

IRM 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 resource to search and read but for questions and debates the source files should be consulted.

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

FileName:./IRM_XML.sch

Code Description:

This is the root file for the IRM 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"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
queryBinding="xslt2">
<sch:ns uri="http://www.w3.org/2001/XMLSchema" prefix="xsd"/>
<sch:ns uri="urn:us:gov:ic:ism" prefix="ism"/>
<sch:ns uri="urn:us:gov:ic:irm" prefix="irm"/>
<sch:ns uri="http://metadata.dod.mil/mdr/ns/DDMS/3.0/" prefix="ddms"/>
<sch:ns uri="urn:us:gov:ic:cve:v1" prefix="cve"/>
<sch:ns uri="http://www.w3.org/1999/xlink" prefix="xlink"/>
<sch:ns uri="http://www.w3.org/1999/XSL/Transform" prefix="xsl"/>
<sch:ns uri="date:time:function" prefix="dtf"/>

<!-- (U) Resources -->
<sch:let name="coverageFipsDigraphList"
value="document('.../CVE/IRM/CVEnumIRMCoverageFIPSDigraph.xml')//cve:CVE/
cve:Enumeration/cve:Term/cve:Value"/>
<sch:let name="coverageIso3166DigraphList"
value="document('.../CVE/IRM/CVEnumIRMCoverageISO3166Digraph.xml')//cve:CVE/
cve:Enumeration/cve:Term/cve:Value"/>
<sch:let name="coverageIso3166TrigraphList"
value="document('.../CVE/IRM/CVEnumIRMCoverageISO3166Trigraph.xml')//cve:CVE/
cve:Enumeration/cve:Term/cve:Value"/>
<sch:let name="coverageIso3166-2SubCountryList"
value="document('.../CVE/IRM/CVEnumIRMCoverageISO3166-2SubCountry.xml')//cve:CVE/
cve:Enumeration/cve:Term/cve:Value"/>
<sch:let name="iso639DigraphList"
value="document('.../CVE/IRM/CVEnumIRMISO639Digraph.xml')//cve:CVE/cve:Enumeration/
cve:Term/cve:Value"/>
<sch:let name="iso639-2TrigraphList"
value="document('.../CVE/IRM/CVEnumIRMISO639-2Trigraph.xml')//cve:CVE/cve:Enumeration/
cve:Term/cve:Value"/>
<sch:let name="iso639-3TrigraphList"
```

```

value="document(' ../../CVE/IRM/CVEnumIRMISO639-3Trigraph.xml')//cve:CVE/cve:Enumeration/
cve:Term/cve:Value"/>
<sch:let name="mimeTypeList"
value="document(' ../../CVE/IRM/CVEnumIRMMimeType.xml')//cve:Value"/>
<sch:let name="compoundCountryCodeQualifierTypeList"
value="document(' ../../CVE/IRM/CVEnumIRMCompoundCountryCodeQualifierType.xml')//
cve:Value"/>
<sch:let name="compoundLanguageQualifierTypeList"
value="document(' ../../CVE/IRM/CVEnumIRMCompoundLanguageQualifierType.xml')//cve:Value"/>

<!-- ***** -->
<!-- * General Global Variables * -->
<!-- ***** -->

<sch:let name="currentYear" value="year-from-dateTime(current-dateTime())"/>
<sch:let name="timeZoneRegEx" value="'Z|[\+-]\d{2}:\d{2}'"/>
<sch:let name="endsWithTimeZoneRegEx" value="concat('^.*',$timeZoneRegEx,'$')"/>
<sch:let name="startDateTimeTemplate" value="'0001-01-01T00:00:00.000'"/>
<sch:let name="endDateTimeTemplate" value="'9999-12-01T23:59:59.999'"/>
<sch:let name="defaultTimeZone" value="'Z'"/>

<!-- ***** -->
<!-- * Abstract Rule and Pattern Includes * -->
<!-- ***** -->

<sch:include href="./Lib/ValidateValueExistenceInList.sch"/>
<sch:include href="./Lib/DateListYearRangeRule.sch"/>
<sch:include href="./Lib/DateYearRangeRule.sch"/>
<sch:include href="./Lib/CompareDateTimes.sch"/>

<!-- ***** -->
<!-- * Custom XSLT2 Function Definitions * -->
<!-- ***** -->
<!--
Returns the maximum day of the month for an xs:dateTime as an xs:string.
@param {xs:dateTime} date The date time from which to get the month
@returns {xs:string} String representation of the maximum day of the month
-->
<xsl:function name="dtf:getMaxDay" as="xs:string">
<xsl:param name="date" as="xs:dateTime"/>
<xsl:variable name="month" select="number(dtf:getMonth(string($date)))"/>
<xsl:choose>
<xsl:when test="$month = (1,3,5,7,8,10,12)">
<xsl:value-of select="31"/>
</xsl:when>
<xsl:when test="$month = (4,6,9,11)">
<xsl:value-of select="30"/>
</xsl:when>

```

```

<xsl:otherwise>
<xsl:choose>
<xsl:when test="dtf:isLeapYear(string($date))">
<xsl:value-of select="29"/>
</xsl:when>
<xsl:otherwise>
<xsl:value-of select="28"/>
</xsl:otherwise>
</xsl:choose>
</xsl:otherwise>
</xsl:choose>
</xsl:function>

<!--
@param {xs:date} date String representation of a date
@returns {xs:boolean} Returns true if the date provided occurs in a
leap year; otherwise returns false.
-->
<xsl:function name="dtf:isLeapYear" as="xs:boolean">
<xsl:param name="date" as="xs:string"/>
<xsl:variable name="year" as="xs:integer" select="xs:integer(dtf:getYear($date))"/>
<xsl:choose>
<xsl:when test="$year mod 100 = 0">
<xsl:choose>
<xsl:when test="$year mod 400 = 0">
<xsl:value-of select="true()"/>
</xsl:when>
<xsl:otherwise>
<xsl:value-of select="false()"/>
</xsl:otherwise>
</xsl:choose>
</xsl:when>
<xsl:otherwise>
<xsl:choose>
<xsl:when test="$year mod 4 = 0">
<xsl:value-of select="true()"/>
</xsl:when>
<xsl:otherwise>
<xsl:value-of select="false()"/>
</xsl:otherwise>
</xsl:choose>
</xsl:otherwise>
</xsl:choose>
</xsl:function>

<!--
Replaces the day portion of the provided dateTime with the new day provided.
@param {xs:dateTime} dateTime An xs:dateTime to be updated with new day.

```

```

@param {xs:string} newDayString String representation of day portion of a date.
@returns {xs:dateTime} Returns new xs:dateTime with updated day portion.
leap year; otherwise returns false.
-->
<xsl:function name="dtf:replaceDateTimeDay" as="xs:dateTime">
<xsl:param name="dateTime" as="xs:dateTime"/>
<xsl:param name="newDayString" as="xs:string"/>
<xsl:variable name="beforeDay" select="substring(string($dateTime), 1, 8)"/>
<xsl:variable name="afterDay" select="substring(string($dateTime), 11)"/>
<xsl:value-of select="concat($beforeDay, $newDayString, $afterDay)"/>
</xsl:function>

<!--
Returns a string representation of the year portion of the date
represented by the provided string.
@param {xs:string} dateString String representation of a date in one
of the allowable formats.
@returns {xs:string} String representation of the year portion of the
date represented by the provided string.
-->
<xsl:function name="dtf:getYear" as="xs:string">
<xsl:param name="dateString" as="xs:string"/>
<xsl:value-of select="substring(dtf:removeTimeZone($dateString), 1, 4)"/>
</xsl:function>

<!--
Returns a string representation of the month portion of the date
represented by the provided string.
@param {xs:string} dateString String representation of a date in one
of the allowable formats.
@returns {xs:string} String representation of the month portion of the
date represented by the provided string.
-->
<xsl:function name="dtf:getMonth" as="xs:string">
<xsl:param name="dateString" as="xs:string"/>
<xsl:value-of select="substring(dtf:removeTimeZone($dateString), 6, 2)"/>
</xsl:function>

<!--
Returns a string representation of the day portion of the date
represented by the provided string.
@param {xs:string} dateString String representation of a date in one
of the allowable formats.
@returns {xs:string} String representation of the day portion of the
date represented by the provided string.
-->
<xsl:function name="dtf:getDay" as="xs:string">
<xsl:param name="dateString" as="xs:string"/>

```

```

<xsl:value-of select="substring(dtf:removeTimeZone($dateString), 9, 2)"/>
</xsl:function>

<!--
Returns a string representation of the timezone portion of the date
represented by the provided string.
@param {xs:string} dateString String representation of a date in one
of the allowable formats.
@returns {xs:string} String representation of the timezone portion of
the date represented by the provided string.
-->
<xsl:function name="dtf:getTimeZone" as="xs:string">
  <xsl:param name="dateString" as="xs:string"/>
  <xsl:variable name="dateTimeEndingWithTimezone" as="xs:string"
  select="concat('^.*(', $timezoneRegex, ')$')"/>
  <xsl:choose>
    <xsl:when test="matches($dateString, $dateTimeEndingWithTimezone)">
      <xsl:value-of select="replace($dateString, $dateTimeEndingWithTimezone, '$1')"/>
    </xsl:when>
    <xsl:otherwise>
      <xsl:value-of select="$defaultTimeZone"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:function>

<!--
Returns true if the year portion of the date represented by the provided
string contains four (4) digits; otherwise returns false.
@param {xs:string} dateString String representation of a date in one
of the allowable formats.
@returns {xs:string} true if the year portion of the date represented by
the provided string contains four (4) digits; otherwise returns false.
-->
<xsl:function name="dtf:yearPortionHasFourDigits" as="xs:boolean">
  <xsl:param name="dateString" as="xs:string"/>
  <xsl:variable name="dateWithOnlyFourDigitYearAndOptionalTimeZoneRegex" as="xs:string"
  select="concat('^\d{4}(', $timezoneRegex, ')?$')"/>
  <xsl:variable name="dateStartingWithFourDigitYearRegex" as="xs:string" select="'^\d{4}-.*
  $'"/>
  <xsl:value-of select="matches($dateString,
  $dateWithOnlyFourDigitYearAndOptionalTimeZoneRegex) or matches($dateString,
  $dateStartingWithFourDigitYearRegex)"/>
</xsl:function>

<!--
Removes the timezone portion of the date represented by the provided
string and returns all remaining portions.
@param {xs:string} dateString String representation of a date in one

```

of the allowable formats.

@returns {xs:string} String representation of a date without a timezone portion.

-->

```
<xsl:function name="dtf:removeTimeZone" as="xs:string">
<xsl:param name="dateString" as="xs:string"/>
<xsl:value-of select="replace($dateString, $timeZoneRegEx, '')"/>
</xsl:function>
```

<!--

Uses the template provided to fill in missing portions of the string representation of a dateTime provided and returns a full xs:dateTime. The dateString provided must not contain a timezone.

@param {xs:string} dateString String representation of a date in one of the allowable formats.

@param {xs:string} dateTemplateString String template of a default date from which to pad missing portions of the dateString parameter.

@returns {xs:dateTime} An xs:dateTime represented by the string date provided.

-->

