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00073

FileName:./Rules/resourceElement/ISM_ID_00073.sch

Rule Description:

[ISM-ID-00073][Error] If ISM-CAPCO-RESOURCE and any element meeting ISM-CONTRIBUTES in the document have the attribute atomicEnergyMarkings containing [RD-CNWDI] then the ISM-RESOURCE-ELEMENT must have atomicEnergyMarkings containing [RD-CNWDI]. Human Readable: USA documents having Restricted CNWDI Data must have Restricted CNWDI Data at the resource level.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. We make sure that no element that does not have attribute excludeFromRollup set to true has attribute atomicEnergyMarkings specified with a value containing [RD-CNWDI] unless the resourceElement also has attribute atomicEnergyMarkings specified with a value containing [RD-CNWDI].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00073">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00073"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else not(index-of($aeaTagsFound,'RD-
CNWDI') > 0) "
flag="error">
[ISM-ID-00073][Error] USA documents having Restricted CNWDI Data must have Restricted
CNWDI Data at the resource level.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00074

FileName:./Rules/resourceElement/ISM_ID_00074.sch

Rule Description:

[ISM-ID-00074][Error] If ISM-CAPCO-RESOURCE and any element meeting ISM-CONTRIBUTES in the document has the attribute atomicEnergyMarkings containing [RD-SG-##] then the ISM-RESOURCE-ELEMENT must have atomicEnergyMarkings containing [RD-SG-##]. ## represent digits 1 through 99 the ## must match. Human Readable: USA documents having Restricted SIGMA-## Data must have the same Restricted SIGMA-## Data at the resource level.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. We make sure that no element that does not have attribute excludeFromRollup set to true has attribute atomicEnergyMarkings specified with a value containing [RD-SG-##], where ## is represented by a regular expression matching numbers 1 through 99, unless the resourceElement also has attribute atomicEnergyMarkings specified with a value containing [RD-SG-##].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00074">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00074"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else sum( for $item in for $token in
$partAtomicEnergyMarkings_tok return if(matches($token,'^RD-SG-[1-9][0-9]?$')) then $token
else null return if(index-of($bannerAtomicEnergyMarkings_tok, $item)>0) then 0 else
1 )=0 "
flag="error">
[ISM-ID-00074][Error] USA documents having Restricted SIGMA-## Data must have the same
Restricted SIGMA-## Data at the resource level.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00075

FileName:./Rules/resourceElement/ISM_ID_00075.sch

Rule Description:

[ISM-ID-00075][Error] If ISM-CAPCO-RESOURCE and any element meeting ISM-CONTRIBUTES in the document have the attribute atomicEnergyMarkings containing [FRD] then the ISM-RESOURCE-ELEMENT must have atomicEnergyMarkings containing [FRD]. Human Readable: USA documents having Formerly Restricted Data must have Formerly Restricted Data at the resource level.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. We make sure that no element that does not have attribute excludeFromRollup set to true has attribute atomicEnergyMarkings specified with a value containing [FRD], unless the resourceElement also has attribute atomicEnergyMarkings specified with a value containing [FRD].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00075">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00075"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else not(index-of($aeaTagsFound,'FRD')
    &gt; 0) "
flag="error">
[ISM-ID-00075][Error] USA documents having Formerly Restricted Data must have Formerly
Restricted Data at the resource level.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00077

FileName:./Rules/resourceElement/ISM_ID_00077.sch

Rule Description:

[ISM-ID-00077][Error] If ISM-CAPCO-RESOURCE and any element meeting ISM-CONTRIBUTES in the document has the attribute atomicEnergyMarkings containing [FRD-SG-##] then the ISM-RESOURCE-ELEMENT must have atomicEnergyMarkings containing [FRD-SG-##]. ## represent digits 1 through 99 the ## must match. Human Readable: USA documents having Formerly Restricted SIGMA-## Data must have the same Formerly Restricted SIGMA-## Data at the resource level.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. We make sure that no element that does not have attribute excludeFromRollup set to true has attribute atomicEnergyMarkings specified with a value containing [FRD-SG-##], where ## is represented by a regular expression matching numbers 1 through 99, unless the resourceElement also has attribute atomicEnergyMarkings specified with a value containing [FRD-SG-##].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00077">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00077"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else sum( for $item in for $token in
$partAtomicEnergyMarkings_tok return if(matches($token,'^FRD-SG-[1-9][0-9]?$')) then
$token else null return if(index-of($bannerAtomicEnergyMarkings_tok, $item)>0) then 0
else 1 )=0 "
flag="error">
[ISM-ID-00077][Error] USA documents having Formerly Restricted SIGMA-## Data must have the
same Formerly Restricted SIGMA-## Data at
the resource level.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00078

FileName:./Rules/resourceElement/ISM_ID_00078.sch

Rule Description:

[ISM-ID-00078][Error] If ISM-CAPCO-RESOURCE RESOURCE and ISM-RESOURCE-ELEMENT element's classification has the value of [U] and any element meeting ISM-CONTRIBUTES in the document has the attribute atomicEnergyMarkings containing [DCNI] then the ISM-RESOURCE-ELEMENT must have atomicEnergyMarkings containing [DCNI]. Human Readable: Unclassified USA documents having DCNI Data must have DCNI at the resource level.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If the resourceElement has attribute classification specified with a value other than [U] then the rule does not apply and we return true. Otherwise, we make sure that no element has attribute atomicEnergyMarkings specified with a value containing [DCNI] and does not have attribute excludeFromRollup set to true, unless the resourceElement also has attribute atomicEnergyMarkings specified with a value containing [DCNI].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00078">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00078"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(not($bannerClassification='U'))
then true() else not(index-of($aeaTagsFound,'DCNI') &gt; 0) "
flag="error">
[ISM-ID-00078][Error] Unclassified USA documents having DCNI Data must have DCNI at the
resource level.
</sch:assert>
</sch:rule>
</sch:pattern>
```


Rule: ISM-ID-00079

FileName:./Rules/resourceElement/ISM_ID_00079.sch

Rule Description:

[ISM-ID-00079][Error] If ISM-CAPCO-RESOURCE and ISM-RESOURCE-ELEMENT element's classification has the value of [U] and any element meeting ISM-CONTRIBUTES in the document has the attribute atomicEnergyMarkings containing [UCNI] then the ISM-RESOURCE-ELEMENT must have atomicEnergyMarkings containing [UCNI]. Human Readable: Unclassified USA documents having UCNI Data must have UCNI at the resource level.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If the resourceElement has attribute classification specified with a value other than [U] then the rule does not apply and we return true. Otherwise, we make sure that no element has attribute atomicEnergyMarkings specified with a value containing [UCNI] and does not have attribute excludeFromRollup set to true, unless the resourceElement also has attribute atomicEnergyMarkings specified with a value containing [UCNI].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00079">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00079"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(not($bannerClassification='U'))
then true() else not(index-of($aeaTagsFound,'UCNI') > 0) "
flag="error">
[ISM-ID-00079][Error] Unclassified USA documents having UCNI Data must have UCNI at the
resource level.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00080

FileName:./Rules/resourceElement/ISM_ID_00080.sch

Rule Description:

[ISM-ID-00080][Error] If ISM-CAPCO-RESOURCE and any element meeting ISM-CONTRIBUTES in the document have the attribute disseminationControls containing [DSEN] then the ISM-RESOURCE-ELEMENT must have disseminationControls containing [DSEN]. Human Readable: USA documents having DSEN Data must have DSEN at the resource level.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. We make sure that no element that does not have attribute excludeFromRollup set to true has attribute disseminationControls specified with a value containing [DSEN], unless the resourceElement also has attribute disseminationControls specified with a value containing [DSEN].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00080">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00080"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else not(index-of($dcTagsFound,'DSEN')
> 0) "
flag="error">
[ISM-ID-00080][Error] USA documents having DSEN Data must have DSEN at the resource level.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00081

FileName:./Rules/resourceElement/ISM_ID_00081.sch

Rule Description:

[ISM-ID-00081][Error] If ISM-CAPCO-RESOURCE and any element meeting ISM-CONTRIBUTES in the document have the attribute disseminationControls containing [FISA] then the ISM-RESOURCE-ELEMENT must have disseminationControls containing [FISA]. Human Readable: USA documents having FISA Data must have FISA at the resource level.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. We make sure that no element that does not have attribute excludeFromRollup set to true has attribute disseminationControls specified with a value containing [FISA] and does not have attribute excludeFromRollup set to true, unless the resourceElement also has attribute disseminationControls specified with a value containing [FISA].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00081">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00081"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else not(index-of($dcTagsFound,'FISA')
&gt; 0) "
flag="error">
[ISM-ID-00081][Error] USA documents having FISA Data must have FISA at the resource level.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00082

FileName:./Rules/resourceElement/ISM_ID_00082.sch

Rule Description:

[ISM-ID-00082][Error] If ISM-CAPCO-RESOURCE and any element meeting ISM-CONTRIBUTES in the document have the attribute nonICmarkings containing [SC] then the ISM-RESOURCE-ELEMENT must have nonICmarkings containing [SC]. Human Readable: USA documents having SC Data must have SC at the resource level.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If any element has attribute nonICmarkings specified with a value containing [SC] and does not have attribute excludeFromRollup set to true, then we make sure that the resourceElement also has attribute nonICmarkings specified with a value containing [SC].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00082">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00082"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(index-of($partNonICmarkings_tok,
'SC') > 0) then (index-of($bannerNonICmarkings_tok, 'SC') > 0) else true() "
flag="error">
[ISM-ID-00082][Error] USA documents having SC Data must have SC at the resource level.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00083

FileName:../Rules/resourceElement/ISM_ID_00083.sch

Rule Description:

[ISM-ID-00083][Error] If ISM-CAPCO-RESOURCE and any element meeting ISM-CONTRIBUTES in the document have the attribute nonICmarkings containing [SINFO] then the ISM-RESOURCE-ELEMENT must have nonICmarkings containing [SINFO]. Human Readable: USA documents having SINFO Data must have SINFO at the resource level.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If any element has attribute nonICmarkings specified with a value containing [SINFO] and does not have attribute excludeFromRollup set to true, then we make sure that the resourceElement also has attribute nonICmarkings specified with a value containing [SINFO].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00083">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00083"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(index-of($partNonICmarkings_tok,
'SINFO') > 0) then (index-of($bannerNonICmarkings_tok, 'SINFO') > 0) else true() "
flag="error">
[ISM-ID-00083][Error] USA documents having SINFO Data must have SINFO at the resource
level.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00084

FileName:./Rules/resourceElement/ISM_ID_00084.sch

Rule Description:

[ISM-ID-00084][Error] If ISM-CAPCO-RESOURCE and any element meeting ISM-CONTRIBUTES in the document have the attribute nonICmarkings containing [DS] then the ISM-RESOURCE-ELEMENT must have nonICmarkings containing [DS]. Human Readable: USA documents having DS Data must have DS at the resource level.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If any element has attribute nonICmarkings specified with a value containing [DS] and does not have attribute excludeFromRollup set to true, then we make sure that the resourceElement also has attribute nonICmarkings specified with a value containing [DS].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00084">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00084"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(index-of($partNonICmarkings_tok,
'DS') > 0) then (index-of($bannerNonICmarkings_tok, 'DS') > 0) else true() "
flag="error">
[ISM-ID-00084][Error] USA documents having DS Data must have DS at the resource level.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00085

FileName:./Rules/resourceElement/ISM_ID_00085.sch

Rule Description:

[ISM-ID-00085][Error] If ISM-CAPCO-RESOURCE and any element meeting ISM-CONTRIBUTES in the document has the attribute nonICmarkings containing [XD] and does not have any element meeting ISM-CONTRIBUTES in the document having the attribute nonICmarkings containing [ND] then the ISM-RESOURCE-ELEMENT must have nonICmarkings containing [XD]. Human Readable: USA documents having XD Data and not having ND must have XD at the resource level.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If any element has attribute nonICmarkings specified with a value containing [XD] and no element has attribute nonICmarkings specified with a value containing [ND] and does not have attribute excludeFromRollup set to true, then we make sure that the resourceElement has attribute nonICmarkings specified with a value containing [XD].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00085">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00085"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(index-of($partNonICmarkings_tok,
'XD') > 0 and not(index-of($partNonICmarkings_tok, 'ND')>0)) then index-
of($bannerNonICmarkings_tok, 'XD') > 0 else true() "
flag="error">
[ISM-ID-00085][Error] USA documents having XD Data and not having ND must have XD at the
resource level.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00086

FileName:./Rules/resourceElement/ISM_ID_00086.sch

Rule Description:

[ISM-ID-00086][Error] If ISM-CAPCO-RESOURCE and any element in the document: 1. Meets ISM-CONTRIBUTES AND 2. Has the attribute nonICmarkings containing [ND] Then the ISM-RESOURCE-ELEMENT must have nonICmarkings containing [ND]. Human Readable: USA documents having ND Data must have ND at the resource level.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If any element has attribute nonICmarkings specified with a value containing [ND] and does not have attribute excludeFromRollup set to true, then we make sure that the resourceElement has attribute nonICmarkings specified with a value containing [ND].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00086">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00086"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(index-of($partNonICmarkings_tok,
'ND')>0) then index-of($bannerNonICmarkings_tok, 'ND') > 0 else true() "
flag="error">
[ISM-ID-00086][Error] USA documents having ND Data must have ND at the resource level.
</sch:assert>
</sch:rule>
</sch:pattern>
```


Rule: ISM-ID-00087

FileName:./Rules/resourceElement/ISM_ID_00087.sch

Rule Description:

[ISM-ID-00087][Error] If ISM-CAPCO-RESOURCE and there exist at least 2 elements in the document: 1. Each element: Meets ISM-CONTRIBUTES AND 2. One of the elements: Has the attribute nonICmarkings containing [SBU-NF] AND 3. One of the elements: Has the attribute classification NOT having a value of [U] Then the ISM-RESOURCE-ELEMENT must have disseminationControls containing [NF]. Human Readable: Classified USA documents having SBU-NF Data must have NF at the resource level.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If any element has attribute nonICmarkings specified with a value containing [SBU-NF], does not have attribute excludeFromRollup set to true, and the resourceElement has attribute classification specified with a value other than [U], then we make sure that the resourceElement has attribute disseminationControls specified with a value containing [NF].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00087">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00087"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(index-of($partNonICmarkings_tok,
'SBU-NF') > 0 and not($bannerClassification='U')) then (index-
of($bannerDisseminationControls_tok, 'NF') > 0) else true() "
flag="error">
[ISM-ID-00087][Error] Classified USA documents having SBU-NF Data must have NF at the
resource level.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00088

FileName:./Rules/resourceElement/ISM_ID_00088.sch

Rule Description:

[ISM-ID-00088][Error] If ISM-CAPCO-RESOURCE and any element in the document: 1. Meets ISM-CONTRIBUTES AND 2. Has the attribute disseminationControls NOT containing [REL] AND 3. Has the attribute classification NOT having a value of [U] Then the ISM-RESOURCE-ELEMENT must NOT have attribute disseminationControls containing [REL].

Human Readable: USA documents having any classified portion that is not Releasable can't be REL at the resource level.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. Otherwise, we loop over all portions of the document and count the number of elements which have attribute classification specified with a value of [U] and have attribute disseminationControls specified with a value containing [REL]. The loop checks, if the current element has attribute classification specified with a value other than [U], then the rule does not apply to that element and we continue to the next element. Otherwise, if the current element has attribute disseminationControls specified with a value containing [REL], then the rule applies to this element and we return 1. If the count of elements meeting the two requirements stated above is great than zero, then we make sure that the resourceElement has attribute disseminationControls specified with a value containing [REL].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00088">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00088"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if( sum( for $token in $partTags
return if(matches($token/@ism:classification, '^U$')) then 0 else if( count( if(index-
of(tokenize($token/@ism:disseminationControls,' '), 'REL')>0) then null else 1 )=0)
then 0 else 1 ) &gt; 0 ) then not(index-of($bannerDisseminationControls_tok,'REL')>0)
else true() "
flag="error">
[ISM-ID-00088][Error] USA documents having any classified portion that is not Releasable
can't be REL at the resource level.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00090

FileName:./Rules/resourceElement/ISM_ID_00090.sch

Rule Description:

[ISM-ID-00090][Error] If ISM-CAPCO-RESOURCE and any element: 1. Meets ISM-CONTRIBUTES AND 2. Has the attribute disseminationControls containing [REL] Then the ISM-RESOURCE-ELEMENT must not have attribute disseminationControls containing [EYES]. Human Readable: USA documents with any portion that is REL must not be EYES at the resource level.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If any element has attribute disseminationControls specified with a value containing [REL], then we make sure that the resourceElement has attribute disseminationControls specified with a value other than [EYES].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00090">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00090"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(sum(for $token in $partTags return
if(contains($token/@ism:disseminationControls, 'REL')) then 1 else 0) ) then not(index-
of($bannerDisseminationControls_tok, 'EYES')>0) else true() "
flag="error">
[ISM-ID-00090][Error] USA documents with any portion that is REL must not be EYES at the
resource level.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00094

FileName:./Rules/disseminationControls/ISM_ID_00094.sch

Rule Description:

[ISM-ID-00094][Error] ISM-CAPCO-RESOURCE and attribute disseminationControls contains the name token [REL], then attribute classification must not have a value of [U]. Human Readable: REL may not be used on Unclassified portions.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If it is a CAPCO resource then we check the classification attribute of the current element. If it does not have value of [U] we return true since this rule only applies to unclassified elements. If it is not [U] then we check that the attribute disseminationControls does not have a value of [REL]

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00094">

<sch:rule context="//*[@ism:disseminationControls]">
<sch:assert id="ism00094"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(not(./@ism:classification='U'))
then true() else not(index-of(tokenize(./@ism:disseminationControls, ' '), 'REL')>0) "
flag="error">
[ISM-ID-00094][Error] REL may not be used on Unclassified portions.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00095

FileName:./Rules/FGIsourceOpen/ISM_ID_00095.sch

Rule Description:

[ISM-ID-00095][Error] If ISM-CAPCO-RESOURCE and attribute FGIsourceOpen is specified then each of its values must be ordered in accordance with CVEnumISMFGIOpen.xml.

Code Description:

This rule calls upon an abstract pattern to perform the logic work. It makes sure that every value in the FGIsourceProtected attributes are sorted in the order found in the CVEnumISMFGIOpen.xml file

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00095">

<sch:rule context="//*[@ism:FGIsourceOpen]">
<!-- Define variables -->
<sch:let name="errFlag_AlphabeticalOrder" value="error"/>
<sch:let name="errMsg_AlphabeticalOrder"
value="' [ISM-ID-00095][Error] If ISM-CAPCO-RESOURCE and attribute FGIsourceOpen is
specified then each of its values must be ordered in accordance with CVEnumISMFGIOpen.xml.
'"/>

<sch:let name="dataFileElems" value="$FGIsourceOpenList"/>
<sch:let name="attrValues" value="./@ism:FGIsourceOpen"/>
<sch:let name="attrValueTokens" value="tokenize($attrValues,' ')/>

<!-- Get the position of each client node relative to its position in the master list. If
the node is not found, return a -1 -->
<sch:let name="orderNums"
value=" for $token in $attrValueTokens return if
($dataFileElems[matches($token,concat('^',text(),'$'))]) then
count(($dataFileElems[matches($token,concat('^',text(),'$'))])/preceding::* + 1 else -1"/>

<!-- Create a sequence that returns a 0 if the previous sibling has a higher order number,
else return a 1 -->
<sch:let name="sortedOrderNums"
value=" for $peice in distinct-values(for $token in $orderNums return index-of($orderNums,
$token)) return if($peice>1) then if($orderNums[$peice] = $orderNums[$peice - 1]) then
if(compare($attrValueTokens[$peice - 1],$attrValueTokens[$peice]) = 1) then 0 else 2 else
if($orderNums[$peice] > $orderNums[$peice - 1]) then 1 else 0 else 1 "/>
<sch:let name="hasUnsorted" value="count(index-of($sortedOrderNums,0)) > 0"/>
<sch:let name="unsortedValues"
```

```
value=" if ($hasUnsorted) then distinct-values( for $token in index-of($sortedOrderNums,0)
return $attrValueTokens[$token] ) else null "/>

<sch:assert test="not($hasUnsorted)" flag="$errFlag_AlphabeticalOrder">
<sch:value-of select="$errMsg_AlphabeticalOrder"/>
The following values are out of order [<sch:value-of select="$unsortedValues"/>] for
[<sch:value-of select="$attrValueTokens"/>] </sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00096

FileName:./Rules/FGIsourceProtected/ISM_ID_00096.sch

Rule Description:

[ISM-ID-00096][Error] If ISM-CAPCO-RESOURCE and attribute FGIsourceProtected is specified then each of its values must be ordered in accordance with CVEnumISMFGIProtected.xml.

