Guide to XSLs for ISM

ISM XSL Guide

Version 2021-NOVr2022-NOV

December 1, 2022

Table of Contents

Chapter 1 - Introduction

1.1 - Purpose

This is an informative supplement for ISM. This document provides XSL files developed for ISM.

Chapter 2 - XSL Files

2.1 - BannerMapping.xml

```
<?xml-model href="../../Schematron/ISM/ISM_XML.sch" type="application/xml" schematypens="http://purl.oclc.org/dsdl/schematron"?><!--
**************************************************************** --><!--                          UNCLASSIFIED                                --><!--
**************************************************************** --><!-- ****************************************************************
  INTELLIGENCE COMMUNITY TECHNICAL SPECIFICATION
  XML DATA ENCODING SPECIFICATION FOR
  INFORMATION SECURITY MARKING METADATA (ISM.XML)
  ****************************************************************
  Module:   IC-ISM-SecurityBanner.xsl
  Date:     2020-10-21
  Creators: Office of the Director of National Intelligence
  Intelligence Community Chief Information Officer
  **************************************************************** --><!-- **************************************************************** --><!--
DESCRIPTION                            --><!--                                                                        --><!-- This configuration file maps from portion marks which are what   --><!--
ISM uses for tokens to the correct full text for a banner mark   --><!-- **************************************************************** --><BannerMapping xml:lang="en"
                                 ism:resourceElement="true"
                                 ism:DESVersion="202111.202211"
                                 ism:ISMCATCESVersion="202211"
                                 ism:createDate="2013-03-08"
                                 ism:compliesWith="USGov USIC"
                                 ism:classification="U"
                                 ism:ownerProducer="USA">
                    <DissemControls>
                       <BannerMap ism:ownerProducer="USA" ism:classification="U" portion="RS">RSEN</BannerMap>
                       <BannerMap ism:ownerProducer="USA" ism:classification="U" portion="OC">ORCON</BannerMap>
                       <BannerMap ism:ownerProducer="USA" ism:classification="U" portion="IMC">IMCON</BannerMap>
                       <BannerMap ism:ownerProducer="USA" ism:classification="U" portion="NF">NOFORN</BannerMap>
                       <BannerMap ism:ownerProducer="USA" ism:classification="U" portion="PR">PROPIN</BannerMap>
                       <BannerMap ism:ownerProducer="USA" ism:classification="U" portion="DSEN">DEA SENSITIVE</BannerMap>
                       <BannerMap ism:ownerProducer="USA" ism:classification="U" portion="AC">Attorney-Client</BannerMap>
                       <BannerMap ism:ownerProducer="USA" ism:classification="U" portion="AWP">Attorney-WP</BannerMap>
                       <BannerMap ism:ownerProducer="USA" ism:classification="U" portion="DELIB">Deliberative</BannerMap>
                    </DissemControls>

                    <NonICControls>
                       <BannerMap ism:ownerProducer="USA" ism:classification="U" portion="DS">LIMDIS</BannerMap>
                       <BannerMap ism:ownerProducer="USA" ism:classification="U" portion="XD">EXDIS</BannerMap>
                       <BannerMap ism:ownerProducer="USA" ism:classification="U" portion="ND">NODIS</BannerMap>
                       <BannerMap ism:ownerProducer="USA" ism:classification="U" portion="SBU-NF">SBU NOFORN</BannerMap>
                       <BannerMap ism:ownerProducer="USA" ism:classification="U" portion="LES-NF">LES NOFORN</BannerMap>
                    </NonICControls>

                    <Classification>
                       <BannerMap ism:ownerProducer="USA" ism:classification="U" portion="TS">TOP SECRET</BannerMap>
                       <BannerMap ism:ownerProducer="USA" ism:classification="U" portion="S">SECRET</BannerMap>
                       <BannerMap ism:ownerProducer="USA" ism:classification="U" portion="C">CONFIDENTIAL</BannerMap>
                       <BannerMap ism:ownerProducer="USA" ism:classification="U" portion="R">RESTRICTED</BannerMap>
                       <BannerMap ism:ownerProducer="USA" ism:classification="U" portion="U">UNCLASSIFIED</BannerMap>
                    </Classification>
```

```
            </BannerMapping>
```

2.2 - IC-ISM-ClassDeclass.xsl

```
<!-- **************************************************************** --><!--                           UNCLASSIFIED                          --><!--
**************************************************************** --><!-- ****************************************************************
  INTELLIGENCE COMMUNITY TECHNICAL SPECIFICATION
  XML DATA ENCODING SPECIFICATION FOR
  INFORMATION SECURITY MARKING METADATA (ISM.XML)
  ******************************************************************* --><!-- Module:     IC-ISM-ClassDeclass.xsl                             --><!-- Date:
2011-08-12                                             --><!-- Creators: Office of the Director of National Intelligence