```
<xsl:function name="dtf:padDateTimeWithTemplate" as="xs:dateTime">
<xsl:param name="dateString" as="xs:string"/>
<xsl:param name="dateTemplateString" as="xs:string"/>
<xsl:value-of select="concat($dateString, substring($dateTemplateString, string-length(normalize-space($dateString))+1))"/>
</xsl:function>
```

<!--

Returns true if the string provided represents an allowable dateTime format; false, otherwise. The allowable dateTime formats are defined in the DES for the PUBS.XML specification.

@returns {xs:boolean} Returns true if the string provided represents an allowable dateTime format; false, otherwise.

-->

```
<xsl:function name="dtf:isAllowableDateTimeFormat" as="xs:boolean">
<xsl:param name="input" as="xs:string"/>
<xsl:variable name="trimmedInput" as="xs:string" select="normalize-space($input)"/>
```

<!-- year -->

```
<xsl:variable name="YYYY" as="xs:string" select="'^\d{4}(Z|[\+-]\d{2}:\d{2})?$'"/>
```

<!-- year, month -->

```
<xsl:variable name="YYYY-MM" as="xs:string" select="'^\d{4}-\d{2}(Z|[\+-]\d{2}:\d{2})?$'"/>
```

<!-- year, month, day -->

```
<xsl:variable name="YYYY-MM-DD" as="xs:string"
select="'^\d{4}-\d{2}-\d{2}(Z|[\+-]\d{2}:\d{2})?$'"/>
```



```

<!-- year, month, day, hour, minute -->
<xsl:variable name="YYYY-MM-DDThh-mm" as="xs:string"
select="'^\d{4}-\d{2}-\d{2}T\d{2}:\d{2}(Z|[\+-]\d{2}:\d{2})?$',"/>

<!-- year, month, day, hour, minute, seconds, optional milliseconds -->
<xsl:variable name="YYYY-MM-DDThh-mm-ss" as="xs:string"
select="'^\d{4}-\d{2}-\d{2}T\d{2}:\d{2}:\d{2}(\.\d{1,3})?(Z|[\+-]\d{2}:\d{2})?$',"/>

<xsl:value-of select=" matches($trimmedInput, $YYYY) or matches($trimmedInput, $YYYY-MM)
or matches($trimmedInput, $YYYY-MM-DD) or matches($trimmedInput, $YYYY-MM-DDThh-mm) or
matches($trimmedInput, $YYYY-MM-DDThh-mm-ss) " />
</xsl:function>

<!--
Returns the earliest xs:dateTime possible for the provided string
representation of a dateTime. Fills in missing portions of the
dateTime with the earliest possible values. Default values for missing
portions:
MM = 01
DD = 01
hh = 00
mm = 00
ss = 00
s = 000
@param {xs:string} dateString String representation of a date in one
of the allowable formats.
@returns {xs:dateTime} The earliest xs:dateTime possible for the
provided string representation of a dateTime.
-->
<xsl:function name="dtf:startDate" as="xs:dateTime">
<xsl:param name="dateString" as="xs:string"/>
<xsl:variable name="timeZonePortion" select="dtf:getTimeZone($dateString)"/>
<xsl:variable name="dateTimePortion" select="dtf:removeTimeZone($dateString)"/>
<xsl:variable name="outputDate"
select="dtf:padDateTimeWithTemplate($dateTimePortion, $startDateTimeTemplate)"/>
<xsl:value-of select="concat($outputDate, $timeZonePortion)"/>
</xsl:function>

<!--
Returns the latest xs:dateTime possible for the provided string
representation of a dateTime. Fills in missing portions of the
dateTime with the latest possible values. Default values for missing
portions:
MM = 12
DD = maximum day of the month
hh = 23
mm = 59
ss = 59

```

```

s = 999
@param {xs:string} dateString String representation of a date in one
of the allowable formats.
@returns {xs:dateTime} The latest xs:dateTime possible for the
provided string representation of a dateTime.
-->
<xsl:function name="dtf:endDate" as="xs:dateTime">
<xsl:param name="input" as="xs:string"/>
<xsl:variable name="timeZonePortion" select="dtf:getTimeZone($input)"/>
<xsl:variable name="dateTimePortion" select="dtf:removeTimeZone($input)"/>
<xsl:variable name="outputDate"
select="dtf:padDateTimeWithTemplate($dateTimePortion, $endDateTimeTemplate)"/>
<xsl:variable name="outputWithCorrectedDay"
select="dtf:replaceDateTimeDay($outputDate, dtf:getMaxDay($outputDate))"/>
<xsl:choose>
<xsl:when test="dtf:getDay($input)">
<xsl:value-of select="concat($outputDate, $timeZonePortion)"/>
</xsl:when>
<xsl:otherwise>
<xsl:value-of select="concat($outputWithCorrectedDay, $timeZonePortion)"/>
</xsl:otherwise>
</xsl:choose>
</xsl:function>

<!--
Calculates the date range implied for both primary and secondary and
determines if there is any overlap between the two ranges. Overlap is
defined as the start of primary date range less than or equal to the
end of secondary date range, inclusive, and the start of the secondary
date range less than or equal to the end of the primary date range.
Returns true if there is any overlap; otherwise, returns false.
@param {xs:string} primary String representation of a date in one
of the allowable formats.
@param {xs:string} secondary String representation of a date in one
of the allowable formats.
@returns {xs:boolean} Returns true if the date ranges implied by primary
and secondary overlap at all; otherwise, returns false.
-->
<xsl:function name="dtf:overlaps" as="xs:boolean">
<xsl:param name="primary" as="xs:string"/>
<xsl:param name="secondary" as="xs:string"/>
<xsl:variable name="primaryStart" as="xs:dateTime" select="dtf:startDate($primary)"/>
<xsl:variable name="primaryEnd" as="xs:dateTime" select="dtf:endDate($primary)"/>
<xsl:variable name="secondaryStart" as="xs:dateTime" select="dtf:startDate($secondary)"/>
<xsl:variable name="secondaryEnd" as="xs:dateTime" select="dtf:endDate($secondary)"/>
<xsl:value-of select="$primaryStart &lt;= $secondaryEnd and $secondaryStart &lt;=
$primaryEnd"/>
</xsl:function>

```

```

<!--
Determines if the date range implied by the string representation in
primary is stricly before the date range implied by the string
representation in secondary. Returns true if the end of the date
range implied by primary is less than the start of the date range
implied by secondary; otherwise, returns false.
@param {xs:string} primary String representation of a date in one
of the allowable formats.
@param {xs:string} secondary String representation of a date in one
of the allowable formats.
@returns {xs:boolean} Returns true if the date range implied by primary
is stricly earlier than the date range implied by secondary; otherwise,
returns false.
-->
<xsl:function name="dtf:isBefore" as="xs:boolean">
<xsl:param name="primary" as="xs:string"/>
<xsl:param name="secondary" as="xs:string"/>
<xsl:variable name="primaryEnd" as="xs:dateTime" select="dtf:endDate($primary)"/>
<xsl:variable name="secondaryStart" as="xs:dateTime" select="dtf:startDate($secondary)"/>
<xsl:value-of select="$primaryEnd < $secondaryStart"/>
</xsl:function>

<!--
Determines if the date range implied by the string representation in
primary is stricly after the date range implied by the string
representation in secondary. Returns true if the end of the date
range implied by primary is less than the start of the date range
implied by secondary; otherwise, returns false.
@param {xs:string} primary String representation of a date in one
of the allowable formats.
@param {xs:string} secondary String representation of a date in one
of the allowable formats.
@returns {xs:boolean} Returns true if the date range implied by primary
is stricly later than the date range implied by secondary; otherwise,
returns false.
-->
<xsl:function name="dtf:isAfter" as="xs:boolean">
<xsl:param name="primary" as="xs:string"/>
<xsl:param name="secondary" as="xs:string"/>
<xsl:variable name="primaryStart" as="xs:dateTime" select="dtf:startDate($primary)"/>
<xsl:variable name="secondaryEnd" as="xs:dateTime" select="dtf:endDate($secondary)"/>
<xsl:value-of select="$secondaryEnd < $primaryStart"/>
</xsl:function>

<!--
Determines if the date range implied by the string representation in
primary satisfies the comparison to the date range implied by secondary

```

using the provided comparison operator; otherwise, returns false.

Both primary and secondary must be in one of the allowable formats and represent dates with four digits in the year portion.

@param {xs:string} primary String representation of a date in one of the allowable formats.

@param {xs:string} secondary String representation of a date in one of the allowable formats.

@returns {xs:boolean} Returns true if the date range implied by primary satisfies the comparison to the date range implied by secondary using the provided comparison operator; otherwise, returns false.

```
-->
<xsl:function name="dtf:compareDateTimeRanges" as="xs:boolean">
  <xsl:param name="primary" as="xs:string"/>
  <xsl:param name="operator" as="xs:string"/>
  <xsl:param name="secondary" as="xs:string"/>
  <xsl:variable name="primaryAndSecondaryYearPortionsHaveFourDigits" as="xs:boolean"
    select="dtf:yearPortionHasFourDigits($primary) and
    dtf:yearPortionHasFourDigits($secondary)"/>
  <xsl:choose>
    <xsl:when test="$primaryAndSecondaryYearPortionsHaveFourDigits">
      <xsl:variable name="primaryStart" as="xs:dateTime" select="dtf:startDate($primary)"/>
      <xsl:variable name="primaryEnd" as="xs:dateTime" select="dtf:endDate($primary)"/>
      <xsl:variable name="secondaryStart" as="xs:dateTime" select="dtf:startDate($secondary)"/>
      <xsl:variable name="secondaryEnd" as="xs:dateTime" select="dtf:endDate($secondary)"/>
      <xsl:choose>
        <!-- 'Less Than' Edge Case -->
        <!-- 2010-01-01T00:00:00.000Z < 2010 -->
        <xsl:when test="($operator = 'lt' or $operator = '<') and (($primaryStart = $primaryEnd
        and $primaryStart = $secondaryStart) or ($primaryStart = $primaryEnd and $primaryStart =
        $secondaryEnd) or ($secondaryStart = $secondaryEnd and $primaryStart = $secondaryStart))">
          <xsl:value-of select="false()"/>
        </xsl:when>

        <!-- 'Greater Than' Edge Case -->
        <!-- 2010-12-31T23:59:59.999Z > 2010 -->
        <xsl:when test="($operator = 'gt' or $operator = '>') and (($primaryStart = $primaryEnd
        and $primaryEnd = $secondaryEnd) or ($primaryStart = $primaryEnd and $primaryEnd =
        $secondaryStart) or ($secondaryStart = $secondaryEnd and $primaryEnd = $secondaryEnd))">
          <xsl:value-of select="false()"/>
        </xsl:when>

        <!-- 'Less Than' and 'Less Than or Equal' -->
        <xsl:when test="$operator = 'lt' or $operator = '<=' or $operator = '<='">
          <xsl:value-of select="dtf:isBefore($primary, $secondary) or dtf:overlaps($primary,
          $secondary)"/>
        </xsl:when>
```