Code Description:

This rule makes sure that every value in the FGIsourceProtected attributes are sorted in the order found in the CVEnumISMFGIProtected.xml file

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00096">

<sch:rule context="//*[@ism:FGIsourceProtected]">
<!-- Define variables -->
<sch:let name="errFlag_AlphabeticalOrder" value="error"/>
<sch:let name="errMsg_AlphabeticalOrder"
value="' [ISM-ID-00096][Error] If ISM-CAPCO-RESOURCE and attribute FGIsourceProtected
is specified then each of its values must be ordered in accordance with
CVEnumISMFGIProtected.xml. '"/>

<sch:let name="dataFileElems" value="$FGIsourceProtectedList"/>
<sch:let name="attrValues" value="./@ism:FGIsourceProtected"/>
<sch:let name="attrValueTokens" value="tokenize($attrValues,' ')/>

<!-- Get the position of each client node relative to its position in the master list. If
the node is not found, return a -1 -->
<sch:let name="orderNums"
value=" for $token in $attrValueTokens return if
($dataFileElems[matches($token,concat('^',text(),'$'))]) then
count(($dataFileElems[matches($token,concat('^',text(),'$'))]/preceding::*) + 1 else -1"/>

<!-- Create a sequence that returns a 0 if the previous sibling has a higher order number,
else return a 1 -->
<sch:let name="sortedOrderNums"
value=" for $peice in distinct-values(for $token in $orderNums return index-of($orderNums,
$token)) return if($peice>1) then if($orderNums[$peice] = $orderNums[$peice - 1]) then
if(compare($attrValueTokens[$peice - 1],$attrValueTokens[$peice]) = 1) then 0 else 2 else
if($orderNums[$peice] > $orderNums[$peice - 1]) then 1 else 0 else 1 "/>
<sch:let name="hasUnsorted" value="count(index-of($sortedOrderNums,0)) > 0"/>
<sch:let name="unsortedValues"
```

```
value=" if ($hasUnsorted) then distinct-values( for $token in index-of($sortedOrderNums,0)
return $attrValueTokens[$token] ) else null "/>

<sch:assert test="not($hasUnsorted)" flag="$errFlag_AlphabeticalOrder">
<sch:value-of select="$errMsg_AlphabeticalOrder"/>
The following values are out of order [<sch:value-of select="$unsortedValues"/>] for
[<sch:value-of select="$attrValueTokens"/>] </sch:assert>
</sch:rule>
</sch:pattern>
```


Rule: ISM-ID-00097

FileName:../Rules/FGIsourceProtected/ISM_ID_00097.sch

Rule Description:

[ISM-ID-00097][Warning] If ISM-CAPCO-RESOURCE and attribute FGIsourceProtected is specified with a value other than [FGI] then the value(s) must not be discoverable in IC shared spaces. Human Readable: FGI Protected should rarely if ever be seen outside of an agency's internal systems.

Code Description:

Checks that the resource is a CAPCO resource and that the FGIsourceProtected attribute does not contain the value [FGI]

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00097">

<sch:rule context="//*[@ism:FGIsourceProtected]">
<sch:assert id="ism00097"
test=" $ISM_CAPCO_RESOURCE and normalize-space(./@ism:FGIsourceProtected)='FGI' "
flag="error">
[ISM-ID-00097][Warning] FGI Protected should rarely if ever be seen outside of an agency's
internal systems.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00099

FileName:../Rules/ownerProducer/ISM_ID_00099.sch

Rule Description:

[ISM-ID-00099][Error] If ISM-CAPCO-RESOURCE attribute ownerProducer contains [FGI], it must be the only value.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. Otherwise, we make sure that the current element has attribute ownerProducer specified with [FGI] as its only value.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00099">

<sch:rule context="//*[@ism:ownerProducer]">
<sch:let name="opTok" value="tokenize(./@ism:ownerProducer,' ')" />
<sch:assert id="ism00099"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else not(index-of($opTok,'FGI')>0 and
count($opTok)>1) "
flag="error">
[ISM-ID-00099][Error] If ISM-CAPCO-RESOURCE attribute ownerProducer contains [FGI],
it must be the only value.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00100

FileName:./Rules/ownerProducer/ISM_ID_00100.sch

Rule Description:

[ISM-ID-00100][Error] If ISM-CAPCO-RESOURCE and attribute ownerProducer is specified, then each of its values must be ordered in accordance with CVENumISMOwnerProducer.xml.

Code Description:

This rule calls upon an abstract pattern to perform the logic work. It makes sure that every value in the ownerProducer attributes are sorted in the order found in the CVENumISMOwnerProducer.xml file.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00100">

<sch:rule context="//*[@ism:ownerProducer]">
<!-- Define variables -->
<sch:let name="errFlag_AlphabeticalOrder" value="error"/>
<sch:let name="errMsg_AlphabeticalOrder"
value="' [ISM-ID-00100][Error] If ISM-CAPCO-RESOURCE and attribute ownerProducer
is specified, then each of its values must be ordered in accordance with
CVENumISMOwnerProducer.xml. '"/>

<sch:let name="dataFileElems" value="$ownerProducerList"/>
<sch:let name="attrValues" value="./@ism:ownerProducer"/>
<sch:let name="attrValueTokens" value="tokenize($attrValues,' ')/>

<!-- Get the position of each client node relative to its position in the master list. If
the node is not found, return a -1 -->
<sch:let name="orderNums"
value=" for $token in $attrValueTokens return if
($dataFileElems[matches($token,concat('^',text(),'$'))]) then
count(($dataFileElems[matches($token,concat('^',text(),'$'))]/preceding::*) + 1 else -1"/>

<!-- Create a sequence that returns a 0 if the previous sibling has a higher order number,
else return a 1 -->
<sch:let name="sortedOrderNums"
value=" for $peice in distinct-values(for $token in $orderNums return index-of($orderNums,
$token)) return if($peice>1) then if($orderNums[$peice] = $orderNums[$peice - 1]) then
if(compare($attrValueTokens[$peice - 1],$attrValueTokens[$peice]) = 1) then 0 else 2 else
if($orderNums[$peice] > $orderNums[$peice - 1]) then 1 else 0 else 1 "/>
<sch:let name="hasUnsorted" value="count(index-of($sortedOrderNums,0)) > 0"/>
```

```
<sch:let name="unsortedValues"
value=" if ($hasUnsorted) then distinct-values( for $token in index-of($sortedOrderNums,0)
return $attrValueTokens[$token] ) else null "/>

<sch:assert test="not($hasUnsorted)" flag="$errFlag_AlphabeticalOrder">
<sch:value-of select="$errMsg_AlphabeticalOrder"/>
The following values are out of order [<sch:value-of select="$unsortedValues"/>] for
[<sch:value-of select="$attrValueTokens"/>] </sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00102

FileName:../Rules/generalConstraints/ISM_ID_00102.sch

Rule Description:

[ISM-ID-00102][Error] The root element must have the attribute DESVersion in the namespace urn:us:gov:ic:ism

Code Description:

The code checks that the the root node has attribute DESVersion specified

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00102">

<sch:rule context="/">
<sch:assert id="ism00102" test=" ./@ism:DESVersion " flag="error">
[ISM-ID-00102][Error] The root element must have the attribute
DESVersion in the namespace urn:us:gov:ic:ism
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00103

FileName:../Rules/generalConstraints/ISM_ID_00103.sch

Rule Description:

[ISM-ID-00103][Error] At least one element must have resourceElement true.

Code Description:

The code loops over all elements which have ISM attributes present and counts the elements which specify the the attribute resourceElement. Then it makes sure that the total is greater than zero.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00103">

<sch:rule context="/*">
<sch:assert id="ism00103"
test=" count( for $token in /*[(@ism:*)] return if($token/@ism:resourceElement=true())
then 1 else null ) > 0 "
flag="error">
[ISM-ID-00103][Error] At least one element must have resourceElement true.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00104

FileName:./Rules/resourceElement/ISM_ID_00104.sch

Rule Description:

[ISM-ID-00104][Error] If ISM-CAPCO-RESOURCE and any element in the document: 1. Meets ISM-CONTRIBUTES AND 2. Has the attribute nonICmarkings containing [SBU-NF] AND 3. The ISM-RESOURCE-ELEMENT has attribute classification equal to [U] Then the ISM-RESOURCE-ELEMENT must have nonICmarkings containing [SBU-NF]. Human Readable: Unclassified USA documents having SBU-NF must have SBU-NF at the resource level.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If any element has attribute nonICmarkings specified with a value containing [SBU-NF] and the resourceElement has the attribute classification specified with a value of [U], then we make sure that the resourceElement has attribute nonICmarkings specified with a value other than [SBU-NF].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00104">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00104"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(index-of($partNonICmarkings_tok,
'SBU-NF') > 0 and $bannerClassification='U') then (index-of(./@ism:nonICmarkings, 'SBU-
NF') > 0) else true() "
flag="error">
[ISM-ID-00104][Error] Unclassified USA documents having SBU-NF must have SBU-NF at the
resource level.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00105

FileName:./Rules/resourceElement/ISM_ID_00105.sch

Rule Description:

[ISM-ID-00105][Error] If ISM-CAPCO-RESOURCE and any element in the document: 1. Meets ISM-CONTRIBUTES AND 2. Has the attribute nonICmarkings containing [SBU] AND 3. The ISM-RESOURCE-ELEMENT has attribute classification equal to [U] Then the ISM-RESOURCE-ELEMENT must have nonICmarkings containing [SBU]. Human Readable: Unclassified USA documents having SBU must have SBU at the resource level.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If any element has attribute nonICmarkings specified with a value containing [SBU] and the resourceElement has the attribute classification specified with a value of [U], then we make sure that the resourceElement has attribute nonICmarkings specified with a value of [SBU].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00105">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00105"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(index-of($partNonICmarkings_tok,
'SBU') > 0 and $bannerClassification='U') then index-of($bannerNonICmarkings_tok,
'SBU') > 0 else true() "
flag="error">
[ISM-ID-00105][Error] Unclassified USA documents having SBU must have SBU at the resource
level.
</sch:assert>
</sch:rule>
</sch:pattern>
```


Rule: ISM-ID-00107

FileName:./Rules/disseminationControls/ISM_ID_00107.sch

Rule Description:

[ISM-ID-00107][Error] If ISM-CAPCO-RESOURCE and attribute disseminationControls contains the name token [IMC] then attribute classification must have a value of [TS] or [S]. Human Readable: IMCON may only be used with SECRET (S) information at the portion level. At the document (banner) level, it may be appear with (S) or (TS) banners.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If it is a CAPCO resource then we check the attribute classification of the element and return true if it has a value of [S] or [TS]. Otherwise we check that the attribute disseminationControls does not contain the value [IMC].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00107">

<sch:rule context="//*[@ism:disseminationControls]">
<sch:assert id="ism00107"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(matches(./
@ism:classification, '^(TS|S)$')) then true() else not(index-of(tokenize(./
@ism:disseminationControls, ' '), 'IMC')>0) "
flag="error">
[ISM-ID-00107][Error] IMCON may only be used with SECRET (S) information at the portion
level.
At the document (banner) level, it may be appear with (S) or (TS) banners.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00108

FileName:../Rules/resourceElement/ISM_ID_00108.sch

Rule Description:

[ISM-ID-00108][Error] If ISM-CAPCO-RESOURCE and attribute classification of ISM-RESOURCE-ELEMENT has a value of [TS] and attribute compilationReason does not have a value then at least one element meeting ISM-CONTRIBUTES in the document must have a classification attribute of [TS]. Human Readable: USA TS documents not using compilation must have TS data.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. Then make sure we are looking at the resourceElement and if the resourceElement has attribute classification specified with a value of [TS] and the resourceElement has attribute compilationReason specified with an empty value, then we make sure that at least one element in the document has attribute classification specified with a value of [TS].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00108">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:include href="../../../_Properties/myPosition.sch"/>
<sch:assert id="ism00108"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if($myPosition =
$resourceElementPosition) then true() else if ($bannerClassification='TS' and
not(string-length(normalize-space(./@ism:compilationReason))>0)) then index-
of($partClassification_tok,'TS')>0 else true() "
flag="error">
[ISM-ID-00108][Error] USA TS documents not using compilation must have TS data.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00109

FileName:./Rules/resourceElement/ISM_ID_00109.sch

Rule Description:

[ISM-ID-00109][Error] If ISM-CAPCO-RESOURCE and attribute classification of ISM-RESOURCE-ELEMENT has a value of [S] and attribute compilationReason does not have a value then at least one element meeting ISM-CONTRIBUTES in the document must have a classification attribute of [S]. Human Readable: USA S documents not using compilation must have S data.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If the resourceElement has attribute classification specified with a value of [S] and the resourceElement has attribute compilationReason specified with an empty value, then we make sure that at least one element in the document has attribute classification specified with a value of [S].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00109">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00109"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if ($bannerClassification='S'
and not(string-length(normalize-space(./@ism:compilationReason))>0)) then index-
of($partClassification_tok,'S')>0 else true() "
flag="error">
[ISM-ID-00109][Error] USA S documents not using compilation must have S data.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00110

FileName:./Rules/resourceElement/ISM_ID_00110.sch

Rule Description:

[ISM-ID-00110][Error] If ISM-CAPCO-RESOURCE and attribute classification of ISM-RESOURCE-ELEMENT has a value of [C] and attribute compilationReason does not have a value then at least one element meeting ISM-CONTRIBUTES in the document must have a classification attribute of [C]. Human Readable: USA C documents not using compilation must have C data.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If the resourceElement has attribute classification specified with a value of [C] and the resourceElement has attribute compilationReason specified with an empty value, then we make sure that at least one element in the document has attribute classification specified with a value of [C].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00110">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00110"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if ($bannerClassification='C'
and not(string-length(normalize-space(./@ism:compilationReason))>0)) then index-
of($partClassification_tok,'C')>0 else true() "
flag="error">
[ISM-ID-00110][Error] USA C documents not using compilation must have C data.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00111

FileName:./Rules/resourceElement/ISM_ID_00111.sch

Rule Description:

[ISM-ID-00111][Error] If ISM-CAPCO-RESOURCE and attribute SCIcontrols of ISM-RESOURCE-ELEMENT contains [SI] and attribute compilationReason does not have a value then at least one element meeting ISM-CONTRIBUTES in the document must have a SCIcontrols attribute containing [SI]. Human Readable: USA SI documents not using compilation must have SI data.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If the resourceElement has attribute SCIcontrols specified with a value containing [SI] and the resourceElement has attribute compilationReason specified with an empty value, then we make sure that at least one element in the document has attribute SCIcontrols specified with a value containing [SI].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00111">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00111"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if (index-of($bannerSCIcontrols_tok,
'SI')>0 and not(string-length(normalize-space(./@ism:compilationReason))>0)) then
index-of($partSCIcontrols_tok,'SI')>0 else true() "
flag="error">
[ISM-ID-00111][Error] USA SI documents not using compilation must have SI data.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00112

FileName:./Rules/resourceElement/ISM_ID_00112.sch

Rule Description:

[ISM-ID-00112][Error] If ISM-CAPCO-RESOURCE and attribute SCIcontrols of ISM-RESOURCE-ELEMENT contains [SI-G] and attribute compilationReason does not have a value then at least one element meeting ISM-CONTRIBUTES in the document must have a SCIcontrols attribute containing [SI-G]. Human Readable: USA SI-G documents not using compilation must have SI-G data.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If the resourceElement has attribute SCIcontrols specified with a value containing [SI-G] and the resourceElement has attribute compilationReason specified with an empty value, then we make sure that at least one element in the document has attribute SCIcontrols specified with a value containing [SI-G].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00112">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00112"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if (contains($bannerSCIcontrols, 'SI-
G') and not(string-length(normalize-space(./@ism:compilationReason))>0)) then index-
of($partSCIcontrols_tok,'SI-G')>0 else true() "
flag="error">
[ISM-ID-00112][Error] USA SI-G documents not using compilation must have SI-G data.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00113

FileName:./Rules/resourceElement/ISM_ID_00113.sch

Rule Description:

[ISM-ID-00113][Error] If ISM-CAPCO-RESOURCE and attribute SCIcontrols of ISM-RESOURCE-ELEMENT contains [TK] and attribute compilationReason does not have a value then at least one element meeting ISM-CONTRIBUTES in the document must have a SCIcontrols attribute containing [TK]. Human Readable: USA TK documents not using compilation must have TK data.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If the resourceElement has attribute SCIcontrols specified with a value containing [TK] and the resourceElement has attribute compilationReason specified with an empty value, then we make sure that at least one element in the document has attribute SCIcontrols specified with a value containing [TK].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00113">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00113"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if (index-of($bannerSCIcontrols_tok,
'TK')>0 and not(string-length(normalize-space(./@ism:compilationReason))>0)) then
index-of($partSCIcontrols_tok,'TK')>0 else true() "
flag="error">
[ISM-ID-00113][Error] USA TK documents not using compilation must have TK data.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00116

FileName:./Rules/resourceElement/ISM_ID_00116.sch

Rule Description:

[ISM-ID-00116][Error] If ISM-CAPCO-RESOURCE and attribute SCIcontrols of ISM-RESOURCE-ELEMENT contains [HCS] and attribute compilationReason does not have a value then at least one element meeting ISM-CONTRIBUTES in the document must have a SCIcontrols attribute containing [HCS]. Human Readable: USA HCS documents not using compilation must have HCS data.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If the resourceElement has attribute SCIcontrols specified with a value containing [HCS] and the resourceElement has attribute compilationReason specified with an empty value, then we make sure that at least one element in the document has attribute SCIcontrols specified with a value containing [HCS].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00116">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00116"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if (index-of($bannerSCIcontrols_tok,
'HCS')>0 and not(string-length(normalize-space(./@ism:compilationReason))>0)) then
index-of($partSCIcontrols_tok,'HCS')>0 else true() "
flag="error">
[ISM-ID-00116][Error] USA HCS documents not using compilation must have HCS data.
</sch:assert>
</sch:rule>
</sch:pattern>
```


Rule: ISM-ID-00118

FileName:../Rules/resourceElement/ISM_ID_00118.sch

Rule Description:

[ISM-ID-00118][Error] The first element in document order having resourceElement true must have createDate specified.

Code Description:

We make sure that the resourceElement has attribute createDate specified.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00118">

<sch:rule context="//*[@ism:resourceElement=true()][1]">
<sch:assert id="ism00118"
test=" if($ISM_RESOURCE_ELEMENT/@ism:createDate) then true() else false() "
flag="error">
[ISM-ID-00118][Error] The first element in document order having
resourceElement true must have createDate specified.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00119

FileName:./Rules/generalConstraints/ISM_ID_00119.sch

Rule Description:

[ISM-ID-00119][Error] If ISM-CAPCO-RESOURCE and 1. attribute classification is not [U] AND 2. ISM-ICD-710Applies AND 3. Attribute disseminationControls does not contain one or more of [DISPLAYONLY], [REL], [RELIDO], [EYES], or [NF] Human Readable: All classified NSI that claims compliance with ICD 710 must have an appropriate foreign disclosure or release marking.

Code Description:

If CAPCO rules do not apply to the document, or ICD 710 does not apply to the document, or the resource is unclassified, then the the rule does not apply and we return true. Otherwise, we make sure that the attribute disseminationControls does not contain one of the values [DISPLAYONLY], [RELIDO], [REL], [EYES], or [NF].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00119">

<sch:rule context="//*[@ism:*]">
<sch:let name="dissemTok" value="tokenize(./@ism:disseminationControls,' ')" />
<sch:assert id="ism00119"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(not($ISM_ICD_710_APPLIES))
then true() else if(./@ism:classification='U') then true() else not( count( (if(index-
of($dissemTok, 'RELIDO')>0) then 1 else null, if(index-of($dissemTok, 'REL')>0)
then 1 else null, if(index-of($dissemTok, 'EYES')>0) then 1 else null, if(index-
of($dissemTok, 'DISPLAYONLY')>0) then 1 else null, if(index-of($dissemTok, 'NF')>0)
then 1 else null) )=0 ) "
flag="error">
[ISM-ID-00119][Error] All classified NSI that claims compliance with ICD 710 must have an
appropriate
foreign disclosure or release marking.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00121

FileName:../Rules/SARIdentifier/ISM_ID_00121.sch

Rule Description:

[ISM-ID-00121][Error] If ISM-CAPCO-RESOURCE and attribute SARIdentifier is specified, then each of its values must be ordered in accordance with CVEnumISMSAR.xml.

Code Description:

This rule calls upon an abstract pattern to perform the logic work. It makes sure that every value in the SARIdentifier attributes are sorted in the order found in the CVEnumISMSAR.xml file.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00121">

<sch:rule context="//*[@ism:SARIdentifier]">
<!-- Define variables -->
<sch:let name="errFlag_AlphabeticalOrder" value="error"/>
<sch:let name="errMsg_AlphabeticalOrder"
value="' [ISM-ID-00121][Error] If ISM-CAPCO-RESOURCE and attribute SARIdentifier is
specified, then each of its values must be ordered in accordance with CVEnumISMSAR.xml.'"/>

<sch:let name="dataFileElems" value="$SARIdentifierList"/>
<sch:let name="attrValues" value="./@ism:SARIdentifier"/>
<sch:let name="attrValueTokens" value="tokenize($attrValues,' ')/>

<!-- Get the position of each client node relative to its position in the master list. If
the node is not found, return a -1 -->
<sch:let name="orderNums"
value=" for $token in $attrValueTokens return if
($dataFileElems[matches($token,concat('^',text(),'$'))]) then
count(($dataFileElems[matches($token,concat('^',text(),'$'))]/preceding::*) + 1 else -1"/>

<!-- Create a sequence that returns a 0 if the previous sibling has a higher order number,
else return a 1 -->
<sch:let name="sortedOrderNums"
value=" for $peice in distinct-values(for $token in $orderNums return index-of($orderNums,
$token)) return if($peice>1) then if($orderNums[$peice] = $orderNums[$peice - 1]) then
if(compare($attrValueTokens[$peice - 1],$attrValueTokens[$peice]) = 1) then 0 else 2 else
if($orderNums[$peice] > $orderNums[$peice - 1]) then 1 else 0 else 1 "/>
<sch:let name="hasUnsorted" value="count(index-of($sortedOrderNums,0)) > 0"/>
```

```
<sch:let name="unsortedValues"
value=" if ($hasUnsorted) then distinct-values( for $token in index-of($sortedOrderNums,0)
return $attrValueTokens[$token] ) else null "/>

<sch:assert test="not($hasUnsorted)" flag="$errFlag_AlphabeticalOrder">
<sch:value-of select="$errMsg_AlphabeticalOrder"/>
The following values are out of order [<sch:value-of select="$unsortedValues"/>] for
[<sch:value-of select="$attrValueTokens"/>] </sch:assert>
</sch:rule>

</sch:pattern>
```

Rule: ISM-ID-00122

FileName:../Rules/SCIcontrols/ISM_ID_00122.sch

Rule Description:

[ISM-ID-00122][Error] If ISM-CAPCO-RESOURCE and attribute SCIcontrols contains the name token [KDK], then attribute classification must have a value of [TS] or [S].

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If the current element has attribute SCIcontrols specified with a value containing [KDK], then we make sure that attribute classification has a value [TS] or [S].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00122">

<sch:rule context="//*[@ism:SCIcontrols]">
<sch:assert id="ism00122"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else not(index-of(tokenize(./
@ism:SCIcontrols, ' '), 'KDK')>0 and not(matches(./@ism:classification, '^(TS|S)$')) ) "
flag="error">
[ISM-ID-00122][Error] If ISM-CAPCO-RESOURCE and attribute SCIcontrols contains the name
token [KDK], then attribute
classification must have a value of [TS] or [S].
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00123

FileName:./Rules/SCIcontrols/ISM_ID_00123.sch

Rule Description:

[ISM-ID-00123][Error] If ISM-CAPCO-RESOURCE and attribute SCIcontrols contains the name token [KDK], then attribute disseminationControls must contain the name token [NF].

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If the current element has attribute SCIcontrols specified with a value containing [KDK], then we make sure that attribute disseminationControls contains the value [NF].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00123">

<sch:rule context="//*[@ism:SCIcontrols]">
<sch:assert id="ism00123"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(index-of(tokenize(./
@ism:SCIcontrols, ' '), 'KDK')>0 and not( index-of(tokenize(./
@ism:disseminationControls, ' '), 'NF')) ) then false() else true() "
flag="error">
[ISM-ID-00123][Error] If ISM-CAPCO-RESOURCE and attribute SCIcontrols contains the name
token [KDK], then attribute
disseminationControls must contain the name token [NF].
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00124

FileName:../Rules/disseminationControls/ISM_ID_00124.sch

Rule Description:

[ISM-ID-00124][Warning] If ISM-CAPCO-RESOURCE and 1. not the ISM_CAPCO_RESOURCE AND 2. Attribute ownerProducer does not contain [USA]. AND 3. Attribute disseminationControls contains [RELIDO] Human Readable: RELIDO is not authorized for non US portions.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If it is a CAPCO resource and the current element is not \$ISM_CAPCO_RESOURCE then we check the attribute ownerProducer for not having a value of [USA] and that the attribute disseminationControls contains a value of [RELIDO] then we return false because the resource is not in compliance with the rule.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00124">

<sch:rule context="//*[@ism:disseminationControls]">
<sch:assert id="ism00124"
test=" if (not($ISM_CAPCO_RESOURCE)) then true() else if(not(index-of(tokenize(./
@ism:ownerProducer,' '), 'USA')>0) and index-of(tokenize(./@ism:disseminationControls,'
'), 'RELIDO')>0) then false() else true() "
flag="warning">
[ISM-ID-00124][Warning] Relido is not authorized for non-US portions.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00125

FileName:./Rules/generalConstraints/ISM_ID_00125.sch

Rule Description:

[ISM-ID-00125][Error] If any attributes in namespace urn:us:gov:ic:ism exist the 'local name' must exist in CVEnumISMAttributes. Human Readable: Ensure that no attributes that appear to be in the ISM namespace but are not defined by ISM.XML are used in a schema that might have an xsd:any Attribute.