     Intelligence Community Chief Information Officer                --><!-- **************************************************************** --><!--
**************************************************************** --><!--                           INTRODUCTION                          --
><!--                                                                  --><!-- Intelligence Community Information Security Marking (IC ISM)     --><!-- standard was developed by the IC
Security Panel for the express  --><!-- purpose of promoting CAPCO security marking interoperability     --><!-- between members of the Intelligence Community.                   --><!--
**************************************************************** --><!-- **************************************************************** --><!--
DESCRIPTION                           --><!--                                                                        --><!-- This stylesheet outputs classification/declassification block    --><!--
content including the "Classified by", "Reason", "Derived from", --><!-- and/or "Declassify on" information required by the CAPCO         --><!-- Implementation Manual pursuant to ISOO
Directive 1 and Executive --><!-- Order 13526.                                                                  --><!-- **************************************************************** --><xsl:stylesheet
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                               xmlns:xs="http://www.w3.org/2001/XMLSchema"
                               xmlns:ism-func="urn:us:gov:ic:ism:functions"
                               version="2.0">

                  <xsl:output method="text"
                              encoding="UTF-8"
                              media-type="text-plain"
                              indent="no"/>
                  <!-- If including this xsl causes "Content is not allowed in prolog" the importing
  XSL is likely missing an output declaration -->

  <xsl:import href="IC-ISM-SecurityBanner.xsl"/>

                  <xsl:param name="CUIRenderingRuleSet" select="''"/>

                  <!--***********************************************-->
  <!-- Generate the Classification Authority Block for the current element-->
  <!--***********************************************-->
  <xsl:template match="*[@ism:*]" mode="ism:authority">
                     <xsl:param name="delimiter"/>
                     <xsl:call-template name="get.class.declass">
                        <xsl:with-param name="delimiter" select="$delimiter"/>
                     </xsl:call-template>
                  </xsl:template>

                  <!-- **************************************************************** -->
  <!-- get.class.declass - Calls template class.declass with parameters from the element's ISM attributes-->
  <!-- **************************************************************** -->
  <xsl:template name="get.class.declass">
                     <xsl:param name="delimiter"/>

                     <xsl:call-template name="class.declass">
                        <xsl:with-param name="classification" select="@ism:classification"/>
                        <xsl:with-param name="classifiedby" select="@ism:classifiedBy"/>
                        <xsl:with-param name="derivativelyclassifiedby" select="@ism:derivativelyClassifiedBy"/>
                        <xsl:with-param name="classificationreason" select="@ism:classificationReason"/>
```

```
                      <xsl:with-param name="derivedfrom" select="@ism:derivedFrom"/>
                      <xsl:with-param name="declassdate" select="@ism:declassDate"/>
                      <xsl:with-param name="declassexception" select="@ism:declassException"/>
                      <xsl:with-param name="declassevent" select="@ism:declassEvent"/>
                      <xsl:with-param name="cuiControlledBy" select="@ism:cuiControlledBy"/>
                      <xsl:with-param name="cuiDecontrolDate" select="@ism:cuiDecontrolDate"/>
                      <xsl:with-param name="cuiDecontrolEvent" select="@ism:cuiDecontrolEvent"/>
                      <xsl:with-param name="cuiControlledByOffice" select="@ism:cuiControlledByOffice"/>
                      <xsl:with-param name="cuiPOC" select="@ism:cuiPOC"/>
                      <xsl:with-param name="sci" select="./@ism:SCIcontrols"/>
                      <xsl:with-param name="atomicenergymarkings" select="./@ism:atomicEnergyMarkings"/>
                      <xsl:with-param name="sar" select="./@ism:SARIdentifier"/>
                      <xsl:with-param name="fgiopen" select="./@ism:FGIsourceOpen"/>
                      <xsl:with-param name="fgiprotect" select="./@ism:FGIsourceProtected"/>
                      <xsl:with-param name="nonic" select="./@ism:nonICmarkings"/>
                      <xsl:with-param name="dissem" select="./@ism:disseminationControls"/>
                      <xsl:with-param name="releaseto" select="./@ism:releasableTo"/>
                      <xsl:with-param name="displayonly" select="./@ism:displayOnlyTo"/>