```

<!-- 'Greater Than' and 'Greater Than or Equal' -->
<xsl:when test="$operator = 'gt' or $operator = '>';' or $operator = '>=;'>
<xsl:value-of select="dtf:isAfter($primary, $secondary) or dtf:overlaps($primary,
$secondary)"/>
</xsl:when>

<!-- Default to false() -->
<xsl:otherwise>
<xsl:value-of select="false()"/>
</xsl:otherwise>
</xsl:choose>
</xsl:when>
<xsl:otherwise>
<xsl:value-of select="false()"/>
</xsl:otherwise>
</xsl:choose>
</xsl:function>

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

<!--(U) ddmsConstraints-->
<sch:include href="./Rules/ddmsConstraints/IRM_ID_00031.sch"/>
<sch:include href="./Rules/ddmsConstraints/IRM_ID_00032.sch"/>
<sch:include href="./Rules/ddmsConstraints/IRM_ID_00033.sch"/>
<sch:include href="./Rules/ddmsConstraints/IRM_ID_00034.sch"/>
<sch:include href="./Rules/ddmsConstraints/IRM_ID_00035.sch"/>

<!--(U) generalConstraints-->
<sch:include href="./Rules/generalConstraints/IRM_ID_00011.sch"/>
<sch:include href="./Rules/generalConstraints/IRM_ID_00012.sch"/>
<sch:include href="./Rules/generalConstraints/IRM_ID_00013.sch"/>
<sch:include href="./Rules/generalConstraints/IRM_ID_00014.sch"/>
<sch:include href="./Rules/generalConstraints/IRM_ID_00015.sch"/>
<sch:include href="./Rules/generalConstraints/IRM_ID_00016.sch"/>
<sch:include href="./Rules/generalConstraints/IRM_ID_00017.sch"/>
<sch:include href="./Rules/generalConstraints/IRM_ID_00018.sch"/>
<sch:include href="./Rules/generalConstraints/IRM_ID_00019.sch"/>
<sch:include href="./Rules/generalConstraints/IRM_ID_00020.sch"/>
<sch:include href="./Rules/generalConstraints/IRM_ID_00021.sch"/>
<sch:include href="./Rules/generalConstraints/IRM_ID_00022.sch"/>
<sch:include href="./Rules/generalConstraints/IRM_ID_00023.sch"/>
<sch:include href="./Rules/generalConstraints/IRM_ID_00024.sch"/>
<sch:include href="./Rules/generalConstraints/IRM_ID_00029.sch"/>
<sch:include href="./Rules/generalConstraints/IRM_ID_00030.sch"/>
<sch:include href="./Rules/generalConstraints/IRM_ID_00037.sch"/>

```

```
<!--(U) globalConstraints-->
<sch:include href="./Rules/globalConstraints/IRM_ID_00002.sch"/>

<!--(U) ismConstraints-->
<sch:include href="./Rules/ismConstraints/IRM_ID_00025.sch"/>

<!--(U) valueEnumerationConstraints-->
<sch:include href="./Rules/valueEnumerationConstraints/IRM_ID_00001.sch"/>
<sch:include href="./Rules/valueEnumerationConstraints/IRM_ID_00003.sch"/>
<sch:include href="./Rules/valueEnumerationConstraints/IRM_ID_00005.sch"/>
<sch:include href="./Rules/valueEnumerationConstraints/IRM_ID_00006.sch"/>
<sch:include href="./Rules/valueEnumerationConstraints/IRM_ID_00007.sch"/>
<sch:include href="./Rules/valueEnumerationConstraints/IRM_ID_00008.sch"/>
<sch:include href="./Rules/valueEnumerationConstraints/IRM_ID_00009.sch"/>
<sch:include href="./Rules/valueEnumerationConstraints/IRM_ID_00010.sch"/>
<sch:include href="./Rules/valueEnumerationConstraints/IRM_ID_00027.sch"/>
<sch:include href="./Rules/valueEnumerationConstraints/IRM_ID_00028.sch"/>

<!--(U) xlinkConstraints-->
<sch:include href="./Rules/xlinkConstraints/IRM_ID_00036.sch"/>
</sch:schema>
```

FileName: //Lib/DateListYearRangeRule.sch

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Abstract rule, which asserts that each date contained within
the list $dateList has a year value within the range
$minYear and $maxYear, inclusive. If any value in $dateList is not a valid
date format, then we return true because we cannot guarantee the value
provided is not allowed.-->
<sch:rule xmlns:sch="http://purl.oclc.org/dsdl/schematron" abstract="true"
id="abs.dateListYearRangeRule">
<sch:assert test=" every $date in tokenize(normalize-space(string-join($dateList,
' '), ' ')) satisfies if(not(dtf:isAllowableDateTimeFormat(string($date))))
then true() else xs:integer(substring(string($date), 1, 4)) <= $maxYear and
xs:integer(substring(string($date), 1, 4)) >= $minYear"
flag="error">
<sch:value-of select="$errMsg"/>
</sch:assert>
</sch:rule>
```

FileName: //Lib/DateYearRangeRule.sch

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Abstract rule, which asserts that the date contained
within $dateValue has a year value within the range
$minYear and $maxYear, inclusive. -->
<sch:rule xmlns:sch="http://purl.oclc.org/dsdl/schematron" abstract="true"
id="abs.dateYearRangeRule">
<sch:assert test=" if(not(dtf:yearPortionHasFourDigits(string($dateValue)))) then
false() else xs:integer(substring(string($dateValue), 1, 4)) <= $maxYear and
xs:integer(substring(string($dateValue), 1, 4)) >= $minYear"
flag="error">
<sch:value-of select="$errMsg"/>
</sch:assert>
</sch:rule>
```


Rule: IRM-ID-00001

FileName:./Rules/valueEnumerationConstraints/IRM_ID_00001.sch

Rule Description:

[IRM-ID-00001][Error] If element ddms:countryCode has attribute ddms:qualifier specified as [urn:us:gov:ic:cvenum:irm:coverage:fips:digraph:v1] then the value of attribute ddms:value must be in CVENumIRMCoverageFIPSDigraph.xml. Human Readable: FIPS CountryCodes must appear in the FIPS CVE.

Code Description:

This rule uses an abstract pattern to consolidate logic. It checks that the value in parameter \$searchTerm is contained in the parameter \$list. The parameter \$searchTerm is relative in scope to the parameter \$context. The value for the parameter \$list is a variable defined in the main document (IRM_XML.sch), which reads values from a specific CVE file.

Schematron Code:

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

<sch:param name="ruleText"
value="' [IRM-ID-00001][Error] If element ddms:countryCode has attribute ddms:qualifier
specified as [urn:us:gov:ic:cvenum:irm:coverage:fips:digraph:v1] then the value of
attribute ddms:value must be in CVENumIRMCoverageFIPSDigraph.xml. Human Readable: FIPS
CountryCodes must in the the FIPS CVE. '"/>

<sch:param name="codeDesc"
value="' This rule uses an abstract pattern to consolidate logic. It checks that the value
in parameter $searchTerm is contained in the parameter $list. The parameter $searchTerm
is relative in scope to the parameter $context. The value for the parameter $list is a
variable defined in the main document (IRM_XML.sch), which reads values from a specific
CVE file. '"/>

<sch:param name="context"
value="ddms:countryCode[@ddms:qualifier='urn:us:gov:ic:cvenum:irm:coverage:fips:digraph:v1']"/>
<sch:param name="searchTerm" value="@ddms:value"/>
<sch:param name="list" value="$coverageFipsDigraphList"/>
<sch:param name="errMsg"
value="' [IRM-ID-00001][Error] If element ddms:countryCode has attribute ddms:qualifier
specified as [urn:us:gov:ic:cvenum:irm:coverage:fips:digraph:v1] then the value of
attribute ddms:value must be in CVENumIRMCoverageFIPSDigraph.xml. Human Readable: FIPS
CountryCodes must in the the FIPS CVE. '"/>
</sch:pattern>
```

Rule: IRM-ID-00002

FileName:./Rules/globalConstraints/IRM_ID_00002.sch

Rule Description:

[IRM-ID-00002][Warning] For every optional attribute that exists in the document, a non-whitespace value must be specified. Human Readable: If an optional attribute is specified, it must have a value.

Code Description:

The pattern applies to any element that contains one or more of the optional attributes. It joins each of these attribute values, if present, into a larger space-separated string. It then breaks this string into tokens at each space character. If the length of the normalized, space-separated attribute string is greater than zero, that means that at least one of the tokens is a non-null, non-whitespace value. If that is true and every token in the list is a non-null, non-whitespace value, then the rule passes and true is returned.

Schematron Code:

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

<sch:rule context="*[ @irm:taskingSystem | @irm:otherDiscipline | @irm:dateApproved |
@irm:othersubdisciplineTechnique | @irm:indicator | @irm:precedence | @irm:order ]">
<sch:let name="attributeList"
value=" normalize-space(string-join(( string(@irm:taskingSystem),
string(@irm:otherDiscipline), string(@irm:dateApproved),
string(@irm:othersubdisciplineTechnique), string(@irm:indicator), string(@irm:precedence),
string(@irm:order)), ' ')) "/>
<sch:assert id="IRM-00002"
test="string-length($attributeList)>0 and (every $token in tokenize($attributeList, '
') satisfies string-length($token)>0 )"
flag="warning">
[IRM-ID-00002][Warning] For every optional attribute that exists in the document,
a non-whitespace value must be specified.
</sch:assert>
</sch:rule>

</sch:pattern>
```

Rule: IRM-ID-00003

FileName:./Rules/valueEnumerationConstraints/IRM_ID_00003.sch

Rule Description:

[IRM-ID-00003][Error] If element ddms:countryCode has attribute ddms:qualifier specified as [urn:us:gov:ic:cvenum:irm:coverage:iso3166:trigraph:v1] then the value of attribute ddms:value must be in CVEnumIRMCoverageISO3166trigraph.xml. Human Readable: ISO trigraph CountryCodes must appear in the ISO trigraph CVE

Code Description:

This rule uses an abstract pattern to consolidate logic. It checks that the value in parameter \$searchTerm is contained in the parameter \$list. The parameter \$searchTerm is relative in scope to the parameter \$context. The value for the parameter \$list is a variable defined in the main document (IRM_XML.sch), which reads values from a specific CVE file.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="IRM-ID-00003"
is-a="ValidateValueExistenceInList">

<sch:param name="ruleText"
value="' [IRM-ID-00003][Error] If element ddms:countryCode has attribute ddms:qualifier
specified as [urn:us:gov:ic:cvenum:irm:coverage:iso3166:trigraph:v1] then the value of
attribute ddms:value must be in CVEnumIRMCoverageISO3166trigraph.xml. Human Readable: ISO
trigraph CountryCodes must in the the ISO trigraph CVE. '"/>

<sch:param name="codeDesc"
value="' This rule uses an abstract pattern to consolidate logic. It checks that the value
in parameter $searchTerm is contained in the parameter $list. The parameter $searchTerm
is relative in scope to the parameter $context. The value for the parameter $list is a
variable defined in the main document (IRM_XML.sch), which reads values from a specific
CVE file. '"/>

<sch:param name="context"
value="ddms:countryCode[@ddms:qualifier='urn:us:gov:ic:cvenum:irm:coverage:iso3166:trigraph:v1']"
>
<sch:param name="searchTerm" value="@ddms:value"/>
<sch:param name="list" value="$coverageIso3166TrigraphList"/>
<sch:param name="errMsg"
value="' [IRM-ID-00003][Error] If element ddms:countryCode has attribute ddms:qualifier
specified as [urn:us:gov:ic:cvenum:irm:coverage:iso3166:trigraph:v1] then the value of
attribute ddms:value must be in CVEnumIRMCoverageISO3166trigraph.xml. Human Readable: ISO
trigraph CountryCodes must in the the ISO trigraph CVE. '"/>
</sch:pattern>
```

Rule: IRM-ID-00005

FileName:./Rules/valueEnumerationConstraints/IRM_ID_00005.sch

Rule Description:

[IRM-ID-00005][Error] If element ddms:language has the attribute ddms:qualifier value of [urn:us:gov:ic:cvenum:irm:iso639:digraph:v1] then the value of attribute ddms:value must be in CVENumIRMISO639Digraph.xml and no country code portion may be specified in the Language element value. Human Readable: ISO 639 digraph language codes must be in the ISO 639 digraph CVE.