Code Description:

This rule calls upon an abstract pattern to perform the logic work. It makes sure that every specified ISM attribute is a value found in the CVEnumISMAttributes.xml file.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00125">

<sch:rule context="//*[@ism:*]">
<!-- Define variables -->
<sch:let name="errFlag_ValueNotFound" value="error"/>
<sch:let name="errMsg_ValueNotFound"
value="' [ISM-ID-00125][Error] Ensure that no attributes that appear to be in the ISM
namespace but are not defined by ISM.XML are used in a schema that might have an xsd:any
Attribute. '"/>

<sch:let name="capco" value="false()"/>
<sch:let name="dataFileElems" value="$validAttributeList"/>
<sch:let name="attrValues" value="string-join(./@ism:*/local-name(),' ')/>
<sch:let name="attrValueTokens" value="tokenize($attrValues,' ')/>

<!-- Get the position of each client node relative to its position in the master list. If
the node is not found, return a -1 -->
<sch:let name="orderNums"
value=" for $token in $attrValueTokens return if
($dataFileElems[matches($token,concat('^',text(),'$'))]) then
count(($dataFileElems[matches($token,concat('^',text(),'$'))])/preceding::* ) + 1 else -1"/>

<!-- Determine if the list has invalid values. If and only if it does, figure out which
ones are invalids -->
<sch:let name="hasInvalids" value="count(index-of($orderNums,-1)) > 0"/>
<sch:let name="invalidValues"
value=" if ($hasInvalids) then distinct-values( for $token in index-of($orderNums,-1)
return $attrValueTokens[$token] ) else null "/>
```



```

<!-- Determine if the list has duplicate values. If and only if it does, figure out which
ones are duplicates -->
<sch:let name="hasDups"
value="count(distinct-values($attrValueTokens)) != count($attrValueTokens)"/>
<sch:let name="dupValues"
value=" if ($hasDups) then distinct-values( for $token in $attrValueTokens return
if (count(index-of($attrValueTokens,$token)) > 1) then $attrValueTokens[index-
of($attrValueTokens,$token)[1]] else null ) else null "/>

<!-- Execute tests -->
<sch:assert test="if(not($ISM_CAPCO_RESOURCE) and $capco) then 1 else not($hasInvalids)"
flag="$errFlag_ValueNotFound">
<sch:value-of select="$errMsg_ValueNotFound"/>
Invalid value of [<sch:value-of select="$invalidValues"/>]</sch:assert>
<sch:assert test="not($hasDups)" flag="undefined">Duplicate values found [<sch:value-of
select="$dupValues"/>] for [<sch:value-of select="$attrValueTokens"/>] </sch:assert>
</sch:rule>
</sch:pattern>

```

Rule: ISM-ID-00126

FileName:./Rules/generalConstraints/ISM_ID_00126.sch

Rule Description:

[ISM-ID-00126][Error] If any elements in namespace urn:us:gov:ic:ism exist. Human Readable: Ensure that no elements that appear to be in the ISM namespace but are not defined by ISM.XML are used in a schema that might have an xsd:any.

Code Description:

The code checks that any element having ISM attributes does not have the attribute xsd:any specified.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00126">

  <sch:rule context="//*[@ism:*]">
    <sch:assert id="ism00126" test=" not(./@xsd:any) " flag="error">
      [ISM-ID-00126][Error] Ensure that no elements that appear to be in the ISM namespace but
      are not
      defined by ISM.XML are used in a schema that might have an xsd:any.
    </sch:assert>
  </sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00127

FileName:../Rules/notice/ISM_ID_00127.sch

Rule Description:

[ISM-ID-00127][Error] If ISM-CAPCO-RESOURCE and: 1. Any element meeting ISM-CONTRIBUTES in the document has the attribute atomicEnergyMarkings containing [RD] AND 2. No element meeting ISM-CONTRIBUTES in the document has notice containing [RD].

Code Description:

If CAPCO rules do not apply to the document and the element is not excluded from the rollup then the rule does not apply and we return true. If the current element has attribute atomicEnergyMarkings specified with a value containing [RD], then we make sure that attribute notice is specified with a value containing [RD] in one of the portions of the document.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00127">

<sch:rule context="//*[@ism:atomicEnergyMarkings]">
<sch:assert id="ism00127"
test=" if(not($ISM_CAPCO_RESOURCE)or ./@ism:excludeFromRollup=true()) then true()
else if(index-of(tokenize(./@ism:atomicEnergyMarkings, ' '), 'RD')>0) then index-
of($partNotice, 'RD')>0 else true() "
flag="error">
[ISM-ID-00127][Error] If ISM-CAPCO-RESOURCE and:
1. Any element meeting ISM-CONTRIBUTES in the document has the attribute
atomicEnergyMarkings containing [RD]
AND
2. No element meeting ISM-CONTRIBUTES in the document has notice containing [RD].
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00128

FileName:./Rules/notice/ISM_ID_00128.sch

Rule Description:

[ISM-ID-00128][Error] If ISM-CAPCO-RESOURCE and: 1. Any element meeting ISM-CONTRIBUTES in the document has the attribute atomicEnergyMarkings containing [FRD] AND 2. No element meeting ISM-CONTRIBUTES in the document has notice containing [FRD]

Code Description:

If CAPCO rules do not apply to the document and the element is not excluded from the rollup then the rule does not apply and we return true. If the current element has attribute atomicEnergyMarkings specified with a value containing [FRD], then we make sure that attribute notice is specified with a value containing [FRD] in one of the portions of the document.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00128">

<sch:rule context="//*[@ism:atomicEnergyMarkings]">
<sch:assert id="ism00128"
test=" if(not($ISM_CAPCO_RESOURCE) or ./@ism:excludeFromRollup=true()) then true()
else if(index-of(tokenize(./@ism:atomicEnergyMarkings, ' '), 'FRD')>0) then index-
of($partNotice, 'FRD')>0 else true() "
flag="error">
[ISM-ID-00128][Error] If ISM-CAPCO-RESOURCE and:
1. Any element meeting ISM-CONTRIBUTES in the document has the attribute
atomicEnergyMarkings containing [FRD]
AND
2. No element meeting ISM-CONTRIBUTES in the document has notice containing [FRD]
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00129

FileName:../Rules/notice/ISM_ID_00129.sch

Rule Description:

[ISM-ID-00129][Error] If ISM-CAPCO-RESOURCE and: 1. Any element meeting ISM-CONTRIBUTES in the document has the attribute disseminationControls containing [IMC] AND 2. No element meeting ISM-CONTRIBUTES in the document has notice containing [IMC]

Code Description:

If CAPCO rules do not apply to the document and the element is not excluded from the rollup then the rule does not apply and we return true. If the current element has attribute disseminationControls specified with a value containing [IMC], then we make sure that attribute notice is specified with a value containing [IMC] in one of the portions of the document.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00129">

<sch:rule context="//*[@ism:disseminationControls]">
<sch:assert id="ism00129"
test=" if(not($ISM_CAPCO_RESOURCE) or ./@ism:excludeFromRollup=true()) then true()
else if(index-of(tokenize(./@ism:disseminationControls, ' '), 'IMC')>0) then index-
of($partNotice, 'IMC')>0 else true() "
flag="error">
[ISM-ID-00129][Error] If ISM-CAPCO-RESOURCE and:
1. Any element meeting ISM-CONTRIBUTES in the document has the attribute
disseminationControls containing [IMC]
AND
2. No element meeting ISM-CONTRIBUTES in the document has notice containing [IMC]
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00130

FileName:./Rules/notice/ISM_ID_00130.sch

Rule Description:

[ISM-ID-00130][Error] If ISM-CAPCO-RESOURCE and: 1. Any element meeting ISM-CONTRIBUTES in the document has the attribute disseminationControls containing [FISA] AND 2. No element meeting ISM-CONTRIBUTES in the document has notice containing [FISA]

Code Description:

If CAPCO rules do not apply to the document and the element is not excluded from the rollup then the rule does not apply and we return true. If the current element has attribute disseminationControls specified with a value containing [FISA], then we make sure that attribute notice is specified with a value containing [FISA] in one of the portions of the document.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00130">

<sch:rule context="//*[@ism:disseminationControls]">
<sch:assert id="ism00130"
test=" if(not($ISM_CAPCO_RESOURCE) or ./@ism:excludeFromRollup=true()) then true()
else if(index-of(tokenize(./@ism:disseminationControls, ' '), 'FISA')>0) then index-
of($partNotice, 'FISA')>0 else true() "
flag="error">
[ISM-ID-00130][Error] If ISM-CAPCO-RESOURCE and:
1. Any element meeting ISM-CONTRIBUTES in the document has the attribute
disseminationControls containing [FISA]
AND
2. No element meeting ISM-CONTRIBUTES in the document has notice containing [FISA]
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00132

FileName:./Rules/resourceElement/ISM_ID_00132.sch

Rule Description:

[ISM-ID-00132][Error] If ISM-CAPCO-RESOURCE and the ISM-RESOURCE-ELEMENT has the attribute disseminationControls containing [RELIDO] then every element meeting ISM-CONTRIBUTES-CLASSIFIED in the document must have the attribute disseminationControls containing [RELIDO]. Human Readable: USA documents having RELIDO at the resource level must have every classified portion having RELIDO.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If the resourceElement has attribute disseminationControls specified with a value containing [RELIDO], then we make sure that the number of elements in the document that have attribute classification specified with a value other than [U] and attribute disseminationControls specified with a value containing [RELIDO] is the same as the number of elements in the document that have attribute classification specified with a value other than [U].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00132">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00132"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(index-
of($bannerDisseminationControls_tok, 'RELIDO')>0) then sum(for $token in
$partTags return if(not($token/@ism:classification='U') and index-of(tokenize($token/
@ism:disseminationControls,' '), 'RELIDO')>0) then 1 else 0 ) = count(for $tag in
$partTags return if($tag/@ism:classification='U') then null else 1) else true() "
flag="error">
[ISM-ID-00132][Error] USA documents having RELIDO at the resource level must have every
classified portion having RELIDO.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00133

FileName:./Rules/declassException/ISM_ID_00133.sch

Rule Description:

[ISM-ID-00133][Error] If ISM-NSI-EO-APPLIES and attribute declassException is specified and does contain the name token [25X1-human], then attribute declassDate or declassEvent must NOT be specified.

Code Description:

If current Classified National Security Information Executive Order does not apply to this resource then the rule does not apply and we return true. Otherwise we ensure that any element with the attribute declassException specified and having a value of [25X1-human] then we ensure that the attribute declassDate or attribute declassEvent are not specified.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00133">

<sch:rule context="//*[@ism:declassException]">
<sch:let name="dd" value="if(./@ism:declassDate) then true() else false()"/>
<sch:let name="de" value="if(./@ism:declassEvent) then true() else false()"/>
<sch:assert id="ism00133"
test=" if(not($ISM_NSI_EO_APPLIES)) then true() else if(not(index-of(./
@ism:declassException, '25X1-human')>0)) then true() else not($dd or $de) "
flag="error">
[ISM-ID-00133][Error] If ISM-NSI-EO-APPLIES and attribute
declassException is specified and does contain the name token [25X1-human],
then attributes declassDate or declassEvent must NOT be specified. Invalid
presence of <sch:value-of select="if($dd) then 'declassDate' else null"/>
<sch:value-of select="if($dd and $de) then ' and ' else null"/>
<sch:value-of select="if($de) then 'declassEvent' else null"/>.
</sch:assert>
</sch:rule>
</sch:pattern>
```


Rule: ISM-ID-00134

FileName:../Rules/notice/ISM_ID_00134.sch

Rule Description:

[ISM-ID-00134][Error] If ISM-CAPCO-RESOURCE and: 1. Any element meeting ISM-CONTRIBUTES in the document has the attribute nonICmarkings containing [DS] AND 2. No element meeting ISM-CONTRIBUTES in the document has the attribute notice containing [DS]

Code Description:

If CAPCO rules do not apply to the document and the element is not excluded from the rollup then the rule does not apply and we return true. If the current element has attribute nonICmarkings specified with a value containing [DS], then we make sure that attribute notice is specified with a value containing [DS] in one of the portions of the document.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00134">

<sch:rule context="//*[@ism:nonICmarkings]">
<sch:assert id="ism00134"
test=" if(not($ISM_CAPCO_RESOURCE) or ./@ism:excludeFromRollup=true()) then true() else
if(index-of(tokenize(./@ism:nonICmarkings, ' '), 'DS')>0) then index-of($partNotice,
'DS')>0 else true() "
flag="error">
[ISM-ID-00134][Error] If ISM-CAPCO-RESOURCE and:
1. Any element meeting ISM-CONTRIBUTES in the document has the attribute nonICmarkings
containing [DS]
AND
2. No element meeting ISM-CONTRIBUTES in the document has the attribute notice containing
[DS]
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00135

FileName:../Rules/notice/ISM_ID_00135.sch

Rule Description:

[ISM-ID-00135][Warning] If ISM-CAPCO-RESOURCE and: 1. No element meeting ISM-CONTRIBUTES in the document has the attribute atomicEnergyMarkings containing [RD] AND 2. Any element meeting ISM-CONTRIBUTES in the document has the attribute notice containing [RD]

Code Description:

If CAPCO rules do not apply to the document and the element is not excluded from the rollup then the rule does not apply and we return true. If the current element is the resourceElement then the rule does not apply and we return true. If the current element has attribute notice specified with a value containing [RD], then we make sure that attribute atomicEnergyMarkings is specified with a value containing [RD] in one of the portions of the document.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00135">

<sch:rule context="//*[@ism:notice]">
<sch:include href="../../../_Properties/myPosition.sch"/>
<sch:assert id="ism00135"
test=" if(not($ISM_CAPCO_RESOURCE) or ./@ism:excludeFromRollup=true()) then true() else
if($myPosition = $resourceElementPosition) then true() else if(index-of(tokenize(./
@ism:notice, ' '), 'RD')>0 and not(./@ism:excludeFromRollup=true())) then index-
of($partAtomicEnergyMarkings, 'RD')>0 else true() "
flag="warning">
[ISM-ID-00135][Warning] If ISM-CAPCO-RESOURCE and:
1. No element meeting ISM-CONTRIBUTES in the document has the attribute
atomicEnergyMarkings containing [RD]
AND
2. Any element meeting ISM-CONTRIBUTES in the document has the attribute notice containing
[RD]
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00136

FileName:./Rules/notice/ISM_ID_00136.sch

Rule Description:

[ISM-ID-00136][Warning] If ISM-CAPCO-RESOURCE and: 1. No element without ism:excludeFromRollup=true() in the document has the attribute atomicEnergyMarkings containing [FRD] AND 2. Any element without ism:excludeFromRollup=true() in the document has the attribute notice containing [FRD]

Code Description:

If CAPCO rules do not apply to the document and the element is not excluded from the rollup then the rule does not apply and we return true. If the current element has attribute notice specified with a value containing [FRD], then we make sure that attribute atomicEnergyMarkings is specified with a value containing [FRD] in one of the portions of the document.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00136">

<sch:rule context="//*[@ism:notice]">
<sch:include href="../../../_Properties/myPosition.sch"/>
<sch:assert id="ism00136"
test=" if(not($ISM_CAPCO_RESOURCE) or ./@ism:excludeFromRollup=true()) then
true() else if(index-of(tokenize(./@ism:notice, ' '), 'FRD')>0 and not(./
@ism:excludeFromRollup=true())) then index-of($partAtomicEnergyMarkings, 'FRD')>0 else
true() "
flag="warning">
[ISM-ID-00136][Warning] If ISM-CAPCO-RESOURCE and:
1. No element without ism:excludeFromRollup=true() in the document has the attribute
atomicEnergyMarkings containing [FRD]
AND
2. Any element without ism:excludeFromRollup=true() in the document has the attribute
notice containing [FRD]
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00137

FileName:./Rules/notice/ISM_ID_00137.sch

Rule Description:

[ISM-ID-00137][Warning] If ISM-CAPCO-RESOURCE and: 1. No element without ism:excludeFromRollup=true() in the document has the attribute disseminationControls containing [IMC] AND 2. Any element without ism:excludeFromRollup=true() in the document has the attribute notice containing [IMC]

Code Description:

If CAPCO rules do not apply to the document and the element is not excluded from the rollup then the rule does not apply and we return true. If the current element has attribute notice specified with a value containing [IMC] and is not excluded from the rollup, then we make sure that attribute disseminationControls is specified with a value containing [IMC] in one of the portions of the document.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00137">

<sch:rule context="//*[@ism:notice]">
<sch:include href="../../../_Properties/myPosition.sch"/>
<sch:assert id="ism00137"
test=" if(not($ISM_CAPCO_RESOURCE) or ./@ism:excludeFromRollup=true()) then
true() else if(index-of(tokenize(./@ism:notice, ' '), 'IMC')>0 and not(./
@ism:excludeFromRollup=true())) then index-of($partDisseminationControls, 'IMC')>0 else
true() "
flag="warning">
[ISM-ID-00137][Warning] If ISM-CAPCO-RESOURCE and:
1. No element without ism:excludeFromRollup=true() in the document has the attribute
disseminationControls containing [IMC]
AND
2. Any element without ism:excludeFromRollup=true() in the document has the attribute
notice containing [IMC]
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00138

FileName:./Rules/notice/ISM_ID_00138.sch

Rule Description:

[ISM-ID-00138][Warning] If ISM-CAPCO-RESOURCE and: 1. No element without ism:excludeFromRollup=true() in the document has the attribute nonICmarkings containing [DS] AND 2. Any element without ism:excludeFromRollup=true() in the document has the attribute notice containing [DS]

Code Description:

If CAPCO rules do not apply to the document and the element is not excluded from the rollup then the rule does not apply and we return true. If the current element has attribute notice specified with a value containing [DS] and is not excluded from the rollup, then we make sure that attribute nonICmarkings is specified with a value containing [DS] in one of the portions of the document.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00138">

<sch:rule context="//*[@ism:notice]">
<sch:include href="../../../_Properties/myPosition.sch"/>
<sch:assert id="ism00138"
test=" if(not($ISM_CAPCO_RESOURCE) or ./@ism:excludeFromRollup=true()) then
true() else if(index-of(tokenize(./@ism:notice, ' '), 'DS')>0 and not(./
@ism:excludeFromRollup=true())) then index-of($partNonICmarkings, 'DS')>0 else true() "
flag="warning">
[ISM-ID-00138][Warning] If ISM-CAPCO-RESOURCE and:
1. No element without ism:excludeFromRollup=true() in the document has the attribute
nonICmarkings containing [DS]
AND
2. Any element without ism:excludeFromRollup=true() in the document has the attribute
notice containing [DS]
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00139

FileName:./Rules/notice/ISM_ID_00139.sch

Rule Description:

[ISM-ID-00139][Warning] If ISM-CAPCO-RESOURCE and: 1. No element without ism:excludeFromRollup=true() in the document has the attribute disseminationControls containing [FISA] AND 2. Any element without ism:excludeFromRollup=true() in the document has the attribute notice containing [FISA]

Code Description:

If CAPCO rules do not apply to the document and the element is not excluded from the rollup then the rule does not apply and we return true. If the current element has attribute notice specified with a value containing [FISA] and is not excluded from the rollup, then we make sure that attribute disseminationControls is specified with a value containing [FISA] in one of the portions of the document.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00139">

<sch:rule context="//*[@ism:notice]">
<sch:include href="../../../_Properties/myPosition.sch"/>
<sch:assert id="ism00139"
test=" if(not($ISM_CAPCO_RESOURCE) or ./@ism:excludeFromRollup=true()) then
true() else if(index-of(tokenize(./@ism:notice, ' '), 'FISA')>0 and not(./
@ism:excludeFromRollup=true())) then index-of($partDisseminationControls, 'FISA')>0
else true() "
flag="warning">
[ISM-ID-00139][Warning] If ISM-CAPCO-RESOURCE and:
1. No element without ism:excludeFromRollup=true() in the document has the attribute
disseminationControls containing [FISA]
AND
2. Any element without ism:excludeFromRollup=true() in the document has the attribute
notice containing [FISA]
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00140

FileName:./Rules/disseminationControls/ISM_ID_00140.sch

Rule Description:

[ISM-ID-00140][Error] If ISM-CAPCO-RESOURCE and attribute disseminationControls contains the name token [NF], then attribute classification must not have a value of [U] Human Readable: NF may not be used on Unclass portions.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If it is a CAPCO resource then we check the attribute classification of the element and return true if it has a value of [U]. Otherwise we check that the attribute disseminationControls does not contain the value [NF]

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00140">

<sch:rule context="//*[@ism:disseminationControls]">
<sch:assert id="ism00140"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(not(./@ism:classification='U'))
then true() else not(index-of(tokenize(./@ism:disseminationControls,' '), 'NF')>0) "
flag="error">
[ISM-ID-00140][Error] NF may not be used on Unclass portions.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00141

FileName:./Rules/resourceElement/ISM_ID_00141.sch

Rule Description:

[ISM-ID-00141][Error] If ISM-NSI-EO-APPLIES and 1. ISM-RESOURCE-ELEMENT attribute declassException does not have a value of "25X1-human" AND 2. ISM-RESOURCE-ELEMENT attribute declassDate is not specified AND 3. ISM-RESOURCE-ELEMENT attribute declassEvent is not specified AND 4. ISM-RESOURCE-ELEMENT attribute atomicEnergyMarkings is not specified with a value of [RD] or [FRD] Human Readable: declassDate or declassEvent are required unless 25X1-human, RD, or FRD is specified.