                      <xsl:with-param name="cuiBasic" select="./@ism:cuiBasic"/>
                      <xsl:with-param name="cuiSpecified" select="./@ism:cuiSpecified"/>
                      <xsl:with-param name="ownerproducer" select="./@ism:ownerProducer"/>
                      <xsl:with-param name="overallCompliesWith" select="./@ism:compliesWith"/>
                      <xsl:with-param name="delimiter" select="$delimiter"/>
                   </xsl:call-template>

                </xsl:template>

                <!-- **************************************************************** -->
<!-- class.declass - Determines the class/declass block content and   -->
<!--                 calls a template to concatenate the content into -->
<!--                 a delimited string                               -->
<!-- **************************************************************** -->
<xsl:template name="class.declass">
                   <xsl:param name="classification"/>
                   <xsl:param name="classifiedby"/>
                   <xsl:param name="derivativelyclassifiedby"/>
                   <xsl:param name="classificationreason"/>
                   <xsl:param name="derivedfrom"/>
                   <xsl:param name="declassdate"/>
                   <xsl:param name="declassexception"/>
                   <xsl:param name="declassevent"/>
                   <xsl:param name="delimiter"/>
                   <xsl:param name="cuiControlledBy"/>
                   <xsl:param name="cuiDecontrolDate"/>
                   <xsl:param name="cuiDecontrolEvent"/>
                   <xsl:param name="cuiControlledByOffice"/>
                   <xsl:param name="cuiPOC"/>
                   <xsl:param name="cuiBasic"/>
                   <xsl:param name="cuiSpecified"/>
                   <xsl:param name="sci"/>
                   <xsl:param name="atomicenergymarkings"/>
                   <xsl:param name="sar"/>
                   <xsl:param name="fgiopen"/>
                   <xsl:param name="fgiprotect"/>
```

```
                 <xsl:param name="nonic"/>
                 <xsl:param name="dissem"/>
                 <xsl:param name="releaseto"/>
                 <xsl:param name="displayonly"/>
                 <xsl:param name="ownerproducer"/>
                 <xsl:param name="overallCompliesWith"/>
                 <xsl:param name="compliesWith" select="./@ism:compliesWith"/>
                 <!-- replace with overall complies with when build umbrella stylesheets -->

<xsl:variable name="class-declass-delimiter">
                    <xsl:choose>
                       <xsl:when test="not($delimiter) or ($delimiter = '')">
                          <xsl:text>|</xsl:text>
                       </xsl:when>
                       <xsl:otherwise>
                          <xsl:value-of select="$delimiter"/>
                       </xsl:otherwise>
                    </xsl:choose>
                 </xsl:variable>

                 <xsl:variable name="n-classification" select="normalize-space($classification)"/>
                 <xsl:variable name="n-classifiedby" select="normalize-space($classifiedby)"/>
                 <xsl:variable name="n-derivativelyclassifiedby"
                               select="normalize-space($derivativelyclassifiedby)"/>
                 <xsl:variable name="n-classificationreason"
                               select="normalize-space($classificationreason)"/>
                 <xsl:variable name="n-derivedfrom" select="normalize-space($derivedfrom)"/>
                 <xsl:variable name="n-declassdate" select="normalize-space($declassdate)"/>
                 <xsl:variable name="n-declassexception" select="normalize-space($declassexception)"/>
                 <xsl:variable name="n-declassevent" select="normalize-space($declassevent)"/>
                 <xsl:variable name="n-cuiControlledBy" select="normalize-space($cuiControlledBy)"/>
                 <xsl:variable name="n-cuiDecontrolDate" select="normalize-space($cuiDecontrolDate)"/>
                 <xsl:variable name="n-cuiDecontrolEvent" select="normalize-space($cuiDecontrolEvent)"/>
                 <xsl:variable name="n-overallCompliesWith"
                               select="normalize-space($overallCompliesWith)"/>
                 <xsl:variable name="n-compliesWith" select="normalize-space($compliesWith)"/>
                 <xsl:variable name="n-cuiControlledByOffice"
                               select="normalize-space($cuiControlledByOffice)"/>
                 <xsl:variable name="n-cuiPOC" select="normalize-space($cuiPOC)"/>

                 <!-- Sort ownerproducer Based on CVE -->
<xsl:variable name="n-ownerproducer">
                    <xsl:value-of select="ism-func:sortOwnerProducer($ownerproducer)"/>
                 </xsl:variable>

                 <!-- Sort cuiBasic alphabetically -->
<xsl:variable name="n-cuiBasic">
                    <xsl:value-of select="ism-func:sortCuiBasic($cuiBasic)"/>
                 </xsl:variable>

                 <!-- Sort cuiSpecified alphabetically -->
<xsl:variable name="n-cuiSpecified">