Code Description:

This rule uses an abstract pattern to consolidate logic. It checks that the value in parameter \$searchTerm is contained in the parameter \$list. The parameter \$searchTerm is relative in scope to the parameter \$context. The value for the parameter \$list is a variable defined in the main document (IRM_XML.sch), which reads values from a specific CVE file. This rule can directly check if the value of element Language is in the appropriate list because if a country code portion is specified in the element ddms:language's value, then the value of element ddms:language will not be found in the appropriate list and the assertion will fail as expected.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="IRM-ID-00005"
is-a="ValidateValueExistenceInList">

<sch:param name="ruleText"
value="' [IRM-ID-00005][Error] If element ddms:language has the attribute ddms:qualifier
value of [urn:us:gov:ic:cvenum:irm:iso639:digraph:v1] then the value of attribute
ddms:value must be in CVENumIRMISO639Digraph.xml and no country code portion may be
specified in the Language element value. Human Readable: ISO 639 digraph language codes
must be in the ISO 639 digraph CVE. '"/>

<sch:param name="codeDesc"
value="' This rule uses an abstract pattern to consolidate logic. It checks that the value
in parameter $searchTerm is contained in the parameter $list. The parameter $searchTerm
is relative in scope to the parameter $context. The value for the parameter $list is a
variable defined in the main document (IRM_XML.sch), which reads values from a specific
CVE file. '"/>

<sch:param name="context"
value="irm:DescribedItemIRM//
ddms:language[@ddms:qualifier='urn:us:gov:ic:cvenum:irm:iso639:digraph:v1']"/>
<sch:param name="searchTerm" value="@ddms:value"/>
<sch:param name="list" value="$iso639DigraphList"/>
<sch:param name="errMsg"
value="' [IRM-ID-00005][Error] If element ddms:language has the attribute ddms:qualifier
value of [urn:us:gov:ic:cvenum:irm:iso639:digraph:v1] then the value of attribute
ddms:value must be in CVENumIRMISO639Digraph.xml and no country code portion may be
```

```
specified in the Language element value. Human Readable: ISO 639 digraph language codes  
must be in the ISO 639 digraph CVE. '"/>  
</sch:pattern>
```

Rule: IRM-ID-00006

FileName:./Rules/valueEnumerationConstraints/IRM_ID_00006.sch

Rule Description:

[IRM-ID-00006][Error] If element ddms:language has the attribute ddms:qualifier value of [urn:us:gov:ic:cvenum:irm:iso639-2:trigraph:v1] then the value of attribute ddms:value must be in CVENumIRMISO639-2Trigraph.xml and no country code portion may be specified in the ddms:value value. Human Readable: ISO 639-2 trigraph language codes must be in the ISO 639-2 1 trigraph CVE.

Code Description:

This rule uses an abstract pattern to consolidate logic. It checks that the value in parameter \$searchTerm is contained in the parameter \$list. The parameter \$searchTerm is relative in scope to the parameter \$context. The value for the parameter \$list is a variable defined in the main document (IRM_XML.sch), which reads values from a specific CVE file. This rule can directly check if the value of element Language is in the appropriate list because if a country code portion is specified in the element ddms:language's value, then the value of element ddms:language will not be found in the appropriate list and the assertion will fail as expected.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="IRM-ID-00006"
is-a="ValidateValueExistenceInList">

<sch:param name="ruleText"
value="' [IRM-ID-00006][Error] If element ddms:language has the attribute ddms:qualifier
value of [urn:us:gov:ic:cvenum:irm:iso639-2:trigraph:v1] then the value of attribute
ddms:value must be in CVENumIRMISO639-2Trigraph.xml and no country code portion may be
specified in the ddms:value value. Human Readable: ISO 639-2 trigraph language codes must
be in the ISO 639-2 1 trigraph CVE. '"/>

<sch:param name="codeDesc"
value="' This rule uses an abstract pattern to consolidate logic. It checks that the value
in parameter $searchTerm is contained in the parameter $list. The parameter $searchTerm
is relative in scope to the parameter $context. The value for the parameter $list is a
variable defined in the main document (IRM_XML.sch), which reads values from a specific
CVE file. '"/>

<sch:param name="context"
value="ddms:language[@ddms:qualifier='urn:us:gov:ic:cvenum:irm:iso639-2:trigraph:v1']"/>
<sch:param name="searchTerm" value="@ddms:value"/>
<sch:param name="list" value="$iso639-2TrigraphList"/>
<sch:param name="errMsg"
value="' [IRM-ID-00006][Error] If element ddms:language has the attribute ddms:qualifier
value of [urn:us:gov:ic:cvenum:irm:iso639-2:trigraph:v1] then the value of attribute
ddms:value must be in CVENumIRMISO639-2Trigraph.xml and no country code portion may be
```

```
specified in the ddms:value value. Human Readable: ISO 639-2 trigraph language codes must  
be in the ISO 639-2 1 trigraph CVE. '"/>  
</sch:pattern>
```

Rule: IRM-ID-00007

FileName:./Rules/valueEnumerationConstraints/IRM_ID_00007.sch

Rule Description:

[IRM-ID-00007][Error] If element ddms:language has the attribute ddms:qualifier value of [urn:us:gov:ic:cvenum:irm:iso639-3:trigraph:v1] then the value of attribute ddms:value must be in CVENumIRMISO639-3Trigraph.xml and no country code portion may be specified in the ddms:value value. Human Readable: ISO 639-3 trigraph language codes must be in the ISO 639-3 trigraph CVE.

Code Description:

This rule uses an abstract pattern to consolidate logic. It checks that the value in parameter \$searchTerm is contained in the parameter \$list. The parameter \$searchTerm is relative in scope to the parameter \$context. The value for the parameter \$list is a variable defined in the main document (IRM_XML.sch), which reads values from a specific CVE file. This rule can directly check if the value of element Language is in the appropriate list because if a country code portion is specified in the element ddms:language's value, then the value of element ddms:language will not be found in the appropriate list and the assertion will fail as expected.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="IRM-ID-00007"
is-a="ValidateValueExistenceInList">

<sch:param name="ruleText"
value="' [IRM-ID-00007][Error] If element ddms:language has the attribute ddms:qualifier
value of [urn:us:gov:ic:cvenum:irm:iso639-3:trigraph:v1] then the value of attribute
ddms:value must be in CVENumIRMISO639-3Trigraph.xml and no country code portion may be
specified in the ddms:value value. Human Readable: ISO 639-3 trigraph language codes must
in the the ISO 639-3 trigraph CVE. '"/>

<sch:param name="codeDesc"
value="' This rule uses an abstract pattern to consolidate logic. It checks that the value
in parameter $searchTerm is contained in the parameter $list. The parameter $searchTerm
is relative in scope to the parameter $context. The value for the parameter $list is a
variable defined in the main document (IRM_XML.sch), which reads values from a specific
CVE file. '"/>

<sch:param name="context"
value="irm:DescribedItemIRM//
ddms:language[@ddms:qualifier='urn:us:gov:ic:cvenum:irm:iso639-3:trigraph:v1']"/>
<sch:param name="searchTerm" value="@ddms:value"/>
<sch:param name="list" value="$iso639-3TrigraphList"/>
<sch:param name="errMsg"
value="' [IRM-ID-00007][Error] If element ddms:language has the attribute ddms:qualifier
value of [urn:us:gov:ic:cvenum:irm:iso639-3:trigraph:v1] then the value of attribute
ddms:value must be in CVENumIRMISO639-3Trigraph.xml and no country code portion may be
```



```
specified in the ddms:value value. Human Readable: ISO 639-3 trigraph language codes must  
in the the ISO 639-3 trigraph CVE. '"/>  
</sch:pattern>
```

Rule: IRM-ID-00008

FileName:./Rules/valueEnumerationConstraints/IRM_ID_00008.sch

Rule Description:

[IRM-ID-00008][Error] If element ddms:language has the attribute ddms:qualifier value of [RFC1766] then the language code portion of the ddms:value attribute value must be in CVENumIRMISO639Digraph.xml and the country code portion, if present, must be in CVENumIRMCoverageISO3166Digraph.xml. Human Readable: RFC1766 language codes must comply with the RFC by using parts from ISO 639 Digraph and ISO 3166 Digraph.

Code Description:

Finds ddms:language element and checks its qualifier attribute for a value of [RFC4646]. If this value is found it will ensure that the value of the element's ddms:value attribute exists in the CVENumIRMISO639Digraph.xml enumeration file represented by the \$iso639DigraphList variable and country code portions (denoted by '-' separation) must be in CVENumIRMCoverageISO3166Digraph.xml enumeration file represented by the \$coverageIso3166DigraphList variable.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="IRM-ID-00008">

  <sch:rule context="irm:DescribedItemIRM//ddms:language[@ddms:qualifier='RFC1766']">
    <!-- Tokenize the element Language value into a list -->
    <sch:let name="tokens" value="tokenize(@ddms:value, '-')"/>

    <!-- For convenience and readability, save the primary and secondary subtags
    as defined in RFC 4646 -->
    <sch:let name="primarySubtag" value="$tokens[1]"/>
    <sch:let name="secondarySubtag" value="$tokens[2]"/>

    <sch:let name="badPrimaryValues"
    value=" if(index-of($iso639DigraphList,$primarySubtag)>0) then null else
    $primarySubtag"/>

    <sch:let name="badSecondaryValues"
    value=" if($secondarySubtag and index-of($coverageIso3166DigraphList,
    $secondarySubtag)>0) then null else $secondarySubtag"/>

    <sch:let name="badValues"
    value="string-join(($badPrimaryValues, $badSecondaryValues), ' ')" />

    <!-- Check if primary subtag is valid -->
    <sch:let name="primarySubtagValid"
    value=" count($badPrimaryValues) = 0 " />

    <!-- Check if secondary subtag is valid -->
    <sch:let name="secondarySubtagValid"
```

```

value=" if(not($secondarySubtag)) then true() else count($badSecondaryValues) = 0 "/>
<sch:assert id="IRM-00008" test="$primarySubtagValid and $secondarySubtagValid"
flag="error">
[IRM-ID-00008][Error] If element ddms:language has the attribute ddms:qualifier
value of [RFC1766] then the language code portion of the ddms:value attribute
value must be in CVEnumIRMISO639Digraph.xml and the country code portion, if
present, must be in CVEnumIRMCoverageISO3166Digraph.xml.

Human Readable: RFC1766 language codes must comply with the RFC by using parts from ISO
639 Digraph
and ISO 3166 Digraph. The following values were found but are not in the CVEs:
<sch:value-of select="for $each in tokenize($badValues, ' ') return concat('[',$each,']
')"/>
</sch:assert>
</sch:rule>
</sch:pattern>

```

Rule: IRM-ID-00009

FileName:../Rules/valueEnumerationConstraints/IRM_ID_00009.sch

Rule Description:

[IRM-ID-00009][Error] If element ddms:language has the attribute ddms:qualifier value of [RFC3066] then the language code portion of the ddms:value attribute value must be in CVEnumIRMISO639Digraph.xml or CVEnumIRMISO639-2Trigraph.xml and the country code portion, if present, must be in CVEnumIRMCoverageISO3166Digraph.xml. Human Readable: RFC3066 language codes must comply with the RFC by using parts from ISO 639 Digraph, 639-2 Trigraph, and ISO 3166 Digraph.