Code Description:

If the current Classified National Security Information Executive Order does not apply to the document then the rule does not apply and we return true. If the resourceElement has attribute declassException specified with a value of [25X1-human] then the rule does not apply and we return true. If the resourceElement has attribute atomicEnergyMarkings specified with a value containing [RD] or [FRD] then the rule does not apply and we return true. Otherwise, we make sure that the resourceElement has attribute declassDate specified or attribute declassEvent specified.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00141">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00141"
test=" if(not($ISM_NSI_EO_APPLIES)) then true() else if(matches($ISM_RESOURCE_ELEMENT/
@ism:declassException, '25X1-human')) then true() else if( sum( for $each in
tokenize($ISM_RESOURCE_ELEMENT/@ism:atomicEnergyMarkings, ' ') return if(matches($each,
'^F?RD-?.*$', ' ') then 1 else 0 )>0 ) then true() else ($ISM_RESOURCE_ELEMENT/
@ism:declassDate or $ISM_RESOURCE_ELEMENT/@ism:declassEvent) = ''
flag="error">
[ISM-ID-00141][Error] declassDate or declassEvent are required unless 25X1-human, RD, or
FRD is specified.
</sch:assert>
</sch:rule>
</sch:pattern>
```


Rule: ISM-ID-00142

FileName:./Rules/classification/ISM_ID_00142.sch

Rule Description:

[ISM-ID-00142][Error] If ISM-NSI-EO-APPLIES and attribute classification has a value other than [U] then attribute classifiedBy or derivativelyClassifiedBy must be specified on the ISM-RESOURCE-ELEMENT. Human Readable: Classified data including DOE data requires either an original classifier or a derivative classifier be indentified.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If it is a CAPCO resource then we check that if we have an element having attribute classification with a value of [U] then we return true because the rule does not apply. Otherwise we make sure that the resourceElement has the attribute classifiedBy or derivativelyClassifiedBy.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00142">

<sch:rule context="//*[@ism:classification]">
<sch:assert id="ism00142"
test=" if(not($ISM_NSI_EO_APPLIES)) then true() else if(./@ism:classification='U')
then true() else ($ISM_RESOURCE_ELEMENT/@ism:classifiedBy or $ISM_RESOURCE_ELEMENT/
@ism:derivativelyClassifiedBy) "
flag="error">
[ISM-ID-00142][Error] Classified data including DOE data requires either an original
classifier or a derivtive classifier be indentified.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00143

FileName:./Rules/derivativelyClassifiedBy/ISM_ID_00143.sch

Rule Description:

[ISM-ID-00143][Error] If ISM-CAPCO-RESOURCE and attribute derivativelyClassifiedBy is specified, then attribute derivedFrom must be specified. Human Readable: Derivatively Classified data including DOE data requires a derived from value to be identified.

Code Description:

If CAPCO rules do not apply to the resource then the rule does not apply and we return true. If it is a CAPCO resource then we check that if we have an element having attribute derivativelyClassifiedBy then we also have the attribute derivedFrom on the same element.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00143">

<sch:rule context="//*[@ism:derivativelyClassifiedBy]">
<sch:assert id="ism00143"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else ./@ism:derivedFrom "
flag="error">
[ISM-ID-00143][Error] Derivatively Classified data including DOE data requires a derived
from value to be identified.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00145

FileName:./Rules/resourceElement/ISM_ID_00145.sch

Rule Description:

[ISM-ID-00145][Error] If ISM-CAPCO-RESOURCE and any element in the document: 1. Meets ISM-CONTRIBUTES AND 2. Has the attribute nonICmarkings containing [LES] AND 3. No element meeting ISM-CONTRIBUTES in the document has nonICmarkings containing any of [LES-NF] Then the ISM-RESOURCE-ELEMENT must have nonICmarkings containing [LES]. Human Readable: Documents having LES and not having LES-NF must have LES at the resource level.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If any element has attribute nonICmarkings specified with a value containing [LES] and no element has attribute nonICmarkings specified with a value containing [LES-NF], then we make sure that the resourceElement has attribute nonICmarkings specified with a value containing [LES].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00145">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00145"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(index-of($partNonICmarkings_tok,
'LES') > 0 and not(index-of($partNonICmarkings_tok, 'LES-NF') > 0)) then (index-
of($bannerNonICmarkings, 'LES') > 0) else true() "
flag="error">
[ISM-ID-00145][Error] Documents having LES and not having LES-NF must have LES at the
resource level.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00146

FileName:./Rules/resourceElement/ISM_ID_00146.sch

Rule Description:

[ISM-ID-00146][Error] If ISM-CAPCO-RESOURCE and there exist at least 2 elements in the document: 1. Each element: Meets ISM-CONTRIBUTES AND 2. One of the elements: Has the attribute nonICmarkings containing [LES-NF] AND 3. One of the elements: meets ISM-CONTRIBUTES-CLASSIFIED Then the ISM-RESOURCE-ELEMENT must have disseminationControls containing [NF]. Human Readable: Classified USA documents having LES-NF Data must have NF at the resource level.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If any element has attribute nonICmarkings specified with a value containing [LES-NF] and the resourceElement has attribute classification specified with a value other than [U], then we make sure that the resourceElement has attribute disseminationControls specified with a value containing [NF].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00146">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00146"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(index-of($partNonICmarkings_tok,
'LES-NF') > 0 and not($bannerClassification='U')) then (index-
of($bannerDisseminationControls_tok, 'NF') > 0) else true() "
flag="error">
[ISM-ID-00146][Error] Classified USA documents having LES-NF Data must have NF at the
resource level.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00147

FileName:./Rules/resourceElement/ISM_ID_00147.sch

Rule Description:

[ISM-ID-00147][Error] If ISM-CAPCO-RESOURCE and there exist at least 2 elements in the document: 1. Each element: Meets ISM-CONTRIBUTES AND 2. One of the elements: Has the attribute nonICmarkings containing [LES-NF] AND 3. One of the elements: meets ISM-CONTRIBUTES-CLASSIFIED Then the ISM-RESOURCE-ELEMENT must have nonICmarkings containing [LES]. Human Readable: Classified USA documents having LES-NF Data must have LES at the resource level.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If any element has attribute nonICmarkings specified with a value containing [LES-NF] and the resourceElement has attribute classification specified with a value other than [U], then we make sure that the resourceElement has attribute nonICmarkings specified with a value containing [LES].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00147">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00147"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(index-of($partNonICmarkings_tok,
'LES-NF') > 0 and not($bannerClassification='U')) then (index-
of($bannerNonICmarkings_tok, 'LES') > 0) else true() "
flag="error">
[ISM-ID-00147][Error] Classified USA documents having LES-NF Data must have LES at the
resource level.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00148

FileName:./Rules/nonICmarkings/ISM_ID_00148.sch

Rule Description:

[ISM-ID-00148][Error] If ISM-CAPCO-RESOURCE, then Name tokens [LES] and [LES-NF] are mutually exclusive for attribute nonICmarkings.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If it is a CAPCO resource then we check that the attribute nonICmarkings does not contain both a value of [LES] and a value of [LES-NF].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00148">

<sch:rule context="//*[@ism:nonICmarkings]">
<!-- get list of tokens in nonICmarkings attribute -->
<sch:let name="nicmTok" value="tokenize(./@ism:nonICmarkings, ' ')" />

<sch:assert id="ism00148"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else not(index-of($nicmTok,'LES')>0 and
index-of($nicmTok,'LES-NF')>0) "
flag="error">
[ISM-ID-00148][Error] If ISM-CAPCO-RESOURCE, then tokens [LES] and [LES-NF] are mutually
exclusive for attribute nonICmarkings.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00149

FileName:./Rules/resourceElement/ISM_ID_00149.sch

Rule Description:

[ISM-ID-00149][Error] If ISM-CAPCO-RESOURCE and 1. Any element in the document meets ISM-CONTRIBUTES in the document has the attribute nonICmarkings contain [LES-NF] AND 2. ISM-RESOURCE-ELEMENT has the attribute classification [U] THEN the ISM-RESOURCE-ELEMENT must have nonICmarkings containing [LES-NF] Human Readable: Unclassified documents having LES-NF must have LES-NF at the resource level.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If any element has attribute nonICmarkings specified with a value containing [LES-NF] and the resourceElement has attribute classification specified with a value of [U], then we make sure that the resourceElement has attribute nonICmarkings specified with a value containing [LES-NF].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00149">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00149"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(index-of($partNonICmarkings_tok,
'LES-NF') > 0 and $bannerClassification='U') then (index-of($bannerNonICmarkings, 'LES-
NF') > 0) else true() "
flag="error">
[ISM-ID-00149][Error] Unclassified documents having LES-NF data must have LES-NF at the
resource level.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00150

FileName:./Rules/notice/ISM_ID_00150.sch

Rule Description:

[ISM-ID-00150][Error] If ISM-CAPCO-RESOURCE and: 1. Any element meeting ISM-CONTRIBUTES in the document has the attribute nonICmarkings containing [LES] AND 2. No element meeting ISM-CONTRIBUTES in the document has notice containing [LES]

Code Description:

If CAPCO rules do not apply to the document or the element is excluded from the rollup then the rule does not apply and we return true. If the current element has attribute nonICmarkings specified with a value containing [LES], then we make sure that attribute notice is specified with a value containing [LES] in one of the portions of the document.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00150">

<sch:rule context="//*[@ism:nonICmarkings and not(@ism:resourceElement=true())]">
<sch:assert id="ism00150"
test=" if(not($ISM_CAPCO_RESOURCE) or ./@ism:excludeFromRollup=true()) then true() else
if(index-of(tokenize(./@ism:nonICmarkings, ' '), 'LES')>0) then index-of($partNotice,
'LES')>0 else true() "
flag="error">
[ISM-ID-00150][Error] If ISM-CAPCO-RESOURCE and:
1. Any element meeting ISM-CONTRIBUTES in the document has the attribute nonICmarkings
containing [LES]
AND
2. No element meeting ISM-CONTRIBUTES in the document has notice containing [LES]
</sch:assert>
</sch:rule>
</sch:pattern>
```


Rule: ISM-ID-00151

FileName:./Rules/notice/ISM_ID_00151.sch

Rule Description:

[ISM-ID-00151][Warning] If ISM-CAPCO-RESOURCE and: 1. No element without ism:excludeFromRollup=true() in the document has the attribute nonICmarkings containing [LES] AND 2. Any element without ism:excludeFromRollup=true() in the document has the attribute notice containing [LES]

Code Description:

If CAPCO rules do not apply to the document and the element is not excluded from the rollup then the rule does not apply and we return true. If the current element has attribute notice specified with a value containing [LES] and it is included in the rollup, then we make sure that attribute nonICmarkings is specified with a value containing [LES] in one of the portions of the document.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00151">

<sch:rule context="//*[@ism:notice]">
<sch:include href="../../../_Properties/myPosition.sch"/>
<sch:assert id="ism00151"
test=" if(not($ISM_CAPCO_RESOURCE) or ./@ism:excludeFromRollup=true()) then
true() else if(index-of(tokenize(./@ism:notice, ' '), 'LES')>0 and not(./
@ism:excludeFromRollup=true())) then index-of($partNonICmarkings, 'LES')>0 else true()
"
flag="warning">
[ISM-ID-00151][Warning] If ISM-CAPCO-RESOURCE and:
1. No element without ism:excludeFromRollup=true() in the document has the attribute
nonICmarkings containing [LES]
AND
2. Any element without ism:excludeFromRollup=true() in the document has the attribute
notice containing [LES]
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00152

FileName:../Rules/notice/ISM_ID_00152.sch

Rule Description:

[ISM-ID-00152][Error] If ISM-CAPCO-RESOURCE and: 1. Any element meeting ISM-CONTRIBUTES in the document has the attribute nonICmarkings containing [LES-NF] AND 2. No element meeting ISM-CONTRIBUTES in the document has notice containing [LES-NF]

Code Description:

If CAPCO rules do not apply to the document and the element is not excluded from the rollup then the rule does not apply and we return true. If the current element specifies attribute nonICmarkings with a value containing [LES-NF], then we make sure that attribute notice is specified with a value containing [LES-NF] in one of the portions of the document.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00152">

<sch:rule context="//*[@ism:nonICmarkings and not(@ism:resourceElement=true())]">
<sch:assert id="ism00152"
test=" if(not($ISM_CAPCO_RESOURCE) or ./@ism:excludeFromRollup=true()) then true()
else if(index-of(tokenize(./@ism:nonICmarkings, ' '), 'LES-NF')>0) then index-
of($partNotice, 'LES-NF')>0 else true() "
flag="error">
[ISM-ID-00152][Error] If ISM-CAPCO-RESOURCE and:
1. Any element meeting ISM-CONTRIBUTES in the document has the attribute nonICmarkings
containing [LES-NF]
AND
2. No element meeting ISM-CONTRIBUTES in the document has notice containing [LES-NF]
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00153

FileName:./Rules/notice/ISM_ID_00153.sch

Rule Description:

[ISM-ID-00153][Error] If ISM-CAPCO-RESOURCE and: 1. No element without ism:excludeFromRollup=true() in the document has the attribute nonICmarkings containing [LES-NF] AND 2. Any element without ism:excludeFromRollup=true() in the document has the attribute notice containing [LES-NF].

Code Description:

If CAPCO rules do not apply to the document and the element is not excluded from the rollup then the rule does not apply and we return true. If the current element has attribute notice specified with a value containing [LES-NF] and it is included in the rollup, then we make sure that attribute nonICmarkings is specified with a value containing [LES-NF] in one of the portions of the document.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00153">

<sch:rule context="//*[@ism:notice]">
<sch:include href="../../../_Properties/myPosition.sch"/>
<sch:assert id="ism00153"
test=" if(not($ISM_CAPCO_RESOURCE) or ./@ism:excludeFromRollup=true()) then
true() else if(index-of(tokenize(./@ism:notice, ' '), 'LES-NF')>0 and not(./
@ism:excludeFromRollup=true())) then index-of($partNonICmarkings, 'LES-NF')>0 else
true() "
flag="error">
[ISM-ID-00153][Error] If ISM-CAPCO-RESOURCE and:
1. No element without ism:excludeFromRollup=true() in the document has the attribute
nonICmarkings containing [LES-NF]
AND
2. Any element without ism:excludeFromRollup=true() in the document has the attribute
notice containing [LES-NF]
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00154

FileName:./Rules/resourceElement/ISM_ID_00154.sch

Rule Description:

[ISM-ID-00154][Error] If ISM-CAPCO-RESOURCE and 1. Attribute disseminationControls of ISM-RESOURCE-ELEMENT contains [FOUO] AND 2. Attribute compilationReason does not have a value then at least one element meeting ISM-CONTRIBUTES in the document must have a disseminationControls attribute contain [FOUO]. Human Readable: USA FOUO documents not using compilation must have FOUO data.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If the resourceElement has attribute disseminationControls specified with a value containing [FOUO] and the resourceElement has attribute compilationReason specified with an empty value, then we make sure that at least one element in the document has attribute disseminationControls specified with a value containing [FOUO].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00154">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00154"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if (index-
of($bannerDisseminationControls_tok, 'FOUO')>0 and not(string-length(normalize-space(./
@ism:compilationReason))>0)) then index-of($partDisseminationControls_tok, 'FOUO')>0
else true() "
flag="error">
[ISM-ID-00154][Error] USA FOUO documents not using compilation must have FOUO data.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00155

FileName:./Rules/resourceElement/ISM_ID_00155.sch

Rule Description:

[ISM-ID-00155][Error] If ISM-CAPCO-RESOURCE and 1. ISM-DoD5230.24Applies AND 2. Attribute notice of ISM-RESOURCE-ELEMENT does not contain one of [DoD-Dist-A], [DoD-Dist-B], [DoD-Dist-C], [DoD-Dist-D], [DoD-Dist-E], [DoD-Dist-F], or [DoD-Dist-X] Human Readable: All documents that claim compliance with DoD5230.24 must have a distribution statement for the entire document.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If ISM-DoD5230.24Applies does not apply to the document then the rule does not apply and we return true. Otherwise, we make sure that the resourceElement has attribute notice specified with a value containing [DoD-Dist-A], [DoD-Dist-B], [DoD-Dist-C], [DoD-Dist-D], [DoD-Dist-E], [DoD-Dist-F], or [DoD-Dist-X].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00155">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00155"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(not($ISM_DOD5230_24_APPLIES))
then true() else count( (if(index-of($bannerNotice_tok, 'DoD-Dist-A')>0) then 1 else
null, if(index-of($bannerNotice_tok, 'DoD-Dist-B')>0) then 1 else null, if(index-
of($bannerNotice_tok, 'DoD-Dist-C')>0) then 1 else null, if(index-of($bannerNotice_tok,
'DoD-Dist-D')>0) then 1 else null, if(index-of($bannerNotice_tok, 'DoD-Dist-E')>0)
then 1 else null, if(index-of($bannerNotice_tok, 'DoD-Dist-F')>0) then 1 else null,
if(index-of($bannerNotice_tok, 'DoD-Dist-X')>0) then 1 else null) )>0 "
flag="error">
[ISM-ID-00155][Error] All documents that claim compliance with DoD5230.24 must have a
distribution statement
for the entire document.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00156

FileName:./Rules/notice/ISM_ID_00156.sch

Rule Description:

[ISM-ID-00156][Error] If ISM-CAPCO-RESOURCE and: 1. The attribute notice contains on of the [DoD-Dist-B], [DoD-Dist-C], [DoD-Dist-D], [DoD-Dist-E], [DoD-Dist-F], or [DoD-Dist-X] AND 2. One of the attributes noticeDate and noticePOC is not specified. Human Readable: DoD distribution statements B, C, D ,E ,F, and X all require a Date and a POC.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If the current element has attribute notice specified with a value containing [DoD-Dist-B], [DoD-Dist-C], [DoD-Dist-D], [DoD-Dist-E], [DoD-Dist-F], or [DoD-Dist-X], then we make sure that attributes noticeDate and noticePOC are also specified on the resourceElement.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00156">

<sch:rule context="//*[@ism:notice]">
<!-- tokenize the element's notice attribute -->
<sch:let name="noticeTok" value="tokenize(./@ism:notice, ' ')" />

<sch:assert id="ism00156"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(index-of($noticeTok, 'DoD-Dist-
B')>0 or index-of($noticeTok, 'DoD-Dist-C')>0 or index-of($noticeTok, 'DoD-Dist-
D')>0 or index-of($noticeTok, 'DoD-Dist-E')>0 or index-of($noticeTok, 'DoD-Dist-
F')>0 or index-of($noticeTok, 'DoD-Dist-X')>0) then (./@ism:noticeDate and ./
@ism:noticePOC) else true() "
flag="error">
[ISM-ID-00156][Error] DoD distribution statements B, C, D ,E ,F, and X all require a Date
and a POC.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00157

FileName:./Rules/notice/ISM_ID_00157.sch

Rule Description:

[ISM-ID-00157][Error] If ISM-CAPCO-RESOURCE and: 1. The attribute notice contains one of the [DoD-Dist-B], [DoD-Dist-C], [DoD-Dist-D], or [DoD-Dist-E] AND 2. The attribute noticeReason is not specified. Human Readable: DoD distribution statements B, C, D , or E all require a reason.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If the current element has attribute notice specified with a value containing [DoD-Dist-B], [DoD-Dist-C], [DoD-Dist-D], [DoD-Dist-E], or [DoD-Dist-F], then we make sure that attribute noticeReason is also specified on the resourceElement.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00157">

<sch:rule context="//*[@ism:notice]">
<!-- tokenize the notice attribute -->
<sch:let name="noticeTok" value="tokenize(./@ism:notice, ' ')" />

<sch:assert id="ism00157"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if( count( (if(index-of($noticeTok,
'DoD-Dist-B')>0) then 1 else null, if(index-of($noticeTok, 'DoD-Dist-C')>0)
then 1 else null, if(index-of($noticeTok, 'DoD-Dist-D')>0) then 1 else null,
if(index-of($noticeTok, 'DoD-Dist-E')>0) then 1 else null) )=0 ) then true() else ./
@ism:noticeReason "
flag="error">
[ISM-ID-00157][Error] DoD distribution statements B, C, D , or E all require a reason.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00158

FileName:./Rules/notice/ISM_ID_00158.sch

Rule Description:

[ISM-ID-00158][Error] If ISM-CAPCO-RESOURCE and: 1. ISM-DoD5230.24Applies AND 2. attribute classification of ISM-RESOURCE-ELEMENT is not [U] AND 3. The attribute notice does not contain one of [DoD-Dist-B], [DoD-Dist-C], [DoD-Dist-D], [DoD-Dist-E], or [DoD-Dist-F]. Human Readable: All classified documents that claim compliance with DoD5230.24 must use one of DoD distribution statements B, C, D, E, or F.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If DoD-5230-24 does not apply then the rule does not apply and we return true. If the resource is Unclassified then the rule does not apply and we return true. Otherwise, we make sure that the resourceElement attribute notice does not contain a value of [DoD-Dist-B], [DoD-Dist-C], [DoD-Dist-D], [DoD-Dist-E], or [DoD-Dist-F].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00158">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00158"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(not($ISM_DOD5230_24_APPLIES))
then true() else if(./@ism:classification='U') then true() else if( count( (if(index-
of($bannerNotice_tok, 'DoD-Dist-B')>0) then 1 else null, if(index-of($bannerNotice_tok,
'DoD-Dist-C')>0) then 1 else null, if(index-of($bannerNotice_tok, 'DoD-Dist-D')>0)
then 1 else null, if(index-of($bannerNotice_tok, 'DoD-Dist-E')>0) then 1 else null,
if(index-of($bannerNotice_tok, 'DoD-Dist-F')>0) then 1 else null) )=0 ) then false()
else true() "
flag="error">
[ISM-ID-00158][Error] All classified documents that claim compliance with DoD5230.24 must
use one of DoD
distribution statements B, C, D, E, or F.
</sch:assert>
</sch:rule>
</sch:pattern>
```


Rule: ISM-ID-00159

FileName:../Rules/notice/ISM_ID_00159.sch

Rule Description:

[ISM-ID-00159][Error] If ISM-CAPCO-RESOURCE and: 1. attribute classification of ISM-RESOURCE-ELEMENT is not [U] AND 2. The attribute notice does contains [DoD-Dist-A]. Human Readable: Distribution statement A (Public Release) is forbidden on classified documents.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If the document is Unclassified then the rule does not apply and we return true. Otherwise, we check that the current element does not have attribute notice specified with a value containing [DoD-Dist-A].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00159">

<sch:rule context="//*[@ism:notice]">
<!-- tokenize the notice attribute -->
<sch:let name="noticeTok" value="tokenize(./@ism:notice, ' ')" />

<sch:assert id="ism00159"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if($bannerClassification='U') then
true() else not(index-of($noticeTok, 'DoD-Dist-A')>0) "
flag="error">
[ISM-ID-00159][Error] Distribution statement A (Public Release) is forbidden on classified
documents.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00160

FileName:./Rules/notice/ISM_ID_00160.sch

Rule Description:

[ISM-ID-00160][Error] If ISM-CAPCO-RESOURCE and: 1. The attribute notice of ISM-RESOURCE-ELEMENT does contain [DoD-Dist-A] AND 2. attribute disseminationControls contains any of [FOUO], [PR], [DSEN], OR [FISA] AND 3. attribute atomicEnergyMarkings contains any of [DCNI] or [UCNI]. Human Readable: Distribution statement A (Public Release) is incompatible with [FOUO], [PR], [DCNI], [UCNI], [DSEN], OR [FISA].

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If it is a CAPCO resource then we check that if the resourceElement has attribute notice containing a value of [DoD-Dist-A] that the resourceElement's attribute disseminationControls does not contain values [FOUO], [PR], [DSEN], or [FISA] and attribute atomicEnergyMarkings does not contain values [UCNI] or [DCNI].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00160">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00160"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(index-of($bannerNotice_tok, 'DoD-
Dist-A')>0) then count( (if(index-of($bannerDisseminationControls_tok, 'FOUO')>0)
then 1 else null, if(index-of($bannerDisseminationControls_tok, 'PR')>0) then 1
else null, if(index-of($bannerAtomicEnergyMarkings_tok, 'DCNI')>0) then 1 else
null, if(index-of($bannerAtomicEnergyMarkings_tok, 'UCNI')>0) then 1 else null,
if(index-of($bannerDisseminationControls_tok, 'DSEN')>0) then 1 else null, if(index-
of($bannerDisseminationControls_tok, 'FISA')>0) then 1 else null) )=0 else true() "
flag="error">
[ISM-ID-00160][Error] Distribution statement A (Public Release) is
incompatible with [FOUO], [PR], [DCNI], [UCNI], [DSEN], OR [FISA].
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00161

FileName:./Rules/notice/ISM_ID_00161.sch

Rule Description:

[ISM-ID-00161][Error] If ISM-CAPCO-RESOURCE and: 1. The attribute notice of ISM-RESOURCE-ELEMENT does contains [DoD-Dist-A] AND 2. attribute nonICmarkings contains any of [SINFO], [XD], [ND], [SBU], [SBU-NF], [LES], OR [LES-NF]. Human Readable: Distribution statement A (Public Release) is incompatible with [SINFO], [XD], [ND], [SBU], [SBU-NF], [LES], OR [LES-NF].

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If it is a CAPCO resource and the resourceElement has attribute notice specified with a value containing [DoD-Dist-A], then we make sure that the resourceElement's attribute nonICmarkings does not contain values [SINFO], [XD], [ND], [SBU], [SBU-NF], [LES], or [LES-NF].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00161">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00161"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(index-of($bannerNotice_tok,
'DoD-Dist-A')>0) then count( (if(index-of($bannerNonICmarkings_tok,
'SINFO')>0) then 1 else null, if(index-of($bannerNonICmarkings_tok, 'XD')>0)
then 1 else null, if(index-of($bannerNonICmarkings_tok, 'ND')>0) then 1
else null, if(index-of($bannerNonICmarkings_tok, 'SBU')>0) then 1 else
null, if(index-of($bannerNonICmarkings_tok, 'SBU-NF')>0) then 1 else null,
if(index-of($bannerNonICmarkings_tok, 'LES')>0) then 1 else null, if(index-
of($bannerNonICmarkings_tok, 'LES-NF')>0) then 1 else null) )=0 else true() "
flag="error">
[ISM-ID-00161][Error] Distribution statement A (Public Release) is incompatible with
[SINFO], [XD], [ND], [SBU], [SBU-NF], [LES], OR [LES-NF].
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00162

FileName:./Rules/resourceElement/ISM_ID_00162.sch

Rule Description:

[ISM-ID-00162][Error] If ISM-CAPCO-RESOURCE and 1. ISM-DoD5230.24Applies AND 2. Attribute notice of ISM-RESOURCE-ELEMENT contains more than one of [DoD-Dist-A], [DoD-Dist-B], [DoD-Dist-C], [DoD-Dist-D], [DoD-Dist-E], [DoD-Dist-F], or [DoD-Dist-X] Human Readable: All documents that claim compliance with DoD5230.24 must have only 1 distribution statement for the entire document.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If ISM-DoD5230.24Applies does not apply to the document then the rule does not apply and we return true. Otherwise, we make sure that the resourceElement has attribute notice specified with a value containing only one of [DoD-Dist-A], [DoD-Dist-B], [DoD-Dist-C], [DoD-Dist-D], [DoD-Dist-E], [DoD-Dist-F], or [DoD-Dist-X].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00162">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00155"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(not($ISM_DOD5230_24_APPLIES))
then true() else not(count( ( if(index-of($bannerNotice_tok, 'DoD-Dist-A')>0) then 1
else null, if(index-of($bannerNotice_tok, 'DoD-Dist-B')>0) then 1 else null, if(index-
of($bannerNotice_tok, 'DoD-Dist-C')>0) then 1 else null, if(index-of($bannerNotice_tok,
'DoD-Dist-D')>0) then 1 else null, if(index-of($bannerNotice_tok, 'DoD-Dist-E')>0)
then 1 else null, if(index-of($bannerNotice_tok, 'DoD-Dist-F')>0) then 1 else null,
if(index-of($bannerNotice_tok, 'DoD-Dist-X')>0) then 1 else null) )>1) "
flag="error">
[ISM-ID-00162][Error] All documents that claim compliance with DoD5230.24 must have only 1
distribution statement
for the entire document.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00163

FileName:../Rules/nonUSControls/ISM_ID_00163.sch

Rule Description:

[ISM-ID-00163][Error] If attribute nonUSControls exists the attribute ownerProducer must equal [NATO]. Human

Readable: NATO is the only nonUSControls currently authorized.

Code Description:

The code ensures that any element containing the attribute nonUSControls also has attribute ownerProducer specified with a value of [NATO].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00163">

<sch:rule context="//*[@ism:nonUSControls]">
<sch:assert id="ism00163" test="./@ism:ownerProducer='NATO'" flag="error">
[ISM-ID-00163][Error] NATO is the only nonUSControls currently authorized.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00164

FileName:../Rules/disseminationControls/ISM_ID_00164.sch

Rule Description:

[ISM-ID-00164][Error] If ISM-CAPCO-RESOURCE and attribute disseminationControls contains the name token [RS], then attribute classification must have a value of [TS] or [S].

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If it is a CAPCO resource then we check the attribute classification of the element and return true if it has a value of [S] or [TS]. Otherwise we check that the attribute disseminationControls does not contain the value [RS]

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00164">

<sch:rule context="//*[@ism:disseminationControls]">
<sch:let name="dissemTok" value="tokenize(./@ism:disseminationControls, ' ')" />
<sch:assert id="ism00164"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(matches(./
@ism:classification, '^(TS|S)$')) then true() else not(index-of($dissemTok, 'RS')>0) "
flag="error">
[ISM-ID-00164][Error] If ISM-CAPCO-RESOURCE and attribute
disseminationControls contains the name token [RS],
then attribute classification must have a value of [TS] or [S].
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00165

FileName:./Rules/resourceElement/ISM_ID_00165.sch

Rule Description:

[ISM-ID-00165][Error] If ISM-CAPCO-RESOURCE and any element meeting ISM-CONTRIBUTES in the document have the attribute disseminationControls containing [RS] then the ISM-RESOURCE-ELEMENT must have disseminationControls containing [RS]. Human Readable: USA documents having RS Data must have RS at the resource level

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If any element has attribute disseminationControls specified with a value containing [RS], then we make sure that the resourceElement has attribute disseminationControls specified with a value containing [RS].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00165">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:assert id="ism00165"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(index-
of($partDisseminationControls_tok, 'RS')>0) then index-
of($bannerDisseminationControls_tok, 'RS') > 0 else true() "
flag="error">
[ISM-ID-00165][Error] USA documents having RS Data must have RS at the resource level.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00166

FileName:../Rules/generalConstraints/ISM_ID_00166.sch

Rule Description:

[ISM-ID-00166][Warning] Attribute classification must not contain values that have passed their deprecation date.

Code Description:

Traverse the CVE file pulling out deprecated values and their dates. Using the current date, determine if the values are being used but it is still prior to the deprecated date

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00166">

  <!--
Any changes made to this code may also be required for the
following rule files:
- ISM_ID_00166.sch
- ISM_ID_00179.sch
- ISM_ID_00188.sch
- ISM_ID_00190.sch
- ISM_ID_00192.sch
- ISM_ID_00194.sch
- ISM_ID_00196.sch
- ISM_ID_00198.sch
- ISM_ID_00200.sch
- ISM_ID_00202.sch
- ISM_ID_00204.sch
- ISM_ID_00206.sch
- ISM_ID_00208.sch
- ISM_ID_00210.sch
-->
  <sch:rule context="//*[@ism:classification]">
    <sch:let name="curDate" value="translate($ISM_RESOURCE_ELEMENT/@ism:createDate,'-','')"/>
    <sch:let name="depValues"
value="document('._CVE/CVEnumISMClassificationAll.xml')//cve:CVE/cve:Enumeration/
cve:Term[./@deprecated]/cve:Value"/>
    <sch:let name="depDates"
value="document('._CVE/CVEnumISMClassificationAll.xml')//cve:CVE/cve:Enumeration/
cve:Term/@deprecated"/>

    <sch:let name="values"
value=" for $each in $depValues return if(index-of(tokenize(./@ism:classification,' '),
$each)>0) then $each else null"/>
    <sch:let name="reportWarn"
```



```
value=" for $each in $values return if($curDate <
translate(subsequence($depDates,index-of($depValues,$each),1),'-','')) then
concat('[' ,string($each),' ] has been deprecated and is not authorized for used after ' ,
string(subsequence($depDates,index-of($depValues,$each),1))) else null "/>

<sch:assert test="count($reportWarn)=0" flag="warning">
[ISM-ID-00166][Warning] For attribute classification, value(s) <sch:value-of
select="$reportWarn"/>
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00167

FileName:./Rules/displayOnlyTo/ISM_ID_00167.sch

Rule Description:

[ISM-ID-00167][Error] If ISM-CAPCO-RESOURCE and attribute displayOnlyTo is specified, then each of its values must be ordered in accordance with CVEnumISMRelTo.xml.

Code Description:

This rule calls upon an abstract pattern to perform the logic work. It makes sure that every value in the displayOnlyTo attributes are sorted in the order found in the CVEnumISMRelTo.xml file

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00167">

<sch:rule context="//*[@ism:displayOnlyTo]">
<!-- Define variables -->
<sch:let name="errFlag_AlphabeticalOrder" value="error"/>
<sch:let name="errMsg_AlphabeticalOrder"
value="' [ISM-ID-00167][Error] If ISM-CAPCO-RESOURCE and attribute displayOnlyTo is
specified, then each of its values must be ordered in accordance with CVEnumISMRelTo.xml.
' "/>

<sch:let name="dataFileElems" value="$displayOnlyToList"/>
<sch:let name="attrValues" value="./@ism:displayOnlyTo"/>
<sch:let name="attrValueTokens" value="tokenize($attrValues, ' ' )"/>

<!-- Get the position of each client node relative to its position in the master list. If
the node is not found, return a -1 -->
<sch:let name="orderNums"
value=" for $token in $attrValueTokens return if
($dataFileElems[matches($token,concat('^',text(),'$'))]) then
count(($dataFileElems[matches($token,concat('^',text(),'$'))]/preceding::*) + 1 else -1"/>

<!-- Create a sequence that returns a 0 if the previous sibling has a higher order number,
else return a 1 -->
<sch:let name="sortedOrderNums"
value=" for $peice in distinct-values(for $token in $orderNums return index-of($orderNums,
$token)) return if($peice>1) then if($orderNums[$peice] = $orderNums[$peice - 1]) then
if(compare($attrValueTokens[$peice - 1],$attrValueTokens[$peice]) = 1) then 0 else 2 else
if($orderNums[$peice] > $orderNums[$peice - 1]) then 1 else 0 else 1 "/>
<sch:let name="hasUnsorted" value="count(index-of($sortedOrderNums,0)) > 0"/>
<sch:let name="unsortedValues"
```

```
value=" if ($hasUnsorted) then distinct-values( for $token in index-of($sortedOrderNums,0)
return $attrValueTokens[$token] ) else null "/>

<sch:assert test="not($hasUnsorted)" flag="$errFlag_AlphabeticalOrder">
<sch:value-of select="$errMsg_AlphabeticalOrder"/>
The following values are out of order [<sch:value-of select="$unsortedValues"/>] for
[<sch:value-of select="$attrValueTokens"/>] </sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00168

FileName:../Rules/displayOnlyTo/ISM_ID_00168.sch

Rule Description:

[ISM-ID-00168][Error] If ISM-CAPCO-RESOURCE and attribute disseminationControls is not specified or is specified and does not contain the name token [DISPLAYONLY], then attribute displayOnlyTo must not be specified.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If it is a CAPCO resource then we check that if the attribute displayOnlyTo is specified then the attribute disseminationControls is also specified that it contains a value of [DISPLAYONLY]

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00168">

<sch:rule context="//*[@ism:displayOnlyTo]">

<sch:let name="dissemTok" value="tokenize(./@ism:disseminationControls,' ')" />
<sch:assert id="ism00168"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else index-
of($dissemTok, 'DISPLAYONLY')>0 "
flag="error">
[ISM-ID-00168][Error] If ISM-CAPCO-RESOURCE and attribute
disseminationControls is not specified or is specified and does not contain the name token
[DISPLAYONLY], then attribute displayOnlyTo must not be specified.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00169

FileName:./Rules/disseminationControls/ISM_ID_00169.sch

Rule Description:

[ISM-ID-00169][Error] If ISM-CAPCO-RESOURCE, and attribute disseminationControls contains name token [DISPLAYONLY] then tokens [RELIDO] and [NF] may not also be used.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If it is a CAPCO resource then we check that if we have an element having attribute disseminationControls with a value of [DISPLAYONLY] then it does not have a value of [RELIDO] or [NF] also.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00169">

<sch:rule context="//*[@ism:disseminationControls]">
<sch:let name="dissemTok" value="tokenize(./@ism:disseminationControls,' ')" />
<sch:assert id="ism00169"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(index-
of($dissemTok,'DISPLAYONLY')>0) then not(index-of($dissemTok,'RELIDO')>0 or index-
of($dissemTok,'NF')>0) else true() "
flag="error">
[ISM-ID-00169][Error] If ISM-CAPCO-RESOURCE, and attribute disseminationControls
contains name token [DISPLAYONLY] then tokens [RELIDO] and [NF] may not also be used.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00170

FileName:./Rules/generalConstraints/ISM_ID_00170.sch

Rule Description:

[ISM-ID-00170][Error] Attribute classification must not contain values that have passed their deprecation date.

Code Description:

Traverse the CVE file pulling out deprecated values and their dates. Using the current date determine if the values are being used passed the deprecated date

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00170">

  <!--
Any changes made to this code may also be required for the
following rule files:
- ISM_ID_00170.sch
- ISM_ID_00180.sch
- ISM_ID_00189.sch
- ISM_ID_00199.sch
- ISM_ID_00193.sch
- ISM_ID_00195.sch
- ISM_ID_00197.sch
- ISM_ID_00199.sch
- ISM_ID_00201.sch
- ISM_ID_00203.sch
- ISM_ID_00205.sch
- ISM_ID_00207.sch
- ISM_ID_00209.sch
- ISM_ID_00211.sch
-->
  <sch:rule context="//*[@ism:classification]">
    <sch:let name="curDate" value="translate($ISM_RESOURCE_ELEMENT/@ism:createDate,'-','')"/>
    <sch:let name="depValues"
value="document('._CVE/CVEnumISMClassificationAll.xml')//cve:CVE/cve:Enumeration/
cve:Term[./@deprecated]/cve:Value"/>
    <sch:let name="depDates"
value="document('._CVE/CVEnumISMClassificationAll.xml')//cve:CVE/cve:Enumeration/
cve:Term/@deprecated"/>

    <sch:let name="values"
value=" for $each in $depValues return if(index-of(tokenize(./@ism:classification,' '),
$each)>0) then $each else null"/>
    <sch:let name="reportErr"
```

```
value=" for $each in $values return if($curDate >=
translate(subsequence($depDates,index-of($depValues,$each),1),'-','))
then concat('[' ,string($each),'] is not authorized for use after ',
string(subsequence($depDates,index-of($depValues,$each),1))) else null " />

<sch:assert test="count($reportErr)=0" flag="error">
[ISM-ID-00170][Error] For attribute classification, value(s) <sch:value-of
select="$reportErr"/>
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00171

FileName:./Rules/resourceElement/ISM_ID_00171.sch

Rule Description:

[ISM-ID-00171][Warning] If ISM-CAPCO-RESOURCE and displayOnlyTo is specified on the resource element then all classified portion must be displayOnlyTo the same country Human Readable: USA documents having DISPLAYONLY Data at the resource level must have all classified portions authorized for DISPLAYONLY the same country

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If it is a CAPCO resource then we count the number of classified portions looping through \$partTags. Then we get a list of all country codes listed in displayOnlyTo attributes on those classified portions and tokenize them down so each node of the list is a single token. Then we count how many times we see each country code from the resource element. Then make sure that all codes present on the resourceElement have a count the same as the number of classified portions.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00171">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:let name="numClassParts"
value=" count( for $each in $partTags return if(matches($each/@ism:classification,'^(TS|S|
C)$')) then 1 else null ) "/>
<sch:let name="codes"
value=" for $each in $partTags return if(matches($each/@ism:classification,'^(TS|S|
C)$')) then distinct-values(tokenize(string-join(($each/@ism:displayOnlyTo, $each/
@ism:releasableTo),' '), ' ')) else null "/>
<sch:let name="code_tok" value="for $each in $codes return tokenize($each, ' ')" />
<sch:let name="codeCount"
value=" for $each in $bannerDisplayOnlyTo_tok return count( for $tok in $code_tok return
if($each=$tok) then $tok else null ) "/>

<sch:assert id="ism00171"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else count( for $each in $codeCount return
if($each = $numClassParts) then null else $each )=0 "
flag="warning">
[ISM-ID-00171][Warning]USA documents having DISPLAYONLY Data at the resource level
must have all classified portions authorized for DISPLAYONLY the same
country
</sch:assert>
</sch:rule>
</sch:pattern>
```


Rule: ISM-ID-00172

FileName:./Rules/resourceElement/ISM_ID_00172.sch

Rule Description:

[ISM-ID-00172][Warning] If ISM-CAPCO-RESOURCE and: There exist at least 1 element in the document that 1. Meets ISM-CONTRIBUTES AND 2. Has disseminationControls containing [DISPLAYONLY] AND 3. Does not share any countries in attribute displayOnlyTo or releasableTo with all of the other classified elements Then the ISM-RESOURCE-ELEMENT must NOT have disseminationControls containing [DISPLAYONLY]. Human Readable: The resourceElement must not have disseminationControls containing [DISPLAYONLY] if the document contains a mixture of authorized foreign disclosure/release decisions where there is no common country.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If it is a CAPCO resource then we count the number of classified portions looping through \$partTags. Then we get a list of all country codes listed in displayOnlyTo attributes on those classified portions and tokenize them down so each node of the list is a single token. Then we count how many times we see each country code on the resource element. Then if a code does not have the same count as the number of classified portions we check that disseminationControls on the resourceElement does not contain [DISPLAYONLY].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00172">