                    <xsl:value-of select="ism-func:sortCuiSpecified($cuiSpecified)"/>
                 </xsl:variable>
```

```
                     <!-- **** Determine the cuiSpecified marking **** -->
    <xsl:variable name="cuiSpecifiedVal">
                        <xsl:if test="$n-cuiSpecified != ''">
                           <xsl:value-of select="ism-func:get.cuiSpecified($n-cuiSpecified)"/>
                        </xsl:if>
                     </xsl:variable>

                     <!-- **** Determine the set of dissemination controls that are CUI limited dissem controls -->
    <xsl:variable name="dissemCUI">
                        <xsl:if test="$dissem != ''">
                           <xsl:value-of select="ism-func:get.dissemCUI($dissem)"/>
                        </xsl:if>
                     </xsl:variable>

                     <!-- **** Determine the set of dissemination controls that are NOT CUI limited dissem controls -->
    <xsl:variable name="dissemsNotCui">
                        <xsl:if test="$dissem != ''">
                           <xsl:value-of select="ism-func:get.dissemNotCUI($dissem)"/>
                        </xsl:if>
                     </xsl:variable>

                     <!-- Variable to determine if there are any IC Register-specific control markings that are not CUI limited dissem controls -->
    <xsl:variable name="CUIandICcontrolMarkings">
                        <xsl:choose>
                           <xsl:when test="($cuiBasic != '' or $cuiSpecified != '') and ($n-classification = '' or $n-classification = 'U') and           (string($sci) = '' and string($sar) =
'' and string($atomicenergymarkings) = '' and            string($fgiopen) = '' and string($fgiprotect) = '' and string($nonic) = '' and string($dissemsNotCui) = '')">
                              <xsl:value-of select="false()"/>
                           </xsl:when>
                           <xsl:otherwise>
                              <xsl:value-of select="true()"/>
                           </xsl:otherwise>
                        </xsl:choose>
                     </xsl:variable>

                     <!-- Sort Dissem Based on CVE and on the value of the document's ism:compliesWith -->
      <xsl:variable name="n-dissem">
                        <xsl:variable name="sortedDissem"
                                      select="ism-func:sortDissemControls($dissemCUI, $compliesWith, $CUIandICcontrolMarkings)"/>
                        <xsl:value-of select="replace(normalize-space($sortedDissem), 'OC OC-USGOV', 'ORCON-USGOV')"/>
                     </xsl:variable>

                     <!-- Sort RelTo Based on CVE -->
    <xsl:variable name="n-releaseto">
                        <xsl:value-of select="ism-func:sortReleaseto($releaseto)"/>
                     </xsl:variable>

                     <!-- Sort DisplayOnly Based on CVE -->
    <xsl:variable name="n-displayonly">
                        <xsl:value-of select="ism-func:sortDisplayonly($displayonly)"/>
                     </xsl:variable>

                     <xsl:if test="         ($n-classification and ($n-classification != 'U')) or         ($n-classifiedby or $n-classificationreason or $n-derivedfrom or         $n-
declassdate or $n-declassexception or $n-declassevent)         or ($n-compliesWith = 'USA-CUI-ONLY' or contains($n-compliesWith, 'USA-CUI'))">
```

```
                        <xsl:variable name="warning-missing-classif">
                           <xsl:if test="$n-classification = '' and not($n-compliesWith = 'USA-CUI-ONLY')">
                              <xsl:text>( WARNING! This document does not contain a required overall classification marking. )</xsl:text>
                           </xsl:if>
                        </xsl:variable>
                        <xsl:variable name="warning-unclass-and-classified-markings">
                           <xsl:if test="             ($n-classification = 'U' and ($n-classifiedby or $n-classificationreason or $n-derivedfrom or             $n-declassdate or $n-
declassexception or $n-declassevent))">
                              <xsl:text>( WARNING! This document contains overall markings for both an unclassified and a classified document. )</xsl:text>
                           </xsl:if>
                        </xsl:variable>
                        <xsl:variable name="warning-missing-classifiedBy">
                           <xsl:if test="$n-classification and ($n-classification != 'U') and ($n-classifiedby = '') and ($n-derivedfrom = '')">
                              <xsl:text>( WARNING! This document does not contain required markings for either an originally or derivatively classified document. )</xsl:text>
                           </xsl:if>
                        </xsl:variable>
                        <xsl:variable name="classified-by-line" select="$n-classifiedby"/>
                        <xsl:variable name="derivatively-classified-by-line"
                                      select="$n-derivativelyclassifiedby"/>
                        <xsl:variable name="derived-from-line" select="$n-derivedfrom"/>