Code Description:

Finds ddms:language elements and checks its qualifier attribute for a value of [RFC3066]. If this value is found it will ensure that the value of the element's ddms:value attribute exists in the CVEnumIRMISO639Digraph.xml or CVEnumIRMISO639-2Trigraph.xml enumeration files represented by the \$iso639DigraphList or \$iso639-2TrigraphList variables. Country code portions (denoted by '-' separation) must be in the CVEnumIRMCoverageISO3166Trigraph.xml enumeration file represented by the \$coverageIso3166TrigraphList variable.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="IRM-ID-00009">

<sch:rule context="irm:DescribedItemIRM//ddms:language[@ddms:qualifier='RFC3066']">
<!-- Tokenize the element Language value into a list -->
<sch:let name="tokens" value="tokenize(@ddms:value, '-')"/>

<!-- For convenience and readability, save the primary and secondary subtags
as defined in RFC 3066 -->
<sch:let name="primarySubtag" value="$tokens[1]"/>
<sch:let name="secondarySubtag" value="$tokens[2]"/>

<sch:let name="badPrimaryValues"
value=" if(index-of($iso639-2TrigraphList,$primarySubtag)>0 or index-
of($iso639DigraphList,$primarySubtag)>0) then null else $primarySubtag"/>

<sch:let name="badSecondaryValues"
value=" if($secondarySubtag and index-of($coverageIso3166DigraphList,
$secondarySubtag)>0) then null else $secondarySubtag"/>

<sch:let name="badValues"
value="string-join(($badPrimaryValues, $badSecondaryValues), ' ')" />

<!-- Check if primary subtag is valid -->
<sch:let name="primarySubtagValid"
value=" count($badPrimaryValues) = 0 " />
```

```

<!-- Check if secondary subtag is valid -->
<sch:let name="secondarySubtagValid"
value=" if(not($secondarySubtag)) then true() else count($badSecondaryValues) = 0 "/>

<sch:assert id="IRM-00009" test="$primarySubtagValid and $secondarySubtagValid"
flag="error">
[IRM-ID-00009][Error] If element ddms:language has the attribute ddms:qualifier
value of [RFC3066] then the language code portion of the ddms:value attribute
value must be in CVEnumIRMISO639Digraph.xml or CVEnumIRMISO639-2Trigraph.xml
and the country code portion, if present, must be in CVEnumIRMCoverageISO3166Digraph.xml.

Human Readable: RFC3066 language codes must comply with the RFC by using parts from
ISO 639 Digraph or ISO 639-2 Trigraph and ISO 3166 Digraph. The following values were
found but
are not in the CVEs:
<sch:value-of select="for $each in tokenize($badValues, ' ') return concat('[',$each,']
')"/>
</sch:assert>
</sch:rule>
</sch:pattern>

```

Rule: IRM-ID-00010

FileName:./Rules/valueEnumerationConstraints/IRM_ID_00010.sch

Rule Description:

[IRM-ID-00010][Error] If element ddms:language has the attribute ddms:qualifier value of [RFC4646] then the language code portion of the ddms:value attribute value must be in CVENumIRMISO639Digraph.xml or CVENumIRMISO639-2Trigraph.xml and the country code portion, if present, must be in CVENumIRMCoverageISO3166Digraph.xml. Human Readable: RFC4646 language codes must comply with the RFC by using parts from ISO 639 Digraph or ISO 639-2 Trigraph and ISO 3166 Digraph.

Code Description:

Finds ddms:language elements and checks its qualifier attribute for a value of [RFC4646]. If this value is found it will ensure that the value of the element's ddms:value attribute exists in the CVENumIRMISO639Digraph.xml or CVENumIRMISO639-2Trigraph.xml enumeration files represented by the \$iso639DigraphList or \$iso639-2TrigraphList variables. Country code portions (denoted by '-' separation) must be in the CVENumIRMCoverageISO3166Digraph.xml enumeration file represented by the \$coverageIso3166TrigraphList variable.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="IRM-ID-00010">

  <sch:rule context="irm:DescribedItemIRM//ddms:language[@ddms:qualifier='RFC4646']">
    <!-- Tokenize the element Language value into a list -->
    <sch:let name="tokens" value="tokenize(@ddms:value, '-')"/>

    <!-- For convenience and readability, save the primary and secondary subtags
    as defined in RFC 4646 -->
    <sch:let name="primarySubtag" value="$tokens[1]"/>
    <sch:let name="secondarySubtag" value="$tokens[2]"/>

    <sch:let name="badPrimaryValues"
    value=" if(index-of($iso639-2TrigraphList,lower-case($primarySubtag))>0 or index-
    of($iso639DigraphList,$primarySubtag)>0) then null else $primarySubtag"/>

    <sch:let name="badSecondaryValues"
    value=" if($secondarySubtag and index-of($coverageIso3166DigraphList,
    $secondarySubtag)>0) then null else $secondarySubtag"/>

    <sch:let name="badValues"
    value="string-join(($badPrimaryValues, $badSecondaryValues), ' ')" />

    <!-- Check if primary subtag is valid -->
    <sch:let name="primarySubtagValid"
    value=" count($badPrimaryValues) = 0 " />
```

```

<!-- Check if secondary subtag is valid -->
<sch:let name="secondarySubtagValid"
value=" if(not($secondarySubtag)) then true() else count($badSecondaryValues) = 0 "/>

<sch:assert id="IRM-00010" test="$primarySubtagValid and $secondarySubtagValid"
flag="error">
[IRM-ID-00010][Error] If element ddms:language has the attribute ddms:qualifier
value of [RFC4646] then the language code portion of the ddms:value attribute
value must be in CVENumIRMISO639Digraph.xml or CVENumIRMISO639-2Trigraph.xml
and the country code portion, if present, must be in CVENumIRMCoverageISO3166Digraph.xml.

Human Readable: RFC4646 language codes must comply with the RFC by using parts
from ISO 639 Digraph or ISO 639-2 Trigraph and ISO 3166 Digraph. The following
values were found but are not in the CVEs:
<sch:value-of select="for $each in tokenize($badValues, ' ') return concat('[',$each,']
' )"/>
</sch:assert>
</sch:rule>
</sch:pattern>

```

Rule: IRM-ID-00011

FileName:./Rules/generalConstraints/IRM_ID_00011.sch

Rule Description:

[IRM-ID-00011][Error] If an element `irm:MetaCardIRM/irm:IdentifierList/ddms:identifier` does not exist with the attribute `ddms: qualifier` value of [IC-ID]. Human Readable: Every metacard must have an ID identified as an [IC-ID].

Code Description:

This rule checks that for each qualifier attribute in path `//irm:MetaCardIRM/irm:IdentifierList/ddms:identifier/@ddms:qualifier`, that its normalized value with leading- and trailing-spaces removed is equal to 'IC-ID'. If there is at least one attribute with the given qualifier attribute, the rule is satisfied.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="IRM-ID-00011">