<sch:rule context="//*[@ism:resourceElement=true()]">
<sch:let name="numClassParts"
value=" count( for $each in $partTags return if(matches($each/@ism:classification,'^(TS|S|
C)$')) then 1 else null ) "/>
<sch:let name="codes"
value=" for $each in $partTags return if(matches($each/@ism:classification,'^(TS|S|
C)$')) then distinct-values(tokenize(string-join(($each/@ism:displayOnlyTo, $each/
@ism:releasableTo),' '), ' ')) else null "/>
<sch:let name="code_tok" value="for $each in $codes return tokenize($each, ' ')" />
<sch:let name="codeCount"
value=" for $each in $bannerDisplayOnlyTo_tok return count( for $tok in $code_tok return
if($each=$tok and $each!='USA') then $tok else null ) "/>
<sch:assert id="ism00172"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if( count($code_tok) = 0) then true()
else if( count( for $each in $codeCount return if($each = $numClassParts) then null else
$each )>0 ) then not(index-of($bannerDisseminationControls_tok, 'DISPLAYONLY')>0)
else true() "
flag="warning">
[ISM-ID-00172][Warning] The resourceElement must not have disseminationControls containing
[DISPLAYONLY]
if the document contains a mixture of authorized foreign disclosure/release
decisions where there is no common country.
</sch:assert>
</sch:rule>
```

</sch:pattern>

Rule: ISM-ID-00173

FileName:./Rules/atomicEnergyMarkings/ISM_ID_00173.sch

Rule Description:

[ISM-ID-00173][Error] If ISM-CAPCO-RESOURCE and attribute atomicEnergyMarkings contains a name token starting with [RD-SG] or [FRD-SG], then attribute classification must have a value of [TS], [S], or [C].

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If it is a CAPCO resource, and the element is classified (C, S or TS), then return true. Then we check that the attribute atomicEnergyMarkings has a value of [RD-SG] or [FRD-SG] with a single digit [1-9] or double digit [10-99]. If this it does contain one of those values then the token is returned and the count will be greater than 0.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00173">

<sch:rule context="//*[@ism:atomicEnergyMarkings]">
<sch:assert id="ism00173"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(not(./@ism:classification='U'))
then true() else count(for $token in tokenize(./@ism:atomicEnergyMarkings, ' ') return
if(matches($token, '^(F?RD-SG-[1-9]\d?)$')) then $token else null )=0 "
flag="error">
[ISM-ID-00173][Error] If ISM-CAPCO-RESOURCE and attribute
atomicEnergyMarkings contains a name token starting with [RD-SG] or [FRD-SG], then
attribute
classification must have a value of [TS], [S], or [C].
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00174

FileName:./Rules/atomicEnergyMarkings/ISM_ID_00174.sch

Rule Description:

[ISM-ID-00174][Error] If ISM-CAPCO-RESOURCE and attribute atomicEnergyMarkings contains the name token [RD] or [FRD], then attribute classification must have a value of [TS], [S], or [C].

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If it is a CAPCO resource then we check that if we have an element having attribute atomicEnergyMarkings with a value of [RD] or [FRD] then we also have the attribute classification specified with a value of [C], [S], or [TS] on the same element.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00174">

<sch:rule context="//*[@ism:atomicEnergyMarkings]">
<sch:assert id="ism00174"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else count( for $token in tokenize(./
@ism:atomicEnergyMarkings, ' ') return if(matches($token,'^(F?RD)$') and not( matches(./
@ism:classification,'^(TS|S|C)$')) then $token else null )=0 "
flag="error">
[ISM-ID-00174][Error] If ISM-CAPCO-RESOURCE and attribute
atomicEnergyMarkings contains the name token [RD] or [FRD],
then attribute classification must have a value of [TS], [S], or [C].
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00175

FileName:../Rules/atomicEnergyMarkings/ISM_ID_00175.sch

Rule Description:

[ISM-ID-00175][Error] If ISM-CAPCO-RESOURCE and attribute atomicEnergyMarkings contains the name token [RD-CNWDI], then attribute classification must have a value of [TS] or [S].

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If it is a CAPCO resource then we check that if we have an element having attribute atomicEnergyMarkings without a value of [RD-CNWDI] then we return true because the rule does not apply. Otherwise we make sure the attribute classification is specified with a value of [S] or [TS] on the same element.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00175">

<sch:rule context="//*[@ism:atomicEnergyMarkings]">
<sch:assert id="ism00175"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(not(index-of(tokenize(./
@ism:atomicEnergyMarkings,' '), 'RD-CNWDI')>0)) then true() else matches(./
@ism:classification, '^(TS|S)$') "
flag="error">
[ISM-ID-00175][Error] If ISM-CAPCO-RESOURCE and attribute
atomicEnergyMarkings contains the name token [RD-CNWDI], then attribute
classification must have a value of [TS] or [S].
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00176

FileName:../Rules/atomicEnergyMarkings/ISM_ID_00176.sch

Rule Description:

[ISM-ID-00176][Error] If ISM-CAPCO-RESOURCE and attribute atomicEnergyMarkings has a name token containing [RD] or [FRD], then attributes declassDate and declassEvent cannot be specified on the resourceElement. Human Readable: Automatic declassification of documents containing RD or FRD information is prohibited. Attributes declassDate and declassEvent cannot be used in the classification authority block when RD or FRD is present.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If it is a CAPCO resource then we check that if we have an element having attribute atomicEnergyMarkings with a value containing [RD] or [FRD] then we make sure that the resourceElement does not have attributes declassDate or declassEvent specified.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00176">

<sch:rule context="//*[@ism:atomicEnergyMarkings]">
<sch:assert id="ism00176"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else count( for $each in tokenize(./
@ism:atomicEnergyMarkings,' ') return if(matches($each, '^(F?RD)$')) then
if($ISM_RESOURCE_ELEMENT/@ism:declassDate or $ISM_RESOURCE_ELEMENT/@ism:declassEvent) then
1 else null else null )=0 "
flag="error">
[ISM-ID-00176][Error] Automatic declassification of documents containing RD or FRD
information is prohibited.
Attributes declassDate and declassEvent cannot be used in the classification authority
block when RD or FRD is present.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00177

FileName:./Rules/SCIcontrols/ISM_ID_00177.sch

Rule Description:

[ISM-ID-00177][Error] If ISM-CAPCO-RESOURCE and attribute SCIcontrols contains the name token [SI-ECI], then it must also contain the name token [SI].

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If the current element has attribute SCIcontrols specified with a value containing [SI-ECI], then we make sure that attribute SCIcontrols also contains the value [SI].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00177">

<sch:rule context="//*[@ism:SCIcontrols]">
<sch:assert id="ism00177"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if( count( for $each in tokenize(./
@ism:SCIcontrols,' ') return if(matches($each,'^SI-ECI')) then 1 else null )>0 ) then
index-of(tokenize(./@ism:SCIcontrols,' '), 'SI')>0 else true() "
flag="error">
[ISM-ID-00177][Error] If ISM-CAPCO-RESOURCE and attribute SCIcontrols contains the name
token [SI-ECI],
then it must also contain the name token [SI].
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00178

FileName:./Rules/atomicEnergyMarkings/ISM_ID_00178.sch

Rule Description:

[ISM-ID-00178][Error] If ISM-CAPCO-RESOURCE and attribute atomicEnergyMarkings is specified, then each of its values must be ordered in accordance with CVEnumISMAAtomicEnergyMarkings.xml.

Code Description:

This rule calls upon an abstract pattern to perform the logic work. It makes sure that every value in the atomicEnergyMarkings attribute are sorted in the order found in the CVEnumISMAAtomicEnergyMarkings.xml file

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00178">

<sch:rule context="//*[@ism:atomicEnergyMarkings]">
<!-- Define variables -->
<sch:let name="capcoRestriction" value="true()"/>
<sch:let name="errFlag_AlphabeticalOrder" value="error"/>
<sch:let name="errMsg_AlphabeticalOrder"
value=" '[ISM-ID-00178][Error] If ISM-CAPCO-RESOURCE and attribute atomicEnergyMarkings
is specified, then each of its values must be ordered in accordance with
CVEnumISMAAtomicEnergyMarkings.xml.' "/>

<sch:let name="dataFileElems" value="$atomicEnergyMarkingsList"/>
<sch:let name="attrValues" value="./@ism:atomicEnergyMarkings"/>
<sch:let name="attrValueTokens" value="tokenize($attrValues,' ')/>

<!-- Get the position of each client node relative to its position in the master list. If
the node is not found, return a -1 -->
<sch:let name="orderNums"
value=" for $token in $attrValueTokens return if
($dataFileElems[matches($token,concat('^',text(),'$'))]) then
count(($dataFileElems[matches($token,concat('^',text(),'$'))])/preceding::*) + 1 else -1"/>

<!-- Create a sequence that returns a 0 if the previous sibling has a higher order number,
else return a 1 -->
<sch:let name="sortedOrderNums"
value=" for $peice in distinct-values(for $token in $orderNums return index-of($orderNums,
$token)) return if($peice>1) then if($orderNums[$peice] = $orderNums[$peice - 1]) then
if(compare($attrValueTokens[$peice - 1],$attrValueTokens[$peice]) = 1) then 0 else 2 else
if($orderNums[$peice] > $orderNums[$peice - 1]) then 1 else 0 else 1 "/>
<sch:let name="hasUnsorted" value="count(index-of($sortedOrderNums,0)) > 0"/>
<sch:let name="unsortedValues"
```



```
value=" if ($hasUnsorted) then distinct-values( for $token in index-of($sortedOrderNums,0)
return $attrValueTokens[$token] ) else null "/>

<sch:assert test="not($hasUnsorted)" flag="$errFlag_AlphabeticalOrder">
<sch:value-of select="$errMsg_AlphabeticalOrder"/>
The following values are out of order [<sch:value-of select="$unsortedValues"/>] for
[<sch:value-of select="$attrValueTokens"/>] </sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00179

FileName:./Rules/generalConstraints/ISM_ID_00179.sch

Rule Description:

[ISM-ID-00179][Warning] Attribute disseminationControls must not contain values that have passed their deprecation date.

Code Description:

Traverse the CVE file pulling out deprecated values and their dates. Using the current date, determine if the values are being used but it is still prior to the deprecated date

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00179">

  <!--
Any changes made to this code may also be required for the
following rule files:
- ISM_ID_00166.sch
- ISM_ID_00179.sch
- ISM_ID_00188.sch
- ISM_ID_00190.sch
- ISM_ID_00192.sch
- ISM_ID_00194.sch
- ISM_ID_00196.sch
- ISM_ID_00198.sch
- ISM_ID_00200.sch
- ISM_ID_00202.sch
- ISM_ID_00204.sch
- ISM_ID_00206.sch
- ISM_ID_00208.sch
- ISM_ID_00210.sch
-->
  <sch:rule context="//*[@ism:disseminationControls]">
    <sch:let name="curDate" value="translate($ISM_RESOURCE_ELEMENT/@ism:createDate,'-','')"/>
    <sch:let name="depValues"
value="document('._CVE/CVEnumISMDissem.xml')//cve:CVE/cve:Enumeration/cve:Term[./
@deprecated]/cve:Value"/>
    <sch:let name="depDates"
value="document('._CVE/CVEnumISMDissem.xml')//cve:CVE/cve:Enumeration/cve:Term/
@deprecated"/>

    <sch:let name="values"
value=" for $each in $depValues return if(index-of(tokenize(./@ism:disseminationControls,'
'), $each)>0) then $each else null"/>
    <sch:let name="reportWarn"
```

```
value=" for $each in $values return if($curDate <
translate(subsequence($depDates,index-of($depValues,$each),1),'-',')) then
concat('[' ,string($each),' ] has been deprecated and is not authorized for used after ' ,
string(subsequence($depDates,index-of($depValues,$each),1))) else null "/>

<sch:assert test="count($reportWarn)=0" flag="warning">
[ISM-ID-00179][Warning] For attribute disseminationControls, value(s) <sch:value-of
select="$reportWarn"/>
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00180

FileName:./Rules/generalConstraints/ISM_ID_00180.sch

Rule Description:

[ISM-ID-00180][Error] Attribute disseminationControls must not contain values that have passed their deprecation date.

Code Description:

Traverse the CVE file pulling out deprecated values and their dates. Using the current date determine if the values are being used passed the deprecated date

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00180">

<!--
Any changes made to this code may also be required for the
following rule files:
- ISM_ID_00170.sch
- ISM_ID_00180.sch
- ISM_ID_00189.sch
- ISM_ID_00191.sch
- ISM_ID_00193.sch
- ISM_ID_00195.sch
- ISM_ID_00197.sch
- ISM_ID_00199.sch
- ISM_ID_00201.sch
- ISM_ID_00203.sch
- ISM_ID_00205.sch
- ISM_ID_00207.sch
- ISM_ID_00209.sch
- ISM_ID_00211.sch
-->
<sch:rule context="//*[@ism:disseminationControls]">
<sch:let name="curDate" value="translate($ISM_RESOURCE_ELEMENT/@ism:createDate,'-','')"/>
<sch:let name="depValues"
value="document('._CVE/CVEnumISMDissem.xml')//cve:CVE/cve:Enumeration/cve:Term[./
@deprecated]/cve:Value"/>
<sch:let name="depDates"
value="document('._CVE/CVEnumISMDissem.xml')//cve:CVE/cve:Enumeration/cve:Term/
@deprecated"/>

<sch:let name="values"
value=" for $each in $depValues return if(index-of(tokenize(./@ism:disseminationControls,'
'), $each)>0) then $each else null"/>
<sch:let name="reportErr"
```

```
value=" for $each in $values return if($curDate >=
translate(subsequence($depDates,index-of($depValues,$each),1),'-','))
then concat('[' ,string($each),' ] is not authorized for use after ',
string(subsequence($depDates,index-of($depValues,$each),1))) else null " />

<sch:assert test="count($reportErr)=0" flag="error">
[ISM-ID-00180][Error] For attribute disseminationControls, value(s) <sch:value-of
select="$reportErr"/>
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00181

FileName:../Rules/atomicEnergyMarkings/ISM_ID_00181.sch

Rule Description:

[ISM-ID-00181][Error] If ISM-CAPCO-RESOURCE and element's classification does not have a value of "U" then attribute atomicEnergyMarkings must not contain the name token [UCNI]. Human Readable: UCNI may only be used on Unclassified portions.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If it is a CAPCO resource then we check the classification attribute of the current element. If it has a value of [U] we return true since this rule only applies to classified elements. If it is not [U] then we check that the attribute atomicEnergyMarkings does not have a value of [UCNI]

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00181">

<sch:rule context="//*[@ism:atomicEnergyMarkings]">
<sch:assert id="ism00181"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(./@ism:classification='U') then
true() else not(index-of(tokenize(./@ism:atomicEnergyMarkings, ' '), 'UCNI')>0) "
flag="error">
[ISM-ID-00181][Error] UCNI may only be used on Unclassified portions.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00182

FileName:../Rules/atomicEnergyMarkings/ISM_ID_00182.sch

Rule Description:

[ISM-ID-00182][Error] If ISM-CAPCO-RESOURCE and element's classification does not have a value of "U" then attribute atomicEnergyMarkings must not contain the name token [DCNI]. Human Readable: DCNI may only be used on Unclassified portions.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If it is a CAPCO resource then we check the classification attribute of the current element. If it has a value of [U] we return true since this rule only applies to classified elements. If it is not [U] then we check that the attribute atomicEnergyMarkings does not have a value of [DCNI]

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00182">

<sch:rule context="//*[@ism:atomicEnergyMarkings]">
<sch:assert id="ism00182"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(./@ism:classification='U') then
true() else not(index-of(tokenize(./@ism:atomicEnergyMarkings, ' '), 'DCNI')>0) "
flag="error">
[ISM-ID-00182][Error] DCNI may only be used on Unclassified portions.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00183

FileName:../Rules/atomicEnergyMarkings/ISM_ID_00183.sch

Rule Description:

[ISM-ID-00183][Error] If ISM-CAPCO-RESOURCE and attribute atomicEnergyMarkings contains the name token [RD-SG], then it must also contain the name token [RD].

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If it is a CAPCO resource then we check that if we have an element having attribute atomicEnergyMarkings with a value of [RD-SG] then we also have a value of [RD].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00183">

<sch:rule context="//*[@ism:atomicEnergyMarkings]">
<sch:let name="atc_tok" value="tokenize(./@ism:atomicEnergyMarkings, ' ')" />
<sch:assert id="ism00183"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(index-of($atc_tok, 'RD-SG')>0)
then index-of($atc_tok, 'RD')>0 else true() "
flag="error">
[ISM-ID-00183][Error] If ISM-CAPCO-RESOURCE and attribute atomicEnergyMarkings contains
the name token [RD-SG],
then it must also contain the name token [RD].
</sch:assert>
</sch:rule>
</sch:pattern>
```


Rule: ISM-ID-00184

FileName:../Rules/atomicEnergyMarkings/ISM_ID_00184.sch

Rule Description:

[ISM-ID-00184][Error] If ISM-CAPCO-RESOURCE and attribute atomicEnergyMarkings contains the name token [FRD-SG], then it must also contain the name token [FRD].

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If it is a CAPCO resource then we check that if we have an element having attribute atomicEnergyMarkings with a value of [FRD-SG] then we also have a value of [FRD].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00184">

<sch:rule context="//*[@ism:atomicEnergyMarkings]">
<sch:let name="atc_tok" value="tokenize(./@ism:atomicEnergyMarkings, ' ')" />
<sch:assert id="ism00184"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(index-of($atc_tok, 'FRD-SG')>0)
then index-of($atc_tok, 'FRD')>0 else true() "
flag="error">
[ISM-ID-00184][Error] If ISM-CAPCO-RESOURCE and attribute atomicEnergyMarkings contains
the name token [FRD-SG],
then it must also contain the name token [FRD].
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00185

FileName:./Rules/atomicEnergyMarkings/ISM_ID_00185.sch

Rule Description:

[ISM-ID-00185][Error] If ISM-CAPCO-RESOURCE and attribute atomicEnergyMarkings contains the name token [RD-CNWDI], then it must also contain the name token [RD].

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If it is a CAPCO resource then we check that if we have an element having attribute atomicEnergyMarkings with a value of [RD-CNWDI] then we also have a value of [RD].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00185">

<sch:rule context="//*[@ism:atomicEnergyMarkings]">
<sch:let name="atc_tok" value="tokenize(./@ism:atomicEnergyMarkings,' ')" />
<sch:assert id="ism00185"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(index-of($atc_tok, 'RD-
CNWDI')>0) then index-of($atc_tok, 'RD')>0 else true() "
flag="error">
[ISM-ID-00185][Error] If ISM-CAPCO-RESOURCE and attribute atomicEnergyMarkings contains
the name token [RD-CNWDI],
then it must also contain the name token [RD].
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00186

FileName:./Rules/SCIcontrols/ISM_ID_00186.sch

Rule Description:

[ISM-ID-00186][Error] If ISM-CAPCO-RESOURCE and attribute SCIcontrols contains the name token [SI-G-XXXX], then it must also contain the name token [SI-G].

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If the current element has attribute SCIcontrols specified with a value containing [SI-G-XXXX], where X is represented by the regular expression character class [A-Z], then we make sure that attribute SCIcontrols also contains the value [SI-G].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00186">

<sch:rule context="//*[@ism:SCIcontrols]">
<sch:assert id="ism00186"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if( count( for $each in tokenize(./
@ism:SCIcontrols,' ') return if(matches($each,'^SI-G-[A-Z][A-Z][A-Z][A-Z]')) then 1 else
null )>0 ) then index-of(tokenize(./@ism:SCIcontrols,' '), 'SI-G')>0 else true() "
flag="error">
[ISM-ID-00186][Error] If ISM-CAPCO-RESOURCE and attribute SCIcontrols contains the name
token [SI-G-XXXX],
then it must also contain the name token [SI-G].
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00187

FileName:./Rules/SCIcontrols/ISM_ID_00187.sch

Rule Description:

[ISM-ID-00187][Error] If ISM-CAPCO-RESOURCE and attribute SCIcontrols contains the name token [SI-G], then it must also contain the name token [SI].

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If the current element has attribute SCIcontrols specified with a value containing [SI-G], then we make sure that attribute SCIcontrols also contains the value [SI].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00187">

<sch:rule context="//*[@ism:SCIcontrols]">
<sch:assert id="ism00187"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if( count( for $each in tokenize(./
@ism:SCIcontrols,' ') return if(matches($each,'^SI-G')) then 1 else null )>0 ) then
index-of(tokenize(./@ism:SCIcontrols,' '), 'SI')>0 else true() "
flag="error">
[ISM-ID-00187][Error] If ISM-CAPCO-RESOURCE and attribute SCIcontrols contains the name
token [SI-G],
then it must also contain the name token [SI].
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00188

FileName:./Rules/generalConstraints/ISM_ID_00188.sch

Rule Description:

[ISM-ID-00188][Warning] Attribute FGIsorceOpen must not contain values that have passed their deprecation date.

Code Description:

Traverse the CVE file pulling out deprecated values and their dates. Using the current date, determine if the values are being used but it is still prior to the deprecated date

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00188">

  <!--
Any changes made to this code may also be required for the
following rule files:
- ISM_ID_00166.sch
- ISM_ID_00179.sch
- ISM_ID_00188.sch
- ISM_ID_00190.sch
- ISM_ID_00192.sch
- ISM_ID_00194.sch
- ISM_ID_00196.sch
- ISM_ID_00198.sch
- ISM_ID_00200.sch
- ISM_ID_00202.sch
- ISM_ID_00204.sch
- ISM_ID_00206.sch
- ISM_ID_00208.sch
- ISM_ID_00210.sch
-->
  <sch:rule context="//*[@ism:FGIsorceOpen]">
    <sch:let name="curDate" value="translate($ISM_RESOURCE_ELEMENT/@ism:createDate,'-','')"/>
    <sch:let name="depValues"
value="document('._CVE/CVEnumISMFGIOpen.xml')//cve:CVE/cve:Enumeration/cve:Term[./
@deprecated]/cve:Value"/>
    <sch:let name="depDates"
value="document('._CVE/CVEnumISMFGIOpen.xml')//cve:CVE/cve:Enumeration/cve:Term/
@deprecated"/>

    <sch:let name="values"
value=" for $each in $depValues return if(index-of(tokenize(./@ism:FGIsorceOpen,' '),
$each)>0) then $each else null"/>
    <sch:let name="reportWarn"
```

```
value=" for $each in $values return if($curDate <
translate(subsequence($depDates,index-of($depValues,$each),1),'-',')) then
concat('[' ,string($each),' ] has been deprecated and is not authorized for used after ' ,
string(subsequence($depDates,index-of($depValues,$each),1))) else null "/>

<sch:assert test="count($reportWarn)=0" flag="warning">
[ISM-ID-00188][Warning] For attribute FGIsorceOpen, value(s) <sch:value-of
select="$reportWarn"/>
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00189

FileName:./Rules/generalConstraints/ISM_ID_00189.sch

Rule Description:

[ISM-ID-00189][Error] Attribute FGIsSourceOpen must not contain values that have passed their deprecation date.

Code Description:

Traverse the CVE file pulling out deprecated values and their dates. Using the current date determine if the values are being used passed the deprecated date

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00189">

  <!--
Any changes made to this code may also be required for the
following rule files:
- ISM_ID_00170.sch
- ISM_ID_00180.sch
- ISM_ID_00189.sch
- ISM_ID_00191.sch
- ISM_ID_00193.sch
- ISM_ID_00195.sch
- ISM_ID_00197.sch
- ISM_ID_00199.sch
- ISM_ID_00201.sch
- ISM_ID_00203.sch
- ISM_ID_00205.sch
- ISM_ID_00207.sch
- ISM_ID_00209.sch
- ISM_ID_00211.sch
-->
  <sch:rule context="//*[@ism:FGIsSourceOpen]">
    <sch:let name="curDate" value="translate($ISM_RESOURCE_ELEMENT/@ism:createDate,'-','')"/>
    <sch:let name="depValues"
value="document('._CVE/CVEnumISMFGIOpen.xml')//cve:CVE/cve:Enumeration/cve:Term[./
@deprecated]/cve:Value"/>
    <sch:let name="depDates"
value="document('._CVE/CVEnumISMFGIOpen.xml')//cve:CVE/cve:Enumeration/cve:Term/
@deprecated"/>

    <sch:let name="values"
value=" for $each in $depValues return if(index-of(tokenize(./@ism:FGIsSourceOpen,' '),
$each)>0) then $each else null"/>
    <sch:let name="reportErr"
```

```
value=" for $each in $values return if($curDate >=
translate(subsequence($depDates,index-of($depValues,$each),1),'-', ''))
then concat('[' ,string($each),'] is not authorized for use after ',
string(subsequence($depDates,index-of($depValues,$each),1))) else null " />