                        <xsl:variable name="reason-line" select="$n-classificationreason"/>
                        <xsl:variable name="warning-both-original-and-derivatively-classified">
                           <xsl:if test="($n-classification != 'U') and $n-classifiedby and $n-derivedfrom">
                              <xsl:text>( WARNING! This document contains markings for both an originally and derivatively classified document. )</xsl:text>
                           </xsl:if>
                        </xsl:variable>
                        <xsl:variable name="warning-missing-classificationReason">
                           <xsl:if test="$n-classification and ($n-classification != 'U') and $n-classifiedby and ($n-classificationreason = '')">
                              <xsl:text>( WARNING! The reason for the classification decision should be specified for an originally classified document. )</xsl:text>
                           </xsl:if>
                        </xsl:variable>
                        <xsl:variable name="warning-missing-declass-instructions">
                           <xsl:if test="             $n-classification and ($n-classification != 'U') and             ($n-declassdate = '') and ($n-declassexception = '') and ($n-declassevent
= '')">
                              <xsl:text>( WARNING! This document does not contain required declassification instructions or markings. )</xsl:text>
                           </xsl:if>
                        </xsl:variable>
                        <xsl:variable name="warning-missing-declass-info">
                           <xsl:if test="             $n-classification and ($n-classification != 'U')             and $n-declassexception             and not(contains($n-declassexception,
'AEA'))             and not(contains($n-declassexception, 'NATO'))             and not(contains($n-declassexception, 'NATO-AEA'))             and not(contains($n-declassexception, '25X1-
human'))             and not(contains($n-declassexception, '50X1-HUM'))             and not(contains($n-declassexception, '50X2-WMD'))             and ($n-declassdate = '')             and ($n-
declassevent = '')">
                              <xsl:text>( WARNING! A declassification date or event should be specified for a document with a 25X or 50X declassification exemption, unless the document has a
declassification exemption of 25X1-human, 50X1-HUM, 50X2-WMD, AEA, or  NATO. )</xsl:text>
                           </xsl:if>
                        </xsl:variable>
                        <xsl:variable name="declassify-on-line">
                           <xsl:if test="$n-declassexception">
                              <xsl:choose>
                                 <xsl:when test="$n-declassexception = 'AEA'">
                                    <xsl:text>Not Applicable to RD/FRD/TFNI portions. See source list for NSI portions.</xsl:text>
                                 </xsl:when>
                                 <xsl:when test="$n-declassexception = 'NATO'">
                                    <xsl:text>Not Applicable to NATO portions. See source list for NSI portions.</xsl:text>
                                 </xsl:when>
                                 <xsl:when test="$n-declassexception = 'NATO-AEA'">
                                    <xsl:text>Not Applicable to RD/FRD/TFNI and NATO portions. See source list for NSI portions.</xsl:text>
                                 </xsl:when>
                                 <xsl:otherwise>
                                    <xsl:value-of select="$n-declassexception"/>
                                 </xsl:otherwise>
                              </xsl:choose>
                           </xsl:if>
                           <xsl:if test="$n-declassdate">
                              <xsl:if test="$n-declassexception">
                                 <xsl:text>, </xsl:text>
                              </xsl:if>
                              <xsl:value-of select="format-date(xs:date($n-declassdate), '[MNn] [D], [Y0001]')"/>
                           </xsl:if>
                           <xsl:if test="$n-declassevent">
                              <xsl:if test="$n-declassexception and ($n-declassdate = '')">
                                 <xsl:text>, </xsl:text>
                              </xsl:if>
                              <xsl:if test="$n-declassdate">
                                 <xsl:text> or </xsl:text>
                              </xsl:if>
                              <xsl:value-of select="$n-declassevent"/>
                           </xsl:if>
                        </xsl:variable>
                        <xsl:variable name="warning-invalid-declass-date-and-exemption">
                           <xsl:if test="$n-declassdate and (contains($n-declassexception, '25X1-human'))">
                              <xsl:text>( WARNING! This document contains both a declassification date and a declassification exemption of 25X1-human. )</xsl:text>
                           </xsl:if>
                        </xsl:variable>
                        <xsl:variable name="warning-invalid-declass-event-and-exemption">