<sch:rule context="irm:MetaCardIRM/irm:IdentifierList">
<sch:assert id="IRM-00011"
test=" some $qualifier in ddms:identifier/@ddms:qualifier satisfies normalize-
space(string($qualifier)) = 'IC-ID' "
flag="error">
[IRM-ID-00011][Error] Every metacard must have an ID identified as an [IC-ID].
</sch:assert>
</sch:rule>
</sch:pattern>
```


Rule: IRM-ID-00012

FileName:../Rules/generalConstraints/IRM_ID_00012.sch

Rule Description:

[IRM-ID-00012][Error] More than 1 element irm:MetaCardIRM/irm:IdentifierList/ddms:identifier exists with the attribute ddms:qualifier having a value of [IC-ID]. Human Readable: Every metacard must have 1 and only 1 ID identified as an [IC-ID].

Code Description:

This rule checks that for each qualifier attribute in path //irm:MetaCardIRM/irm:IdentifierList/ddms:identifier/@ddms:qualifier, that its normalized value with leading- and trailing-spaces removed is equal to 'IC-ID'. If there is exactly one attribute with the given qualifier attribute, the rule is satisfied.

Schematron Code:

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

<sch:rule context="irm:MetaCardIRM/irm:IdentifierList">
<sch:assert id="IRM-00012"
test=" count( for $qualifier in ddms:identifier/@ddms:qualifier return if(normalize-
space(string($qualifier)) = 'IC-ID') then 1 else null ) = 1 "
flag="error">
[IRM-ID-00012][Error] Every metacard must have 1 and only 1 ID identified as an [IC-ID].
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: IRM-ID-00013

FileName:../Rules/generalConstraints/IRM_ID_00013.sch

Rule Description:

[IRM-ID-00013][Error] If an element `irm:DescribedItemIRM/ddms:Resource/ddms:identifier` does not exist with the attribute `ddms:qualifier` value of [IC-ID]. Human Readable: Every `DescribedItemIRM` must have an ID identified as an [IC-ID].

Code Description:

This rule checks that for each qualifier attribute in path `//irm:DescribedItemIRM/ddms:Resource/ddms:identifier/@ddms:qualifier`, that its normalized value with leading- and trailing-spaces removed is equal to 'IC-ID'. If there is at least one attribute with the given qualifier attribute, the rule is satisfied.

Schematron Code:

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

<sch:rule context="irm:DescribedItemIRM/ddms:Resource">
<sch:assert id="IRM-00013"
test=" some $qualifier in ddms:identifier/@ddms:qualifier satisfies normalize-
space(string($qualifier)) = 'IC-ID' "
flag="error">
[IRM-ID-00013][Error] Every DescribedItemIRM must have an ID identified as an [IC-ID].
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: IRM-ID-00014

FileName:../Rules/generalConstraints/IRM_ID_00014.sch

Rule Description:

[IRM-ID-00014][Error] If more than 1 element `irm:DescribedItemIRM/ddms:Resource/ddms:identifier` exists with the attribute `ddms:qualifier` value of [IC-ID]. Human Readable: Every DescribedItemIRM must have 1 and only 1 ID identified as an [IC-ID].

Code Description:

This rule checks that for each qualifier attribute in path `//irm:DescribedItemIRM/ddms:Resource/ddms:identifier/@ddms:qualifier`, that its normalized value with leading- and trailing-spaces removed is equal to 'IC-ID'. If there is exactly one attribute with the given qualifier attribute, the rule is satisfied.

Schematron Code:

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

<sch:rule context="irm:DescribedItemIRM/ddms:Resource">
<sch:assert id="IRM-00014"
test=" count( for $qualifier in ddms:identifier/@ddms:qualifier return if (normalize-
space(string($qualifier)) = 'IC-ID') then 1 else null ) = 1 "
flag="error">
[IRM-ID-00014][Error] Every DescribedItemIRM must have 1 and only 1 ID identified as an
[IC-ID].
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: IRM-ID-00015

FileName:../Rules/generalConstraints/IRM_ID_00015.sch

Rule Description:

[IRM-ID-00015][Error] If element ddms:dates exists without one of the attributes ddms:created or ddms:posted Human Readable: Every ddms:dates element must have at least one of ddms:created or ddms:posted.

Code Description:

This rule checks that for each occurrence of ddms:dates that either ddms:created or ddms:posted is specified.

Schematron Code:

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

<sch:rule context="ddms:dates">
<sch:assert id="IRM-00015" test="@ddms:created or @ddms:posted" flag="error">
[IRM-ID-00015][Error] Every ddms:date must have at least one of ddms:created or
ddms:posted.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: IRM-ID-00016

FileName:./Rules/generalConstraints/IRM_ID_00016.sch

Rule Description:

[IRM-ID-00016][Error] The permissible values for the year range are 1901 through the current year for attributes irm:dateApproved, irm:dateProcessed, irm:dateReceived, ddms:infoCutOff, ddms:posted, and ddms:created. Human Readable: Dates must be after 1901 and in the past for irm:dateApproved, irm:dateProcessed, irm:dateReceived, ddms:infoCutOff, ddms:posted, and ddms:created.

Code Description:

This pattern uses abstract rules to consolidate logic. For attributes, we make sure that each date contained within \$dateList has a year value within the range \$minYear and \$maxYear, inclusive.

Schematron Code:

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

  <!-- Use abstract rule to handle required attributes -->
  <sch:rule context="*[@irm:dateApproved | @irm:dateProcessed | @irm:dateReceived |
  @ddms:posted | @ddms:created | @ddms:infoCutOff]">
    <sch:let name="minYear" value="1901"/>
    <sch:let name="maxYear" value="$currentYear"/>
    <sch:let name="dateList"
    value=" (string(@irm:dateApproved), string(@irm:dateProcessed), string(@irm:dateReceived),
    string(@ddms:posted), string(@ddms:created), string(@ddms:infoCutOff))"/>
    <sch:let name="errMsg"
    value="' [IRM-ID-00016][Error] The permissible values for the year range are 1901 through
    the current year for attributes irm:dateApproved, irm:dateProcessed, irm:dateReceived,
    ddms:infoCutOff, ddms:posted, and ddms:created. Human Readable: Dates must be after
    1901 and in the past for irm:dateApproved, irm:dateProcessed, irm:dateReceived,
    ddms:infoCutOff, ddms:posted, and ddms:created. '"/>
    <sch:extends rule="abs.dateListYearRangeRule"/>
  </sch:rule>
</sch:pattern>
```

Rule: IRM-ID-00017

FileName:./Rules/generalConstraints/IRM_ID_00017.sch

Rule Description:

[IRM-ID-00017][Error] The permissible values for the year range are 1901 through 9999 for element ddms:validTil. Human Readable: ddms:validTil must be after 1901.

Code Description:

This pattern uses abstract rules to consolidate logic. For attributes, we make sure that each date contained within \$dateList has a year value within the range \$minYear and \$maxYear, inclusive.

Schematron Code:

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

  <!-- Use abstract rule to handle required attributes -->
  <sch:rule context="*[@ddms:validTil]">
    <sch:let name="minYear" value="1901"/>
    <sch:let name="maxYear" value="9999"/>
    <sch:let name="dateList" value="(string(@ddms:validTil))"/>
    <sch:let name="errMsg"
value="" [IRM-ID-00017][Error] The permissible values for the year range are 1901 through
9999 for element ddms:validTil. Human Readable: ddms:validTil must be after 1901. ""/>
    <sch:extends rule="abs.dateListYearRangeRule"/>
  </sch:rule>
</sch:pattern>
```

Rule: IRM-ID-00018

FileName:./Rules/generalConstraints/IRM_ID_00018.sch

Rule Description:

[IRM-ID-00018][Error] The permissible values for the decimal seconds are .0 through .999 for elements ddms:start and ddms:end and attributes irm:dateApproved, irm:dateProcessed, irm:dateReceived, ddms:infoCutOff, ddms:posted, ddms:validTil, and ddms:created. Human Readable: For date-time types, decimal second values can be specified up to three decimal places.

Code Description:

This pattern is enforced by two rules, one to handle attributes and the other to handle elements. For attributes, we match any element which has any of the attributes specified and join all of the attribute values into a string. Then, we tokenize that string and for each date we make sure that if the value specified contains a period, then it matches the regular expression for a dateTime containing 1,2, or 3 digits for the millisecond portion. For elements, we make sure that if the value specified contains a period, then it matches the regular expression for a dateTime containing 1,2, or 3 digits for the millisecond portion. Note that elements ddms:start and ddms:end allow the strings 'Unknown' and 'Not Applicable' as valid values. This rule correctly handles these string values by returning true because they do not contain a period '.' character.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="IRM-ID-00018">

  <sch:let name="decimalSecondsRegEx" value="concat('^.*\.\d{1,3}(', $timeZoneRegEx, ')$')"/>

  <!-- Rule to handle attributes -->
  <sch:rule context=" *[@irm:dateApproved | @irm:dateProcessed | @irm:dateReceived |
  @ddms:infoCutOff | @ddms:posted | @ddms:validTil | @ddms:created]">
    <sch:let name="dateList"
    value=" string-join(( normalize-space(string(@irm:dateApproved)), normalize-
    space(string(@irm:dateProcessed)), normalize-space(string(@irm:dateReceived)), normalize-
    space(string(@ddms:infoCutOff)), normalize-space(string(@ddms:posted)), normalize-
    space(string(@ddms:validTil)), normalize-space(string(@ddms:created)) ), ' ')" />
    <sch:assert test=" every $date in tokenize($dateList, ' ') satisfies
    if(contains(string($date), '.')) then matches(normalize-space(string($date)),
    $decimalSecondsRegEx) else true() "
    flag="error">
    [IRM-ID-00018][Error] The permissible values for the decimal seconds are
    .0 through .999 for elements ddms:start and ddms:end and attributes irm:dateApproved,
    irm:dateProcessed, irm:dateReceived, ddms:infoCutOff, ddms:posted, ddms:validTil,
    and ddms:created.
  </sch:assert>
  </sch:rule>

  <!-- Rule to handle elements -->
  <sch:rule context="ddms:start | ddms:end">
```

```
<sch:assert id="PUBS-00043"
test=" if(contains(string(.),'.')) then matches(normalize-space(string(.)),
$decimalSecondsRegEx) else true()"
flag="error">
[IRM-ID-00018][Error] The permissible values for the decimal seconds are
.0 through .999 for elements ddms:start and ddms:end and attributes irm:dateApproved,
irm:dateProcessed, irm:dateReceived, ddms:infoCutOff, ddms:posted, ddms:validTil,
and ddms:created.
</sch:assert>
</sch:rule>
</sch:pattern>
```


Rule: IRM-ID-00019

FileName:./Rules/generalConstraints/IRM_ID_00019.sch

Rule Description:

[IRM-ID-00019][Warning] irm:dateApproved must not be later than ddms:created, and ddms:posted.

Code Description:

This rule uses an abstract pattern to consolidate logic. It compares the date contained within the param \$primaryDate to each date contained within the param \$secondaryDateList (using the comparison operator contained in param \$operator) and makes sure that each comparison returns true. Implementation details for the abstract pattern can be found in the abstract pattern definition file located in the Lib directory.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="IRM-ID-00019"
is-a="CompareDateTimes">

<sch:param name="ruleText"
value="' [IRM-ID-00019][Warning] irm:dateApproved must not be later than ddms:created, and
ddms:posted. '"/>

<sch:param name="codeDesc"
value="' This rule uses an abstract pattern to consolidate logic. It compares the
date contained within the param $primaryDate to each date contained within the param
$secondaryDateList (using the comparison operator contained in param $operator) and makes
sure that each comparison returns true. Implementation details for the abstract pattern
can be found in the abstract pattern definition file located in the Lib directory. '"/>

<sch:param name="context" value="*[@irm:dateApproved]"/>
<sch:param name="primaryDate" value="@irm:dateApproved"/>
<sch:param name="operator" value="'&lt;='"/>
<sch:param name="secondaryDateList"
value="(..//*[@ddms:created]/@ddms:created, ..//*[@ddms:posted]/@ddms:posted)"/>
<sch:param name="flag" value="'error'"/>
</sch:pattern>
```

Rule: IRM-ID-00020

FileName:./Rules/generalConstraints/IRM_ID_00020.sch

Rule Description:

[IRM-ID-00020][Error] ddms:infoCutOff must not be later than ddms:created, and ddms:posted.

Code Description:

This rule uses an abstract pattern to consolidate logic. It compares the date contained within the param \$primaryDate to each date contained within the param \$secondaryDateList (using the comparison operator contained in param \$operator) and makes sure that each comparison returns true. Implementation details for the abstract pattern can be found in the abstract pattern definition file located in the Lib directory.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="IRM-ID-00020"
is-a="CompareDateTimes">

<sch:param name="ruleText"
value="' [IRM-ID-00020][Error] ddms:infoCutOff must not be later than ddms:created, and
ddms:posted. '"/>

<sch:param name="codeDesc"
value="' This rule uses an abstract pattern to consolidate logic. It compares the
date contained within the param $primaryDate to each date contained within the param
$secondaryDateList (using the comparison operator contained in param $operator) and makes
sure that each comparison returns true. Implementation details for the abstract pattern
can be found in the abstract pattern definition file located in the Lib directory. '"/>

<sch:param name="context" value="*[@ddms:infoCutOff]"/>
<sch:param name="primaryDate" value="@ddms:infoCutOff"/>
<sch:param name="operator" value="'&lt;='"/>
<sch:param name="secondaryDateList"
value="(..//*[@ddms:created]/@ddms:created, ..//*[@ddms:posted]/@ddms:posted)"/>
<sch:param name="flag" value="'error'"/>
</sch:pattern>
```

Rule: IRM-ID-00021

FileName:../Rules/generalConstraints/IRM_ID_00021.sch

Rule Description:

[IRM-ID-00021][Warning] ddms:validTil must not be earlier than ddms:created, ddms:posted, ddms:infoCutOff, and irm:dateApproved.