<sch:assert test="count($reportErr)=0" flag="error">
[ISM-ID-00189][Error] For attribute FGIsorceOpen, value(s) <sch:value-of
select="$reportErr"/>
</sch:assert>
</sch:rule>
</sch:pattern>
```


Rule: ISM-ID-00190

FileName:./Rules/generalConstraints/ISM_ID_00190.sch

Rule Description:

[ISM-ID-00190][Warning] Attribute FGIsorceProtected must not contain values that have passed their deprecation date.

Code Description:

Traverse the CVE file pulling out deprecated values and their dates. Using the current date, determine if the values are being used but it is still prior to the deprecated date

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00190">

  <!--
Any changes made to this code may also be required for the
following rule files:
- ISM_ID_00166.sch
- ISM_ID_00179.sch
- ISM_ID_00188.sch
- ISM_ID_00190.sch
- ISM_ID_00192.sch
- ISM_ID_00194.sch
- ISM_ID_00196.sch
- ISM_ID_00198.sch
- ISM_ID_00200.sch
- ISM_ID_00202.sch
- ISM_ID_00204.sch
- ISM_ID_00206.sch
- ISM_ID_00208.sch
- ISM_ID_00210.sch
-->
  <sch:rule context="//*[@ism:FGIsorceProtected]">
    <sch:let name="curDate" value="translate($ISM_RESOURCE_ELEMENT/@ism:createDate,'-','')"/>
    <sch:let name="depValues"
value="document('._CVE/CVEnumISMFGIProtected.xml')//cve:CVE/cve:Enumeration/cve:Term[./
@deprecated]/cve:Value"/>
    <sch:let name="depDates"
value="document('._CVE/CVEnumISMFGIProtected.xml')//cve:CVE/cve:Enumeration/cve:Term/
@deprecated"/>

    <sch:let name="values"
value=" for $each in $depValues return if(index-of(tokenize(./@ism:FGIsorceProtected,'
'), $each)>0) then $each else null"/>
    <sch:let name="reportWarn"
```

```
value=" for $each in $values return if($curDate <
translate(subsequence($depDates,index-of($depValues,$each),1),'-',')) then
concat('[' ,string($each),' ] has been deprecated and is not authorized for used after ' ,
string(subsequence($depDates,index-of($depValues,$each),1))) else null "/>

<sch:assert test="count($reportWarn)=0" flag="warning">
[ISM-ID-00190][Warning] For attribute FGIsourceProtected, value(s) <sch:value-of
select="$reportWarn"/>
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00191

FileName:./Rules/generalConstraints/ISM_ID_00191.sch

Rule Description:

[ISM-ID-00191][Error] Attribute FGIsSourceProtected must not contain values that have passed their deprecation date.

Code Description:

Traverse the CVE file pulling out deprecated values and their dates. Using the current date determine if the values are being used passed the deprecated date

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00191">

  <!--
Any changes made to this code may also be required for the
following rule files:
- ISM_ID_00170.sch
- ISM_ID_00180.sch
- ISM_ID_00189.sch
- ISM_ID_00199.sch
- ISM_ID_00193.sch
- ISM_ID_00195.sch
- ISM_ID_00197.sch
- ISM_ID_00199.sch
- ISM_ID_00201.sch
- ISM_ID_00203.sch
- ISM_ID_00205.sch
- ISM_ID_00207.sch
- ISM_ID_00209.sch
- ISM_ID_00211.sch
-->
  <sch:rule context="//*[@ism:FGIsSourceProtected]">
    <sch:let name="curDate" value="translate($ISM_RESOURCE_ELEMENT/@ism:createDate,'-','')"/>
    <sch:let name="depValues"
value="document('._CVE/CVEnumISMFGIProtected.xml')//cve:CVE/cve:Enumeration/cve:Term[./
@deprecated]/cve:Value"/>
    <sch:let name="depDates"
value="document('._CVE/CVEnumISMFGIProtected.xml')//cve:CVE/cve:Enumeration/cve:Term/
@deprecated"/>

    <sch:let name="values"
value=" for $each in $depValues return if(index-of(tokenize(./@ism:FGIsSourceProtected,'
'), $each)>0) then $each else null"/>
    <sch:let name="reportErr"
```

```
value=" for $each in $values return if($curDate >=
translate(subsequence($depDates,index-of($depValues,$each),1),'-','))
then concat('[' ,string($each),' ] is not authorized for use after ',
string(subsequence($depDates,index-of($depValues,$each),1))) else null "/>

<sch:assert test="count($reportErr)=0" flag="error">
[ISM-ID-00191][Error] For attribute FGIsSourceProtected, value(s) <sch:value-of
select="$reportErr"/>
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00192

FileName:./Rules/generalConstraints/ISM_ID_00192.sch

Rule Description:

[ISM-ID-00192][Warning] Attribute nonICmarkings must not contain values that have passed their deprecation date.

Code Description:

Traverse the CVE file pulling out deprecated values and their dates. Using the current date, determine if the values are being used but it is still prior to the deprecated date

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00192">

  <!--
Any changes made to this code may also be required for the
following rule files:
- ISM_ID_00166.sch
- ISM_ID_00179.sch
- ISM_ID_00188.sch
- ISM_ID_00190.sch
- ISM_ID_00192.sch
- ISM_ID_00194.sch
- ISM_ID_00196.sch
- ISM_ID_00198.sch
- ISM_ID_00200.sch
- ISM_ID_00202.sch
- ISM_ID_00204.sch
- ISM_ID_00206.sch
- ISM_ID_00208.sch
- ISM_ID_00210.sch
-->
  <sch:rule context="//*[@ism:nonICmarkings]">
    <sch:let name="curDate" value="translate($ISM_RESOURCE_ELEMENT/@ism:createDate,'-','')"/>
    <sch:let name="depValues"
value="document('._CVE/CVEnumISMNonIC.xml')//cve:CVE/cve:Enumeration/cve:Term[./
@deprecated]/cve:Value"/>
    <sch:let name="depDates"
value="document('._CVE/CVEnumISMNonIC.xml')//cve:CVE/cve:Enumeration/cve:Term/
@deprecated"/>

    <sch:let name="values"
value=" for $each in $depValues return if(index-of(tokenize(./@ism:nonICmarkings,' '),
$each)>0) then $each else null"/>
    <sch:let name="reportWarn"
```

```
value=" for $each in $values return if($curDate <
translate(subsequence($depDates,index-of($depValues,$each),1),'-',')) then
concat('[',string($each),'] has been deprecated and is not authorized for used after ',
string(subsequence($depDates,index-of($depValues,$each),1))) else null "/>

<sch:assert test="count($reportWarn)=0" flag="warning">
[ISM-ID-00192][Warning] For attribute nonICmarkings, value(s) <sch:value-of
select="$reportWarn"/>
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00193

FileName:./Rules/generalConstraints/ISM_ID_00193.sch

Rule Description:

[ISM-ID-00193][Error] Attribute nonICmarkings must not contain values that have passed their deprecation date.

Code Description:

Traverse the CVE file pulling out deprecated values and their dates. Using the current date determine if the values are being used passed the deprecated date

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00193">

  <!--
Any changes made to this code may also be required for the
following rule files:
- ISM_ID_00170.sch
- ISM_ID_00180.sch
- ISM_ID_00189.sch
- ISM_ID_00199.sch
- ISM_ID_00193.sch
- ISM_ID_00195.sch
- ISM_ID_00197.sch
- ISM_ID_00199.sch
- ISM_ID_00201.sch
- ISM_ID_00203.sch
- ISM_ID_00205.sch
- ISM_ID_00207.sch
- ISM_ID_00209.sch
- ISM_ID_00211.sch
-->
  <sch:rule context="//*[@ism:nonICmarkings]">
    <sch:let name="curDate" value="translate($ISM_RESOURCE_ELEMENT/@ism:createDate,'-','')"/>
    <sch:let name="depValues"
value="document('._CVE/CVEnumISMNonIC.xml')//cve:CVE/cve:Enumeration/cve:Term[./
@deprecated]/cve:Value"/>
    <sch:let name="depDates"
value="document('._CVE/CVEnumISMNonIC.xml')//cve:CVE/cve:Enumeration/cve:Term/
@deprecated"/>

    <sch:let name="values"
value=" for $each in $depValues return if(index-of(tokenize(./@ism:nonICmarkings,' '),
$each)>0) then $each else null"/>
    <sch:let name="reportErr"
```

```
value=" for $each in $values return if($curDate >=
translate(subsequence($depDates,index-of($depValues,$each),1),'-','))
then concat('[' ,string($each),' ] is not authorized for use after ',
string(subsequence($depDates,index-of($depValues,$each),1))) else null " />

<sch:assert test="count($reportErr)=0" flag="error">
[ISM-ID-00193][Error] For attribute nonICmarkings, value(s) <sch:value-of
select="$reportErr"/>
</sch:assert>
</sch:rule>
</sch:pattern>
```


Rule: ISM-ID-00194

FileName:./Rules/generalConstraints/ISM_ID_00194.sch

Rule Description:

[ISM-ID-00194][Warning] Attribute notice must not contain values that have passed their deprecation date.

Code Description:

Traverse the CVE file pulling out deprecated values and their dates. Using the current date, determine if the values are being used but it is still prior to the deprecated date

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00194">

  <!--
Any changes made to this code may also be required for the
following rule files:
- ISM_ID_00166.sch
- ISM_ID_00179.sch
- ISM_ID_00188.sch
- ISM_ID_00190.sch
- ISM_ID_00192.sch
- ISM_ID_00194.sch
- ISM_ID_00196.sch
- ISM_ID_00198.sch
- ISM_ID_00200.sch
- ISM_ID_00202.sch
- ISM_ID_00204.sch
- ISM_ID_00206.sch
- ISM_ID_00208.sch
- ISM_ID_00210.sch
-->
  <sch:rule context="//*[@ism:notice]">
    <sch:let name="curDate" value="translate($ISM_RESOURCE_ELEMENT/@ism:createDate,'-','')"/>
    <sch:let name="depValues"
value="document('._CVE/CVEnumISMNotice.xml')//cve:CVE/cve:Enumeration/cve:Term[./
@deprecated]/cve:Value"/>
    <sch:let name="depDates"
value="document('._CVE/CVEnumISMNotice.xml')//cve:CVE/cve:Enumeration/cve:Term/
@deprecated"/>

    <sch:let name="values"
value=" for $each in $depValues return if(index-of(tokenize(./@ism:notice,' '),
$each)>0) then $each else null"/>
    <sch:let name="reportWarn"
```

```
value=" for $each in $values return if($curDate <
translate(subsequence($depDates,index-of($depValues,$each),1),'-','')) then
concat('[',string($each),'] has been deprecated and is not authorized for used after ',
string(subsequence($depDates,index-of($depValues,$each),1))) else null "/>

<sch:assert test="count($reportWarn)=0" flag="warning">
[ISM-ID-00194][Warning] For attribute notice, value(s) <sch:value-of select="$reportWarn"/
>
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00195

FileName:./Rules/generalConstraints/ISM_ID_00195.sch

Rule Description:

[ISM-ID-00195][Error] Attribute notice must not contain values that have passed their deprecation date.

Code Description:

Traverse the CVE file pulling out deprecated values and their dates. Using the current date determine if the values are being used passed the deprecated date

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00195">

  <!--
Any changes made to this code may also be required for the
following rule files:
- ISM_ID_00170.sch
- ISM_ID_00180.sch
- ISM_ID_00189.sch
- ISM_ID_00199.sch
- ISM_ID_00193.sch
- ISM_ID_00195.sch
- ISM_ID_00197.sch
- ISM_ID_00199.sch
- ISM_ID_00201.sch
- ISM_ID_00203.sch
- ISM_ID_00205.sch
- ISM_ID_00207.sch
- ISM_ID_00209.sch
- ISM_ID_00211.sch
-->
  <sch:rule context="//*[@ism:notice]">
    <sch:let name="curDate" value="translate($ISM_RESOURCE_ELEMENT/@ism:createDate,'-','')"/>
    <sch:let name="depValues"
value="document('._CVE/CVEnumISMNotice.xml')//cve:CVE/cve:Enumeration/cve:Term[./
@deprecated]/cve:Value"/>
    <sch:let name="depDates"
value="document('._CVE/CVEnumISMNotice.xml')//cve:CVE/cve:Enumeration/cve:Term/
@deprecated"/>

    <sch:let name="values"
value=" for $each in $depValues return if(index-of(tokenize(./@ism:notice,' '),
$each)>0) then $each else null"/>
    <sch:let name="reportErr"
```

```
value=" for $each in $values return if($curDate >=
translate(subsequence($depDates,index-of($depValues,$each),1),'-','))
then concat('[' ,string($each),' ] is not authorized for use after ',
string(subsequence($depDates,index-of($depValues,$each),1))) else null "/>

<sch:assert test="count($reportErr)=0" flag="error">
[ISM-ID-00195][Error] For attribute notice, value(s) <sch:value-of select="$reportErr"/>
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00196

FileName:./Rules/generalConstraints/ISM_ID_00196.sch

Rule Description:

[ISM-ID-00196][Warning] Attribute ownerProducer must not contain values that have passed their deprecation date.

Code Description:

Traverse the CVE file pulling out deprecated values and their dates. Using the current date, determine if the values are being used but it is still prior to the deprecated date

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00196">

  <!--
Any changes made to this code may also be required for the
following rule files:
- ISM_ID_00166.sch
- ISM_ID_00179.sch
- ISM_ID_00188.sch
- ISM_ID_00190.sch
- ISM_ID_00192.sch
- ISM_ID_00194.sch
- ISM_ID_00196.sch
- ISM_ID_00198.sch
- ISM_ID_00200.sch
- ISM_ID_00202.sch
- ISM_ID_00204.sch
- ISM_ID_00206.sch
- ISM_ID_00208.sch
- ISM_ID_00210.sch
-->
  <sch:rule context="//*[@ism:ownerProducer]">
    <sch:let name="curDate" value="translate($ISM_RESOURCE_ELEMENT/@ism:createDate,'-','')"/>
    <sch:let name="depValues"
value="document('._CVE/CVEnumISMOwnerProducer.xml')//cve:CVE/cve:Enumeration/cve:Term[./
@deprecated]/cve:Value"/>
    <sch:let name="depDates"
value="document('._CVE/CVEnumISMOwnerProducer.xml')//cve:CVE/cve:Enumeration/cve:Term/
@deprecated"/>

    <sch:let name="values"
value=" for $each in $depValues return if(index-of(tokenize(./@ism:ownerProducer,' '),
$each)>0) then $each else null"/>
    <sch:let name="reportWarn"
```

```
value=" for $each in $values return if($curDate <
translate(subsequence($depDates,index-of($depValues,$each),1),'-','')) then
concat('[' ,string($each),' ] has been deprecated and is not authorized for used after ' ,
string(subsequence($depDates,index-of($depValues,$each),1))) else null " />

<sch:assert test="count($reportWarn)=0" flag="warning">
[ISM-ID-00196][Warning] For attribute ownerProducer, value(s) <sch:value-of
select="$reportWarn" />
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00197

FileName:./Rules/generalConstraints/ISM_ID_00197.sch

Rule Description:

[ISM-ID-00197][Error] Attribute ownerProducer must not contain values that have passed their deprecation date.

Code Description:

Traverse the CVE file pulling out deprecated values and their dates. Using the current date determine if the values are being used passed the deprecated date

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00197">

  <!--
Any changes made to this code may also be required for the
following rule files:
- ISM_ID_00170.sch
- ISM_ID_00180.sch
- ISM_ID_00189.sch
- ISM_ID_00199.sch
- ISM_ID_00193.sch
- ISM_ID_00195.sch
- ISM_ID_00197.sch
- ISM_ID_00199.sch
- ISM_ID_00201.sch
- ISM_ID_00203.sch
- ISM_ID_00205.sch
- ISM_ID_00207.sch
- ISM_ID_00209.sch
- ISM_ID_00211.sch
-->
  <sch:rule context="//*[@ism:ownerProducer]">
    <sch:let name="curDate" value="translate($ISM_RESOURCE_ELEMENT/@ism:createDate,'-','')"/>
    <sch:let name="depValues"
value="document('._CVE/CVEnumISMOwnerProducer.xml')//cve:CVE/cve:Enumeration/cve:Term[./
@deprecated]/cve:Value"/>
    <sch:let name="depDates"
value="document('._CVE/CVEnumISMOwnerProducer.xml')//cve:CVE/cve:Enumeration/cve:Term/
@deprecated"/>

    <sch:let name="values"
value=" for $each in $depValues return if(index-of(tokenize(./@ism:ownerProducer,' '),
$each)>0) then $each else null"/>
    <sch:let name="reportErr"
```

```
value=" for $each in $values return if($curDate >=
translate(subsequence($depDates,index-of($depValues,$each),1),'-','))
then concat('[' ,string($each),' ] is not authorized for use after ',
string(subsequence($depDates,index-of($depValues,$each),1))) else null "/>

<sch:assert test="count($reportErr)=0" flag="error">
[ISM-ID-00197][Error] For attribute ownerProducer, value(s) <sch:value-of
select="$reportErr"/>
</sch:assert>
</sch:rule>
</sch:pattern>
```


Rule: ISM-ID-00198

FileName:./Rules/generalConstraints/ISM_ID_00198.sch

Rule Description:

[ISM-ID-00198][Warning] Attribute releasableTo must not contain values that have passed their deprecation date.

Code Description:

Traverse the CVE file pulling out deprecated values and their dates. Using the current date, determine if the values are being used but it is still prior to the deprecated date

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00198">

  <!--
Any changes made to this code may also be required for the
following rule files:
- ISM_ID_00166.sch
- ISM_ID_00179.sch
- ISM_ID_00188.sch
- ISM_ID_00190.sch
- ISM_ID_00192.sch
- ISM_ID_00194.sch
- ISM_ID_00196.sch
- ISM_ID_00198.sch
- ISM_ID_00200.sch
- ISM_ID_00202.sch
- ISM_ID_00204.sch
- ISM_ID_00206.sch
- ISM_ID_00208.sch
- ISM_ID_00210.sch
-->
  <sch:rule context="//*[@ism:releasableTo]">
    <sch:let name="curDate" value="translate($ISM_RESOURCE_ELEMENT/@ism:createDate,'-','')"/>
    <sch:let name="depValues"
value="document('._CVE/CVEnumISMRelTo.xml')//cve:CVE/cve:Enumeration/cve:Term[./
@deprecated]/cve:Value"/>
    <sch:let name="depDates"
value="document('._CVE/CVEnumISMRelTo.xml')//cve:CVE/cve:Enumeration/cve:Term/
@deprecated"/>

    <sch:let name="values"
value=" for $each in $depValues return if(index-of(tokenize(./@ism:releasableTo,' '),
$each)>0) then $each else null"/>
    <sch:let name="reportWarn"
```

```
value=" for $each in $values return if($curDate <
translate(subsequence($depDates,index-of($depValues,$each),1),'-','')) then
concat('[',string($each),'] has been deprecated and is not authorized for used after ',
string(subsequence($depDates,index-of($depValues,$each),1))) else null "/>

<sch:assert test="count($reportWarn)=0" flag="warning">
[ISM-ID-00198][Warning] For attribute releasableTo, value(s) <sch:value-of
select="$reportWarn"/>
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00199

FileName:./Rules/generalConstraints/ISM_ID_00199.sch

Rule Description:

[ISM-ID-00199][Error] Attribute releasableTo must not contain values that have passed their deprecation date.

Code Description:

Traverse the CVE file pulling out deprecated values and their dates. Using the current date determine if the values are being used passed the deprecated date

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00199">

  <!--
Any changes made to this code may also be required for the
following rule files:
- ISM_ID_00170.sch
- ISM_ID_00180.sch
- ISM_ID_00189.sch
- ISM_ID_00199.sch
- ISM_ID_00193.sch
- ISM_ID_00195.sch
- ISM_ID_00197.sch
- ISM_ID_00199.sch
- ISM_ID_00201.sch
- ISM_ID_00203.sch
- ISM_ID_00205.sch
- ISM_ID_00207.sch
- ISM_ID_00209.sch
- ISM_ID_00211.sch
-->
  <sch:rule context="//*[@ism:releasableTo]">
    <sch:let name="curDate" value="translate($ISM_RESOURCE_ELEMENT/@ism:createDate,'-','')"/>
    <sch:let name="depValues"
value="document('._CVE/CVEnumISMRelTo.xml')//cve:CVE/cve:Enumeration/cve:Term[./
@deprecated]/cve:Value"/>
    <sch:let name="depDates"
value="document('._CVE/CVEnumISMRelTo.xml')//cve:CVE/cve:Enumeration/cve:Term/
@deprecated"/>

    <sch:let name="values"
value=" for $each in $depValues return if(index-of(tokenize(./@ism:releasableTo,' '),
$each)>0) then $each else null"/>
    <sch:let name="reportErr"
```

```
value=" for $each in $values return if($curDate >=
translate(subsequence($depDates,index-of($depValues,$each),1),'-','))
then concat('[' ,string($each),' ] is not authorized for use after ',
string(subsequence($depDates,index-of($depValues,$each),1))) else null " />

<sch:assert test="count($reportErr)=0" flag="error">
[ISM-ID-00199][Error] For attribute releasableTo, value(s) <sch:value-of
select="$reportErr"/>
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00200

FileName:./Rules/generalConstraints/ISM_ID_00200.sch

Rule Description:

[ISM-ID-00200][Warning] Attribute displayOnlyTo must not contain values that have passed their deprecation date.

Code Description:

Traverse the CVE file pulling out deprecated values and their dates. Using the current date, determine if the values are being used but it is still prior to the deprecated date

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00200">

  <!--
Any changes made to this code may also be required for the
following rule files:
- ISM_ID_00166.sch
- ISM_ID_00179.sch
- ISM_ID_00188.sch
- ISM_ID_00190.sch
- ISM_ID_00192.sch
- ISM_ID_00194.sch
- ISM_ID_00196.sch
- ISM_ID_00198.sch
- ISM_ID_00200.sch
- ISM_ID_00202.sch
- ISM_ID_00204.sch
- ISM_ID_00206.sch
- ISM_ID_00208.sch
- ISM_ID_00210.sch
-->
  <sch:rule context="//*[@ism:displayOnlyTo]">
    <sch:let name="curDate" value="translate($ISM_RESOURCE_ELEMENT/@ism:createDate,'-','')"/>
    <sch:let name="depValues"
value="document('._CVE/CVEnumISMRelTo.xml')//cve:CVE/cve:Enumeration/cve:Term[./
@deprecated]/cve:Value"/>
    <sch:let name="depDates"
value="document('._CVE/CVEnumISMRelTo.xml')//cve:CVE/cve:Enumeration/cve:Term/
@deprecated"/>

    <sch:let name="values"
value=" for $each in $depValues return if(index-of(tokenize(./@ism:displayOnlyTo,' '),
$each)>0) then $each else null"/>
    <sch:let name="reportWarn"
```

```
value=" for $each in $values return if($curDate <
translate(subsequence($depDates,index-of($depValues,$each),1),'-',')) then
concat('[',string($each),'] has been deprecated and is not authorized for used after ',
string(subsequence($depDates,index-of($depValues,$each),1))) else null "/>

<sch:assert test="count($reportWarn)=0" flag="warning">
[ISM-ID-00200][Warning] For attribute displayOnlyTo, value(s) <sch:value-of
select="$reportWarn"/>
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00201

FileName:./Rules/generalConstraints/ISM_ID_00201.sch

Rule Description:

[ISM-ID-00201][Error] Attribute displayOnlyTo must not contain values that have passed their deprecation date.

Code Description:

Traverse the CVE file pulling out deprecated values and their dates. Using the current date determine if the values are being used passed the deprecated date

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00201">

<!--
Any changes made to this code may also be required for the
following rule files:
- ISM_ID_00170.sch
- ISM_ID_00180.sch
- ISM_ID_00189.sch
- ISM_ID_00199.sch
- ISM_ID_00193.sch
- ISM_ID_00195.sch
- ISM_ID_00197.sch
- ISM_ID_00199.sch
- ISM_ID_00201.sch
- ISM_ID_00203.sch
- ISM_ID_00205.sch
- ISM_ID_00207.sch
- ISM_ID_00209.sch
- ISM_ID_00211.sch
-->
<sch:rule context="//*[@ism:displayOnlyTo]">
<sch:let name="curDate" value="translate($ISM_RESOURCE_ELEMENT/@ism:createDate,'-','')"/>
<sch:let name="depValues"
value="document('._CVE/CVEnumISMRelTo.xml')//cve:CVE/cve:Enumeration/cve:Term[./
@deprecated]/cve:Value"/>
<sch:let name="depDates"
value="document('._CVE/CVEnumISMRelTo.xml')//cve:CVE/cve:Enumeration/cve:Term/
@deprecated"/>

<sch:let name="values"
value=" for $each in $depValues return if(index-of(tokenize(./@ism:displayOnlyTo,' '),
$each)>0) then $each else null"/>
<sch:let name="reportErr"
```

```
value=" for $each in $values return if($curDate >=
translate(subsequence($depDates,index-of($depValues,$each),1),'-','))
then concat('[' ,string($each),' ] is not authorized for use after ',
string(subsequence($depDates,index-of($depValues,$each),1))) else null " />

<sch:assert test="count($reportErr)=0" flag="error">
[ISM-ID-00201][Error] For attribute displayOnlyTo, value(s) <sch:value-of
select="$reportErr"/>
</sch:assert>
</sch:rule>
</sch:pattern>
```


Rule: ISM-ID-00202

FileName:./Rules/generalConstraints/ISM_ID_00202.sch

Rule Description:

[ISM-ID-00202][Warning] Attribute SARIdentifier must not contain values that have passed their deprecation date.