Code Description:

This rule uses an abstract pattern to consolidate logic. It compares the date contained within the param \$primaryDate to each date contained within the param \$secondaryDateList (using the comparison operator contained in param \$operator) and makes sure that each comparison returns true. Implementation details for the abstract pattern can be found in the abstract pattern definition file located in the Lib directory.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="IRM-ID-00021"
is-a="CompareDateTimes">

<sch:param name="ruleText"
value="' [IRM-ID-00021][Warning] ddms:validTil must not be earlier than ddms:created,
ddms:posted, ddms:infoCutOff, and irm:dateApproved. '"/>

<sch:param name="codeDesc"
value="' This rule uses an abstract pattern to consolidate logic. It compares the
date contained within the param $primaryDate to each date contained within the param
$secondaryDateList (using the comparison operator contained in param $operator) and makes
sure that each comparison returns true. Implementation details for the abstract pattern
can be found in the abstract pattern definition file located in the Lib directory. '"/>

<sch:param name="context" value="*[@ddms:validTil]"/>
<sch:param name="primaryDate" value="@ddms:validTil"/>
<sch:param name="operator" value="'&gt;='"/>
<sch:param name="secondaryDateList"
value="('../*[@ddms:created]/@ddms:created, ../*[@ddms:posted]/@ddms:posted, ../
*[@ddms:infoCutOff]/@ddms:infoCutOff, ../*[@irm:dateApproved]/@irm:dateApproved)"/>
<sch:param name="flag" value="'error'"/>
</sch:pattern>
```

Rule: IRM-ID-00022

FileName:../Rules/generalConstraints/IRM_ID_00022.sch

Rule Description:

[IRM-ID-00022][Error] For any element TimePeriod, child element start must not be later than child element end. Human Readable: For date-time ranges, the start of a range must be earlier than, or equivalent to, the end of that range.

Code Description:

This rule uses an abstract pattern to consolidate logic. It compares the date contained within the param \$primaryDate to each date contained within the param \$secondaryDateList (using the comparison operator contained in param \$operator) and makes sure that each comparison returns true. Implementation details for the abstract pattern can be found in the abstract pattern definition file located in the Lib directory.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="IRM-ID-00022"
is-a="CompareDateTimes">

<sch:param name="ruleText"
value="' [IRM-ID-00022][Error] For any element TimePeriod, child element start must not be
later than child element end. '"/>

<sch:param name="codeDesc"
value="' This rule uses an abstract pattern to consolidate logic. It compares the
date contained within the param $primaryDate to each date contained within the param
$secondaryDateList (using the comparison operator contained in param $operator) and makes
sure that each comparison returns true. Implementation details for the abstract pattern
can be found in the abstract pattern definition file located in the Lib directory. '"/>

<sch:param name="context" value="ddms:TimePeriod"/>
<sch:param name="primaryDate" value="ddms:start"/>
<sch:param name="operator" value="'&lt;='"/>
<sch:param name="secondaryDateList" value="(ddms:end)"/>
<sch:param name="flag" value="'error'"/>
</sch:pattern>
```

Rule: IRM-ID-00023

FileName:./Rules/generalConstraints/IRM_ID_00023.sch

Rule Description:

[IRM-ID-00023][Error] The permissible values for the year range are 0001 through 9999 for elements ddms:start and ddms:end. Human Readable: ddms:start and ddms:end must be positive integers less than 10,000.

Code Description:

This pattern uses abstract rules to consolidate logic. For attributes, we make sure that each date contained within \$dateList has a year value within the range \$minYear and \$maxYear, inclusive.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="IRM-ID-00023">

  <!-- Use abstract rule to handle required attributes -->
  <sch:rule context="ddms:start | ddms:end">
    <sch:let name="minYear" value="0001"/>
    <sch:let name="maxYear" value="9999"/>
    <sch:let name="dateList" value="(string(ddms:start), string(ddms:end))"/>
    <sch:let name="errMsg"
value="" [IRM-ID-00023][Error] The permissible values for the year range are 0001 through
9999 for elements ddms:start and ddms:end. Human Readable: ddms:start and ddms:end must be
positive integers less than 10,000. ""/>
    <sch:extends rule="abs.dateListYearRangeRule"/>
  </sch:rule>
</sch:pattern>
```

Rule: IRM-ID-00024

FileName:./Rules/generalConstraints/IRM_ID_00024.sch

Rule Description:

[IRM-ID-00024][Warning] For elements ddms:start and ddms:end and attributes irm:dateApproved, irm:dateProcessed, irm:dateReceived, ddms:infoCutOff, ddms:posted, ddms:validTil, and ddms:created, if the time designator (T) is specified, it is recommended that time zone be specified. Human Readable: For elements and attributes of date-time types, if the time designator (T) is specified, it is recommended that time zone be specified.

Code Description:

The pattern applies to ddms:start and ddms:end elements, as well as any element that contains one or more attributes irm:dateApproved, ddms:infoCutOff, ddms:posted, and ddms:created. It joins each of these attribute values, if present, into a larger space-separated string. It then breaks this string into tokens at each space character. If the value of the token contains the time zone designator (T), then it makes sure that the token value matches the regular expression for a timeZone, which is defined in the main schema file (IRM_XML.sch).

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="IRM-ID-00024">

  <!-- Abstract rule, which asserts that if the date $primaryDate specifies the
  time designator (T), then the timezone is specified -->
  <sch:rule abstract="true" id="abs.rule00024">
    <sch:assert id="IRM-00024"
    test=" every $date in $dateList satisfies if($date castable as xs:dateTime and
    contains(string($date), 'T')) then matches(string($date), $endsWithTimeZoneRegEx) else
    true() "
    flag="warning">
  </sch:assert>
</sch:rule>

  <!-- Begin using abstract rule on required elements and attributes -->
  <sch:rule context="ddms:start">
    <sch:let name="dateList" value="."/>
    <sch:extends rule="abs.rule00024"/>
  </sch:rule>

  <sch:rule context="ddms:end">
    <sch:let name="dateList" value="."/>
    <sch:extends rule="abs.rule00024"/>
  </sch:rule>

  <sch:rule context="*[@irm:dateApproved | @irm:dateProcessed | @irm:dateReceived |
  @ddms:posted | @ddms:created | @ddms:infoCutOff | @ddms:validTil]">
    <sch:let name="dateList"
```

```
value=" (@irm:dateApproved, @irm:dateProcessed, @irm:dateReceived, @ddms:posted,  
@ddms:created, @ddms:infoCutOff, @ddms:validTil) "/>  
<sch:extends rule="abs.rule00024"/>  
</sch:rule>  
</sch:pattern>
```

Rule: IRM-ID-00025

FileName:../Rules/ismConstraints/IRM_ID_00025.sch

Rule Description:

[IRM-ID-00025][Error] The attribute ism:excludeFromRollup must not be specified for any element in the namespace http://metadata.dod.mil/mdr/ns/DDMS/3.0/ except security. Human Readable: The only DDMS 3.0 element whose ISM attributes are allowed to be excluded from roll-up to the Resource element is ddms:security.

Code Description:

For any element in the ddms namespace containing the ism:excludeFromRollup attribute, the rule will be satisfied if the name of the element is security.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="IRM-ID-00025">

<sch:rule context="*[@ism:excludeFromRollup and namespace-uri()='http://metadata.dod.mil/
mdr/ns/DDMS/3.0/']">
<sch:assert id="IRM-00025" test="local-name() = 'security'" flag="error">
[IRM-ID-00025][Error] The attribute ism:excludeFromRollup must not be specified for any
element
in the namespace http://metadata.dod.mil/mdr/ns/DDMS/3.0/ except security.
</sch:assert>
</sch:rule>
</sch:pattern>
```


Rule: IRM-ID-00027

FileName:./Rules/valueEnumerationConstraints/IRM_ID_00027.sch

Rule Description:

[IRM-ID-00027][Error] If element CountryCode has attribute vocabulary specified as FIPS the element value must be in CVEnumIRMCoverageFIPSDigraph.xml. Human Readable: FIPS country codes must be in the FIPS CVE.

Code Description:

This rule uses an abstract pattern to consolidate logic. It checks that the value in parameter \$searchTerm is contained in the parameter \$list. The parameter \$searchTerm is relative in scope to the parameter \$context. The value for the parameter \$list is a variable defined in the main document (IRM_XML.sch), which reads values from a specific CVE file.

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron"
is-a="ValidateValueExistenceInList"
id="IRM-ID-00027">

<sch:param name="ruleText"
value="' [IRM-ID-00027][Error] If element CountryCode has attribute vocabulary specified
as FIPS the element value must be in CVEnumIRMCoverageFIPSDigraph.xml. '"/>

<sch:param name="codeDesc"
value="' This rule uses an abstract pattern to consolidate logic. It checks that the value
in parameter $searchTerm is contained in the parameter $list. The parameter $searchTerm
is relative in scope to the parameter $context. The value for the parameter $list is a
variable defined in the main document (IRM_XML.sch), which reads values from a specific
CVE file. '"/>

<sch:param name="context" value="//irm:CountryCode[@irm:vocabulary='FIPS']"/>
<sch:param name="searchTerm" value="."/>
<sch:param name="list" value="$coverageFipsDigraphList"/>
<sch:param name="errMsg"
value="' [IRM-ID-00027][Error] If element CountryCode has attribute vocabulary specified
as FIPS the element value must be in CVEnumIRMCoverageFIPSDigraph.xml. '"/>
</sch:pattern>
```

Rule: IRM-ID-00028

FileName:./Rules/valueEnumerationConstraints/IRM_ID_00028.sch

Rule Description:

[IRM-ID-00028][Error] If element CountryCode has attribute vocabulary specified as ISO-3 the element value must be in CVEnumIRMCoverageISO3166Trigraph.xml. Human Readable: ISO 3166 trigraph country codes must be in the ISO 3166 trigraph CVE.

Code Description:

This rule uses an abstract pattern to consolidate logic. It checks that the value in parameter \$searchTerm is contained in the parameter \$list. The parameter \$searchTerm is relative in scope to the parameter \$context. The value for the parameter \$list is a variable defined in the main document (IRM_XML.sch), which reads values from a specific CVE file.

Schematron Code:

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

<sch:param name="ruleText"
value="' [IRM-ID-00028][Error] If element CountryCode has attribute vocabulary specified
as ISO-3 the element value must be in CVEnumIRMCoverageISO3166Trigraph.xml. '"/>

<sch:param name="codeDesc"
value="' This rule uses an abstract pattern to consolidate logic. It checks that the value
in parameter $searchTerm is contained in the parameter $list. The parameter $searchTerm
is relative in scope to the parameter $context. The value for the parameter $list is a
variable defined in the main document (IRM_XML.sch), which reads values from a specific
CVE file. '"/>

<sch:param name="context" value="//irm:CountryCode[@irm:vocabulary='ISO-3']"/>
<sch:param name="searchTerm" value="."/>
<sch:param name="list" value="$coverageIso3166TrigraphList"/>
<sch:param name="errMsg"
value="' [IRM-ID-00028][Error] If element CountryCode has attribute vocabulary specified
as ISO-3 the element value must be in CVEnumIRMCoverageISO3166Trigraph.xml. '"/>
</sch:pattern>
```

Rule: IRM-ID-00029

FileName:./Rules/generalConstraints/IRM_ID_00029.sch

Rule Description:

[IRM-ID-00029][Error] If element CountryCode has attribute precedence with a value of Secondary, there must be at least one sibling element CountryCode for which attribute precedence has a value of Primary. Human Readable: If a secondary country code is provided, there must also be a primary country code.

Code Description:

If there is an element CountryCode with attribute precedence specified with a value of [Secondary], then we make sure that there is at least one sibling CountryCode element with attribute precedence specified with a value of [Primary].

Schematron Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sch:pattern xmlns:sch="http://purl.oclc.org/dsdl/schematron" id="IRM-ID-00029">

<sch:rule context="irm:CountryCode[@irm:precedence='Secondary']">
<sch:assert id="IRM-00029" test="../irm:CountryCode[@irm:precedence='Primary']"
flag="error">
[IRM-ID-00029][Error]
If element CountryCode has attribute precedence with a value of Secondary,
there must be at least one sibling element CountryCode for which attribute
precedence has a value of Primary.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: IRM-ID-00030

FileName:./Rules/generalConstraints/IRM_ID_00030.sch

Rule Description:

[IRM-ID-00030][Error] If attribute order is specified with integer value N, there must exist other order attributes with values 1 to N-1 with no duplicates. Human Readable: The values of attribute order must be numbered sequentially with no duplicates, beginning at 1.

Code Description:

A list, named \$orderList, is created containing the value of each order attribute within the document after normalizing to remove extra white-space. If the total number of items in \$orderList does not equal the number of distinct values in \$orderList, then a duplicate exists and we return false. Otherwise, we make sure that each number from 1 to N, where N is the number of items in \$orderList, is contained within \$orderList. If each number is contained, then we return true. Otherwise, false.

Schematron Code:

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

<sch:rule context="irm:ICResourceMetadataPackage[./*/@irm:order]">
<sch:let name="orderList"
value="tokenize(string-join(./*/@irm:order/normalize-space(), ' '), ' ')" />
<sch:assert id="IRM-00030"
test="count(distinct-values($orderList)) = count($orderList) and (every $index in 1 to
count($orderList) satisfies index-of($orderList, xs:string($index))) "
flag="error">
[IRM-ID-00030][Error]
If attribute order is specified with integer value N, there must exist
other order attributes with values 1 to N-1 with no duplicates.

Human Readable: The values of attribute order must be numbered
sequentially with no duplicates, beginning at 1.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: IRM-ID-00031

FileName:./Rules/ddmsConstraints/IRM_ID_00031.sch

Rule Description:

[IRM-ID-00031][Error] The element ddms:countryCode must have the attribute ddms:qualifier specified with a value in CVEnumIRMCompoundCountryCodeQualifierType.xml. Human Readable: If a qualifier is specified for a country code, it must have be defined in the CompoundCountryCodeQualifierType CVE.

Code Description:

This rule uses an abstract pattern to consolidate logic. It checks that the value in parameter \$searchTerm is contained in the parameter \$list. The parameter \$searchTerm is relative in scope to the parameter \$context. The value for the parameter \$list is a variable defined in the main document (IRM_XML.sch), which reads values from a specific CVE file.

Schematron Code:

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

<sch:param name="ruleText"
value="' [IRM-ID-00031][Error] The element ddms:countryCode must have the attribute
ddms:qualifier specified with a value in CVEnumIRMCompoundCountryCodeQualifierType.xml.
' "/>

<sch:param name="codeDesc"
value="' This rule uses an abstract pattern to consolidate logic. It checks that the value
in parameter $searchTerm is contained in the parameter $list. The parameter $searchTerm
is relative in scope to the parameter $context. The value for the parameter $list is a
variable defined in the main document (IRM_XML.sch), which reads values from a specific
CVE file. ' "/>

<sch:param name="context" value="irm:DescribedItemIRM//ddms:countryCode"/>
<sch:param name="searchTerm" value="@ddms:qualifier"/>
<sch:param name="list" value="$compoundCountryCodeQualifierTypeList"/>
<sch:param name="errMsg"
value="' [IRM-ID-00031][Error] The element countryCode must have the attribute qualifier
specified with a value in CVEnumIRMCompoundCountryCodeQualifierType.xml. ' "/>
</sch:pattern>
```

Rule: IRM-ID-00032

FileName:../Rules/ddmsConstraints/IRM_ID_00032.sch

Rule Description:

[IRM-ID-00032][Error] For element ddms:countryCode, attribute ddms:value must be a single block of text without whitespace. Human Readable: The value for attribute ddms:countryCode/@ddms:value must be a single token.

Code Description:

First, normalize the space of the string value of the attribute ddms:value. Then, make sure that the tokenized list of that string only contains one token.

Schematron Code:

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

<sch:rule context="irm:DescribedItemIRM//ddms:countryCode">
<sch:assert test="count(tokenize(normalize-space(string(@ddms:value)), ' ')) = 1"
flag="error">
[IRM-ID-00032][Error] For element ddms:countryCode, attribute ddms:value must be
a single block of text without whitespace.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: IRM-ID-00033

FileName:./Rules/ddmsConstraints/IRM_ID_00033.sch

Rule Description:

[IRM-ID-00033][Error] If element ddms:mimeType is specified, it must have a value from CVENumIRMMimeType.xml.

Human Readable: Values for ddms:mimeType must be defined in the MimeType CVE.

Code Description:

This rule uses an abstract pattern to consolidate logic. It checks that the value in parameter \$searchTerm is contained in the parameter \$list. The parameter \$searchTerm is relative in scope to the parameter \$context. The value for the parameter \$list is a variable defined in the main document (IRM_XML.sch), which reads values from a specific CVE file.

Schematron Code:

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

<sch:param name="ruleText"
value="' [IRM-ID-00033][Error] If element ddms:mimeType is specified, it must have a value
from CVENumIRMMimeType.xml. '"/>

<sch:param name="codeDesc"
value="' This rule uses an abstract pattern to consolidate logic. It checks that the value
in parameter $searchTerm is contained in the parameter $list. The parameter $searchTerm
is relative in scope to the parameter $context. The value for the parameter $list is a
variable defined in the main document (IRM_XML.sch), which reads values from a specific
CVE file. '"/>

<sch:param name="context" value="irm:DescribedItemIRM//ddms:mimeType"/>
<sch:param name="searchTerm" value="."/>
<sch:param name="list" value="$mimeTypeList"/>
<sch:param name="errMsg"
value="' [IRM-ID-00033][Error] If element ddms:mimeType is specified, it must have a value
from CVENumIRMMimeType.xml. '"/>
</sch:pattern>
```

Rule: IRM-ID-00034

FileName:../Rules/ddmsConstraints/IRM_ID_00034.sch

Rule Description:

[IRM-ID-00034][Error] For element ddms:language, attribute ddms:qualifier must have a value in CVEnumIRMCompoundLanguageQualifierType.xml. Human Readable: If a qualifier is specified for a language, it must appear in the CompoundLanguageQualifierType CVE.

Code Description:

This rule uses an abstract pattern to consolidate logic. It checks that the value in parameter \$searchTerm is contained in the parameter \$list. The parameter \$searchTerm is relative in scope to the parameter \$context. The value for the parameter \$list is a variable defined in the main document (IRM_XML.sch), which reads values from a specific CVE file.

Schematron Code:

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

<sch:param name="ruleText"
value="' [IRM-ID-00034][Error] For element ddms:language, attribute ddms:qualifier must
have a value in CVEnumIRMCompoundLanguageQualifierType.xml. '"/>

<sch:param name="codeDesc"
value="' This rule uses an abstract pattern to consolidate logic. It checks that the value
in parameter $searchTerm is contained in the parameter $list. The parameter $searchTerm
is relative in scope to the parameter $context. The value for the parameter $list is a
variable defined in the main document (IRM_XML.sch), which reads values from a specific
CVE file. '"/>

<sch:param name="context" value="irm:DescribedItemIRM//ddms:language"/>
<sch:param name="searchTerm" value="@ddms:qualifier"/>
<sch:param name="list" value="$compoundLanguageQualifierTypeList"/>
<sch:param name="errMsg"
value="' [IRM-ID-00034][Error] For element ddms:language, attribute ddms:qualifier must
have a value in CVEnumIRMCompoundLanguageQualifierType.xml. '"/>
</sch:pattern>
```


Rule: IRM-ID-00035

FileName:../Rules/ddmsConstraints/IRM_ID_00035.sch

Rule Description:

[IRM-ID-00035][Error] For element ddms:Resource, child element ddms:language must be specified at least once. Human Readable: Every Resource must specify its language.

Code Description:

Make sure that element ddms:Resource has at least one child element ddms:language.

Schematron Code:

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

<sch:rule context="irm:DescribedItemIRM//ddms:Resource">
<sch:assert test="count(ddms:language) > 0">
[IRM-ID-00035][Error] For element ddms:Resource, child element ddms:language must be
specified at least once.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: IRM-ID-00036

FileName:../Rules/xlinkConstraints/IRM_ID_00036.sch

Rule Description:

[IRM-ID-00036][Error] For any element, if any attribute is specified with the xlink namespace 'http://www.w3.org/1999/xlink', then attributes xlink:type and/or xlink:href must be specified. Human Readable: If any XLink attributes are specified for an element, then the type and/or URL of the link must also be specified.

Code Description:

Makes sure that for each element that has any attribute in the xlink namespace has either xlink:type or xlink:href specified.

Schematron Code:

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

<sch:rule context="*[@xlink:*]">
<sch:assert id="IRM-00036"
test="normalize-space(string(@xlink:type)) or normalize-space(string(@xlink:href))"
flag="error">
[IRM-ID-00036][Error] For any element, if any attribute is specified with the
xlink namespace 'http://www.w3.org/1999/xlink', then attributes xlink:type and/or
xlink:href must be specified.
</sch:assert>
</sch:rule>
</sch:pattern>
```

Rule: IRM-ID-00037

FileName:./Rules/generalConstraints/IRM_ID_00037.sch

Rule Description:

[IRM-ID-00037][Error] If the attribute @ism:pocType is specified with a value of [ORCON], then there must exist a descendant element ddms:phone with a value specified. Human Readable: Elements denoted as POCs for an ORIGINATOR CONTROLLED memo must specify a phone number for that POC.

Code Description:

For any node that has the attribute @ism:pocType with a value of 'ORCON', the code will determine if there exists a descendant element ddms:phone with a non-null, non-whitespace text value. If the value exists, the rule will return true; otherwise, false will be returned.

Schematron Code:

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

  <sch:rule context="irm:*[tokenize(string(@ism:pocType),' ')= 'ORCON']">
    <sch:assert id="IRM-00037" test="normalize-space(descendant::ddms:phone/text())"
      flag="error">
      [IRM-ID-00037][Error]
      If the attribute @ism:pocType is specified with a value of [ORCON], then there
      must exist a descendant element ddms:phone with a value specified.

      Human Readable: Elements denoted as POCs for an ORIGINATOR CONTROLLED memo
      must specify a phone number for that POC.
    </sch:assert>
  </sch:rule>

</sch:pattern>
```

For any questions or comments regarding the IRM rules, contact us by email at datastandardssupport@ugov.gov.