Code Description:

Traverse the CVE file pulling out deprecated values and their dates. Using the current date, determine if the values are being used but it is still prior to the deprecated date

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00202">

  <!--
Any changes made to this code may also be required for the
following rule files:
- ISM_ID_00166.sch
- ISM_ID_00179.sch
- ISM_ID_00188.sch
- ISM_ID_00190.sch
- ISM_ID_00192.sch
- ISM_ID_00194.sch
- ISM_ID_00196.sch
- ISM_ID_00198.sch
- ISM_ID_00200.sch
- ISM_ID_00202.sch
- ISM_ID_00204.sch
- ISM_ID_00206.sch
- ISM_ID_00208.sch
- ISM_ID_00210.sch
-->
  <sch:rule context="//*[@ism:SARIdentifier]">
    <sch:let name="curDate" value="translate($ISM_RESOURCE_ELEMENT/@ism:createDate,'-','')"/>
    <sch:let name="depValues"
value="document('._CVE/CVEnumISMSAR.xml')//cve:CVE/cve:Enumeration/cve:Term[./
@deprecated]/cve:Value"/>
    <sch:let name="depDates"
value="document('._CVE/CVEnumISMSAR.xml')//cve:CVE/cve:Enumeration/cve:Term/@deprecated"/
>

    <sch:let name="values"
value=" for $each in $depValues return if(index-of(tokenize(./@ism:SARIdentifier,' '),
$each)>0) then $each else null"/>
    <sch:let name="reportWarn"
```

```
value=" for $each in $values return if($curDate <
translate(subsequence($depDates,index-of($depValues,$each),1),'-','')) then
concat('[',string($each),'] has been deprecated and is not authorized for used after ',
string(subsequence($depDates,index-of($depValues,$each),1))) else null "/>

<sch:assert test="count($reportWarn)=0" flag="warning">
[ISM-ID-00202][Warning] For attribute SARIdentifier, value(s) <sch:value-of
select="$reportWarn"/>
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00203

FileName:./Rules/generalConstraints/ISM_ID_00203.sch

Rule Description:

[ISM-ID-00203][Error] Attribute SARIdentifier must not contain values that have passed their deprecation date.

Code Description:

Traverse the CVE file pulling out deprecated values and their dates. Using the current date determine if the values are being used passed the deprecated date

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00203">

  <!--
Any changes made to this code may also be required for the
following rule files:
- ISM_ID_00170.sch
- ISM_ID_00180.sch
- ISM_ID_00189.sch
- ISM_ID_00199.sch
- ISM_ID_00193.sch
- ISM_ID_00195.sch
- ISM_ID_00197.sch
- ISM_ID_00199.sch
- ISM_ID_00201.sch
- ISM_ID_00203.sch
- ISM_ID_00205.sch
- ISM_ID_00207.sch
- ISM_ID_00209.sch
- ISM_ID_00211.sch
-->
  <sch:rule context="//*[@ism:SARIdentifier]">
    <sch:let name="curDate" value="translate($ISM_RESOURCE_ELEMENT/@ism:createDate,'-','')"/>
    <sch:let name="depValues"
value="document('._CVE/CVEnumISMSAR.xml')//cve:CVE/cve:Enumeration/cve:Term[./
@deprecated]/cve:Value"/>
    <sch:let name="depDates"
value="document('._CVE/CVEnumISMSAR.xml')//cve:CVE/cve:Enumeration/cve:Term/@deprecated"/
>

    <sch:let name="values"
value=" for $each in $depValues return if(index-of(tokenize(./@ism:SARIdentifier,' '),
$each)>0) then $each else null"/>
    <sch:let name="reportErr"
```

```
value=" for $each in $values return if($curDate >=
translate(subsequence($depDates,index-of($depValues,$each),1),'-','))
then concat('[' ,string($each),' ] is not authorized for use after ',
string(subsequence($depDates,index-of($depValues,$each),1))) else null " />

<sch:assert test="count($reportErr)=0" flag="error">
[ISM-ID-00203][Error] For attribute SARIdentifier, value(s) <sch:value-of
select="$reportErr"/>
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00204

FileName:./Rules/generalConstraints/ISM_ID_00204.sch

Rule Description:

[ISM-ID-00204][Warning] Attribute SCIcontrols must not contain values that have passed their deprecation date.

Code Description:

Traverse the CVE file pulling out deprecated values and their dates. Using the current date, determine if the values are being used but it is still prior to the deprecated date

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00204">

  <!--
Any changes made to this code may also be required for the
following rule files:
- ISM_ID_00166.sch
- ISM_ID_00179.sch
- ISM_ID_00188.sch
- ISM_ID_00190.sch
- ISM_ID_00192.sch
- ISM_ID_00194.sch
- ISM_ID_00196.sch
- ISM_ID_00198.sch
- ISM_ID_00200.sch
- ISM_ID_00202.sch
- ISM_ID_00204.sch
- ISM_ID_00206.sch
- ISM_ID_00208.sch
- ISM_ID_00210.sch
-->
  <sch:rule context="//*[@ism:SCIcontrols]">
    <sch:let name="curDate" value="translate($ISM_RESOURCE_ELEMENT/@ism:createDate,'-','')"/>
    <sch:let name="depValues"
value="document('._CVE/CVEnumISMSCIcontrols.xml')//cve:CVE/cve:Enumeration/cve:Term[./
@deprecated]/cve:Value"/>
    <sch:let name="depDates"
value="document('._CVE/CVEnumISMSCIcontrols.xml')//cve:CVE/cve:Enumeration/cve:Term/
@deprecated"/>

    <sch:let name="values"
value=" for $each in $depValues return if(index-of(tokenize(./@ism:SCIcontrols,' '),
$each)>0) then $each else null"/>
    <sch:let name="reportWarn"
```

```
value=" for $each in $values return if($curDate <
translate(subsequence($depDates,index-of($depValues,$each),1),'-','')) then
concat('[',string($each),'] has been deprecated and is not authorized for used after ',
string(subsequence($depDates,index-of($depValues,$each),1))) else null "/>

<sch:assert test="count($reportWarn)=0" flag="warning">
[ISM-ID-00204][Warning] For attribute SCIcontrols, value(s) <sch:value-of
select="$reportWarn"/>
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00205

FileName:./Rules/generalConstraints/ISM_ID_00205.sch

Rule Description:

[ISM-ID-00205][Error] Attribute SCIcontrols must not contain values that have passed their deprecation date.

Code Description:

Traverse the CVE file pulling out deprecated values and their dates. Using the current date determine if the values are being used passed the deprecated date

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00205">

  <!--
Any changes made to this code may also be required for the
following rule files:
- ISM_ID_00170.sch
- ISM_ID_00180.sch
- ISM_ID_00189.sch
- ISM_ID_00199.sch
- ISM_ID_00193.sch
- ISM_ID_00195.sch
- ISM_ID_00197.sch
- ISM_ID_00199.sch
- ISM_ID_00201.sch
- ISM_ID_00203.sch
- ISM_ID_00205.sch
- ISM_ID_00207.sch
- ISM_ID_00209.sch
- ISM_ID_00211.sch
-->
  <sch:rule context="//*[@ism:SCIcontrols]">
    <sch:let name="curDate" value="translate($ISM_RESOURCE_ELEMENT/@ism:createDate,'-','')"/>
    <sch:let name="depValues"
value="document('._CVE/CVEnumISMSCIcontrols.xml')//cve:CVE/cve:Enumeration/cve:Term[./
@deprecated]/cve:Value"/>
    <sch:let name="depDates"
value="document('._CVE/CVEnumISMSCIcontrols.xml')//cve:CVE/cve:Enumeration/cve:Term/
@deprecated"/>

    <sch:let name="values"
value=" for $each in $depValues return if(index-of(tokenize(./@ism:SCIcontrols,' '),
$each)>0) then $each else null"/>
    <sch:let name="reportErr"
```

```
value=" for $each in $values return if($curDate >=
translate(subsequence($depDates,index-of($depValues,$each),1),'-','))
then concat('[' ,string($each),'] is not authorized for use after ',
string(subsequence($depDates,index-of($depValues,$each),1))) else null " />

<sch:assert test="count($reportErr)=0" flag="error">
[ISM-ID-00205][Error] For attribute SCIcontrols, value(s) <sch:value-of
select="$reportErr"/>
</sch:assert>
</sch:rule>
</sch:pattern>
```


Rule: ISM-ID-00206

FileName:./Rules/generalConstraints/ISM_ID_00206.sch

Rule Description:

[ISM-ID-00206][Warning] Attribute declassException must not contain values that have passed their deprecation date.

Code Description:

Traverse the CVE file pulling out deprecated values and their dates. Using the current date, determine if the values are being used but it is still prior to the deprecated date

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00206">

  <!--
Any changes made to this code may also be required for the
following rule files:
- ISM_ID_00166.sch
- ISM_ID_00179.sch
- ISM_ID_00188.sch
- ISM_ID_00190.sch
- ISM_ID_00192.sch
- ISM_ID_00194.sch
- ISM_ID_00196.sch
- ISM_ID_00198.sch
- ISM_ID_00200.sch
- ISM_ID_00202.sch
- ISM_ID_00204.sch
- ISM_ID_00206.sch
- ISM_ID_00208.sch
- ISM_ID_00210.sch
-->
  <sch:rule context="//*[@ism:declassException]">
    <sch:let name="curDate" value="translate($ISM_RESOURCE_ELEMENT/@ism:createDate,'-','')"/>
    <sch:let name="depValues"
value="document('._CVE/CVEnumISM25X.xml')//cve:CVE/cve:Enumeration/cve:Term[./
@deprecated]/cve:Value"/>
    <sch:let name="depDates"
value="document('._CVE/CVEnumISM25X.xml')//cve:CVE/cve:Enumeration/cve:Term/@deprecated"/
>

    <sch:let name="values"
value=" for $each in $depValues return if(index-of(tokenize(./@ism:declassException,' '),
$each)>0) then $each else null"/>
    <sch:let name="reportWarn"
```

```
value=" for $each in $values return if($curDate <
translate(subsequence($depDates,index-of($depValues,$each),1),'-','')) then
concat('[',string($each),'] has been deprecated and is not authorized for used after ',
string(subsequence($depDates,index-of($depValues,$each),1))) else null "/>

<sch:assert test="count($reportWarn)=0" flag="warning">
[ISM-ID-00206][Warning] For attribute declassException, value(s) <sch:value-of
select="$reportWarn"/>
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00207

FileName:./Rules/generalConstraints/ISM_ID_00207.sch

Rule Description:

[ISM-ID-00207][Error] Attribute declassException must not contain values that have passed their deprecation date.

Code Description:

Traverse the CVE file pulling out deprecated values and their dates. Using the current date determine if the values are being used passed the deprecated date

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00207">

<!--
Any changes made to this code may also be required for the
following rule files:
- ISM_ID_00170.sch
- ISM_ID_00180.sch
- ISM_ID_00189.sch
- ISM_ID_00199.sch
- ISM_ID_00193.sch
- ISM_ID_00195.sch
- ISM_ID_00197.sch
- ISM_ID_00199.sch
- ISM_ID_00201.sch
- ISM_ID_00203.sch
- ISM_ID_00205.sch
- ISM_ID_00207.sch
- ISM_ID_00209.sch
- ISM_ID_00211.sch
-->
<sch:rule context="//*[@ism:declassException]">
<sch:let name="curDate" value="translate($ISM_RESOURCE_ELEMENT/@ism:createDate,'-','')"/>
<sch:let name="depValues"
value="document('._CVE/CVEnumISM25X.xml')//cve:CVE/cve:Enumeration/cve:Term[./
@deprecated]/cve:Value"/>
<sch:let name="depDates"
value="document('._CVE/CVEnumISM25X.xml')//cve:CVE/cve:Enumeration/cve:Term/@deprecated"/
>

<sch:let name="values"
value=" for $each in $depValues return if(index-of(tokenize(./@ism:declassException,' '),
$each)>0) then $each else null"/>
<sch:let name="reportErr"
```

```
value=" for $each in $values return if($curDate >=
translate(subsequence($depDates,index-of($depValues,$each),1),'-','))
then concat('[' ,string($each),' ] is not authorized for use after ',
string(subsequence($depDates,index-of($depValues,$each),1))) else null " />

<sch:assert test="count($reportErr)=0" flag="error">
[ISM-ID-00207][Error] For attribute declassException, value(s) <sch:value-of
select="$reportErr"/>
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00208

FileName:./Rules/generalConstraints/ISM_ID_00208.sch

Rule Description:

[ISM-ID-00208][Warning] Attribute atomicEnergyMarkings must not contain values that have passed their deprecation date.

Code Description:

Traverse the CVE file pulling out deprecated values and their dates. Using the current date, determine if the values are being used but it is still prior to the deprecated date

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00208">

<!--
Any changes made to this code may also be required for the
following rule files:
- ISM_ID_00166.sch
- ISM_ID_00179.sch
- ISM_ID_00188.sch
- ISM_ID_00190.sch
- ISM_ID_00192.sch
- ISM_ID_00194.sch
- ISM_ID_00196.sch
- ISM_ID_00198.sch
- ISM_ID_00200.sch
- ISM_ID_00202.sch
- ISM_ID_00204.sch
- ISM_ID_00206.sch
- ISM_ID_00208.sch
- ISM_ID_00210.sch
-->
<sch:rule context="//*[@ism:atomicEnergyMarkings]">
<sch:let name="curDate" value="translate($ISM_RESOURCE_ELEMENT/@ism:createDate,'-','')"/>
<sch:let name="depValues"
value="document('._CVE/CVEnumISMAAtomicEnergyMarkings.xml')//cve:CVE/cve:Enumeration/
cve:Term[./@deprecated]/cve:Value"/>
<sch:let name="depDates"
value="document('._CVE/CVEnumISMAAtomicEnergyMarkings.xml')//cve:CVE/cve:Enumeration/
cve:Term/@deprecated"/>

<sch:let name="values"
value=" for $each in $depValues return if(index-of(tokenize(./@ism:atomicEnergyMarkings,'
'), $each)>0) then $each else null"/>
<sch:let name="reportWarn"
```

```
value=" for $each in $values return if($curDate <
translate(subsequence($depDates,index-of($depValues,$each),1),'-',')) then
concat('[',string($each),'] has been deprecated and is not authorized for used after ',
string(subsequence($depDates,index-of($depValues,$each),1))) else null "/>

<sch:assert test="count($reportWarn)=0" flag="warning">
[ISM-ID-00208][Warning] For attribute atomicEnergyMarkings, value(s) <sch:value-of
select="$reportWarn"/>
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00209

FileName:./Rules/generalConstraints/ISM_ID_00209.sch

Rule Description:

[ISM-ID-00209][Error] Attribute atomicEnergyMarkings must not contain values that have passed their deprecation date.

Code Description:

Traverse the CVE file pulling out deprecated values and their dates. Using the current date determine if the values are being used passed the deprecated date

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00209">

  <!--
Any changes made to this code may also be required for the
following rule files:
- ISM_ID_00170.sch
- ISM_ID_00180.sch
- ISM_ID_00189.sch
- ISM_ID_00199.sch
- ISM_ID_00193.sch
- ISM_ID_00195.sch
- ISM_ID_00197.sch
- ISM_ID_00199.sch
- ISM_ID_00201.sch
- ISM_ID_00203.sch
- ISM_ID_00205.sch
- ISM_ID_00207.sch
- ISM_ID_00209.sch
- ISM_ID_00211.sch
-->
  <sch:rule context="//*[@ism:atomicEnergyMarkings]">
    <sch:let name="curDate" value="translate($ISM_RESOURCE_ELEMENT/@ism:createDate,'-','')"/>
    <sch:let name="depValues"
value="document('._CVE/CVEnumISMAAtomicEnergyMarkings.xml')//cve:CVE/cve:Enumeration/
cve:Term[./@deprecated]/cve:Value"/>
    <sch:let name="depDates"
value="document('._CVE/CVEnumISMAAtomicEnergyMarkings.xml')//cve:CVE/cve:Enumeration/
cve:Term/@deprecated"/>

    <sch:let name="values"
value=" for $each in $depValues return if(index-of(tokenize(./@ism:atomicEnergyMarkings,'
'), $each)>0) then $each else null"/>
    <sch:let name="reportErr"
```

```
value=" for $each in $values return if($curDate >=
translate(subsequence($depDates,index-of($depValues,$each),1),'-','))
then concat('[' ,string($each),' ] is not authorized for use after ',
string(subsequence($depDates,index-of($depValues,$each),1))) else null " />

<sch:assert test="count($reportErr)=0" flag="error">
[ISM-ID-00209][Error] For attribute atomicEnergyMarkings, value(s) <sch:value-of
select="$reportErr"/>
</sch:assert>
</sch:rule>
</sch:pattern>
```


Rule: ISM-ID-00210

FileName:./Rules/generalConstraints/ISM_ID_00210.sch

Rule Description:

[ISM-ID-00210][Warning] Attribute nonUSControls must not contain values that have passed their deprecation date.

Code Description:

Traverse the CVE file pulling out deprecated values and their dates. Using the current date, determine if the values are being used but it is still prior to the deprecated date

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00210">

  <!--
Any changes made to this code may also be required for the
following rule files:
- ISM_ID_00166.sch
- ISM_ID_00179.sch
- ISM_ID_00188.sch
- ISM_ID_00190.sch
- ISM_ID_00192.sch
- ISM_ID_00194.sch
- ISM_ID_00196.sch
- ISM_ID_00198.sch
- ISM_ID_00200.sch
- ISM_ID_00202.sch
- ISM_ID_00204.sch
- ISM_ID_00206.sch
- ISM_ID_00208.sch
- ISM_ID_00210.sch
-->
  <sch:rule context="//*[@ism:nonUSControls]">
    <sch:let name="curDate" value="translate($ISM_RESOURCE_ELEMENT/@ism:createDate,'-','')"/>
    <sch:let name="depValues"
value="document('._CVE/CVEnumISMNonUSControls.xml')//cve:CVE/cve:Enumeration/cve:Term[./
@deprecated]/cve:Value"/>
    <sch:let name="depDates"
value="document('._CVE/CVEnumISMNonUSControls.xml')//cve:CVE/cve:Enumeration/cve:Term/
@deprecated"/>

    <sch:let name="values"
value=" for $each in $depValues return if(index-of(tokenize(./@ism:nonUSControls,' '),
$each)>0) then $each else null"/>
    <sch:let name="reportWarn"
```

```
value=" for $each in $values return if($curDate <
translate(subsequence($depDates,index-of($depValues,$each),1),'-','')) then
concat('[' ,string($each),' ] has been deprecated and is not authorized for used after ' ,
string(subsequence($depDates,index-of($depValues,$each),1))) else null "/>

<sch:assert test="count($reportWarn)=0" flag="warning">
[ISM-ID-00210][Warning] For attribute nonUSControls, value(s) <sch:value-of
select="$reportWarn"/>
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00211

FileName:./Rules/generalConstraints/ISM_ID_00211.sch

Rule Description:

[ISM-ID-00211][Error] Attribute nonUSControls must not contain values that have passed their deprecation date.

Code Description:

Traverse the CVE file pulling out deprecated values and their dates. Using the current date determine if the values are being used passed the deprecated date

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00211">

  <!--
Any changes made to this code may also be required for the
following rule files:
- ISM_ID_00170.sch
- ISM_ID_00180.sch
- ISM_ID_00189.sch
- ISM_ID_00199.sch
- ISM_ID_00193.sch
- ISM_ID_00195.sch
- ISM_ID_00197.sch
- ISM_ID_00199.sch
- ISM_ID_00201.sch
- ISM_ID_00203.sch
- ISM_ID_00205.sch
- ISM_ID_00207.sch
- ISM_ID_00209.sch
- ISM_ID_00211.sch
-->
  <sch:rule context="//*[@ism:nonUSControls]">
    <sch:let name="curDate" value="translate($ISM_RESOURCE_ELEMENT/@ism:createDate,'-','')"/>
    <sch:let name="depValues"
value="document('._CVE/CVEnumISMNonUSControls.xml')//cve:CVE/cve:Enumeration/cve:Term[./
@deprecated]/cve:Value"/>
    <sch:let name="depDates"
value="document('._CVE/CVEnumISMNonUSControls.xml')//cve:CVE/cve:Enumeration/cve:Term/
@deprecated"/>

    <sch:let name="values"
value=" for $each in $depValues return if(index-of(tokenize(./@ism:nonUSControls,' '),
$each)>0) then $each else null"/>
    <sch:let name="reportErr"
```

```
value=" for $each in $values return if($curDate >=
translate(subsequence($depDates,index-of($depValues,$each),1),'-', ''))
then concat('[' ,string($each),'] is not authorized for use after ',
string(subsequence($depDates,index-of($depValues,$each),1))) else null "/>

<sch:assert test="count($reportErr)=0" flag="error">
[ISM-ID-00211][Error] For attribute nonUSControls, value(s) <sch:value-of
select="$reportErr"/>
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00212

FileName:../Rules/generalConstraints/ISM_ID_00212.sch

Rule Description:

[ISM-ID-00212][Error] There must exist at least one element that has attribute resourceElement="true" specified.

Code Description:

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00212">

<sch:rule context="//*">
<sch:assert id="ism00212"
test="count( for $each in /* return if($each/@ism:resourceElement=true()) then $each else
null )>0"
flag="error">
[ISM-ID-00212][Error] There must exist at least one element that has
attribute resourceElement="true" specified.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00213

FileName:../Rules/disseminationControls/ISM_ID_00213.sch

Rule Description:

[ISM-ID-00213][Error] If ISM-CAPCO-RESOURCE and attribute disseminationControls contains the name token [DISPLAYONLY], then attribute displayOnlyTo must be specified.

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If it is a CAPCO resource then we check that if we have an element having attribute disseminationControls with a value of [DISPLAYONLY] then the attribute displayOnlyTo is specified and does not have an empty value set.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00213">

<sch:rule context="//*[@ism:disseminationControls]">
<sch:assert id="ism00031"
test=" if(index-of(tokenize(./@ism:disseminationControls,' '), 'DISPLAYONLY')>0) then ./
@ism:displayOnlyTo else true() "
flag="error">
[ISM-ID-00213][Error] If ISM-CAPCO-RESOURCE and attribute
disseminationControls contains the name token [DISPLAYONLY], then attribute displayOnlyTo
must be specified.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00214

FileName:../Rules/releasableTo/ISM_ID_00214.sch

Rule Description:

[ISM-ID-00214][Error] If ISM-CAPCO-RESOURCE then attribute releasableTo must start with [USA].

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If the attribute releasableTo is specified, then we make sure that it starts with [USA].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00214">

<sch:rule context="//*[@ism:releasableTo]">

<sch:let name="dissemTok" value="tokenize(./@ism:disseminationControls,' ')" />
<sch:assert id="ism00032"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else index-of(tokenize(./
@ism:releasableTo,' '), 'USA')=1 "
flag="error">
[ISM-ID-00214][Error] If ISM-CAPCO-RESOURCE then attribute
releasableTo must start with [USA].
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: ISM-ID-00215

FileName:../Rules/disseminationControls/ISM_ID_00215.sch

Rule Description:

[ISM-ID-00215][Error] If ISM-CAPCO-RESOURCE and attribute disseminationControls contains the name token [DISPLAYONLY], then attribute classification must have a value of [TS], [S], or [C].

Code Description:

If CAPCO rules do not apply to the document then the rule does not apply and we return true. If it is a CAPCO resource then we check that if we have an element having attribute disseminationControls with a value of [DISPLAYONLY] then we also have the attribute classification specified with a value of [C], [S], or [TS] on the same element.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="ISM-ID-00215">

<sch:rule context="//*[@ism:disseminationControls]">
<sch:assert id="ism00215"
test=" if(not($ISM_CAPCO_RESOURCE)) then true() else if(index-of(tokenize(./
@ism:disseminationControls, ' '), 'DISPLAYONLY')>0) then matches(./
@ism:classification, '^(TS|S|C)$') else true()"
flag="error">
[ISM-ID-00215][Error] If ISM-CAPCO-RESOURCE and attribute
disseminationControls contains the name token [DISPLAYONLY],
then attribute classification must have a value of [TS], [S], or [C].
</sch:assert>
</sch:rule>
</sch:pattern>
```