                           <xsl:if test="$n-declassevent and (contains($n-declassexception, '25X1-human'))">
                              <xsl:text>( WARNING! This document contains both a declassification event and a declassification exemption of 25X1-human. )</xsl:text>
                           </xsl:if>
                        </xsl:variable>

                        <!-- **************************************************************** -->
      <!-- Get data for CUI block if document contains CUI                  -->
      <!-- **************************************************************** -->

      <xsl:variable name="warning-missing-cuiControlledBy">
                           <xsl:if test="$n-compliesWith = 'USA-CUI-ONLY' or contains($n-compliesWith, 'USA-CUI')">
                              <xsl:if test="$n-cuiControlledBy = ''">
                                 <xsl:text>( WARNING! This document contains CUI markings but does not contain a required overall Controlled By: marking. )</xsl:text>
                              </xsl:if>
                           </xsl:if>
                        </xsl:variable>
                        <xsl:variable name="cui-controlled-by-line" select="$n-cuiControlledBy"/>

                        <xsl:variable name="cui-decontrol-on-line">
                           <xsl:if test="$n-cuiDecontrolDate">
                              <xsl:value-of select="format-date(xs:date($n-cuiDecontrolDate), '[MNn] [D], [Y0001]')"/>
                           </xsl:if>
                           <xsl:if test="$n-cuiDecontrolEvent">
```

```
                        <xsl:if test="$n-cuiDecontrolDate">
                           <xsl:text> or </xsl:text>
                        </xsl:if>
                        <xsl:value-of select="$n-cuiDecontrolEvent"/>
                     </xsl:if>
                  </xsl:variable>

                  <xsl:variable name="cui-controlled-by-office-line" select="$n-cuiControlledByOffice"/>
                  <xsl:variable name="cui-POC-line" select="$n-cuiPOC"/>

                  <xsl:variable name="cui-basic-line" select="$n-cuiBasic"/>
                  <xsl:variable name="cui-specified-line" select="replace($cuiSpecifiedVal, '/', ' ')"/>

                  <xsl:variable name="cui-categories-line">
                     <xsl:choose>
                        <xsl:when test="$cui-basic-line != '' and $cui-specified-line != ''">
                           <xsl:value-of select="concat(string($cui-basic-line),' ',string($cui-specified-line))"/>
                        </xsl:when>
                        <xsl:otherwise>
                           <xsl:value-of select="concat(string($cui-basic-line), string($cui-specified-line))"/>
                        </xsl:otherwise>
                     </xsl:choose>
                  </xsl:variable>

                  <!-- **** Determine the dissemination marking **** -->
<xsl:variable name="dissemVal">
                     <xsl:if test="$n-dissem != ''">
                        <xsl:call-template name="ism:get.dissem.banner">
                           <xsl:with-param name="all" select="$n-dissem"/>
                           <xsl:with-param name="relto" select="$n-releaseto"/>
                           <xsl:with-param name="displayonly" select="$n-displayonly"/>
                           <xsl:with-param name="ownerproducer" select="$n-ownerproducer"/>
                        </xsl:call-template>
                     </xsl:if>
                  </xsl:variable>

                  <xsl:call-template name="ism:concat.class.declass">
                     <xsl:with-param name="warning-missing-classif"
                                     select="string($warning-missing-classif)"/>
                     <xsl:with-param name="warning-unclass-and-classified-markings"
                                     select="string($warning-unclass-and-classified-markings)"/>
                     <xsl:with-param name="warning-missing-classifiedBy"
                                     select="string($warning-missing-classifiedBy)"/>
                     <xsl:with-param name="warning-both-original-and-derivatively-classified"
                                     select="string($warning-both-original-and-derivatively-classified)"/>
                     <xsl:with-param name="warning-missing-classificationReason"
                                     select="string($warning-missing-classificationReason)"/>
                     <xsl:with-param name="warning-missing-declass-instructions"
                                     select="string($warning-missing-declass-instructions)"/>
                     <xsl:with-param name="warning-missing-declass-info"
                                     select="string($warning-missing-declass-info)"/>
                     <xsl:with-param name="warning-invalid-declass-date-and-exemption"
                                     select="string($warning-invalid-declass-date-and-exemption)"/>
                     <xsl:with-param name="warning-invalid-declass-event-and-exemption"
```

```
                                       select="string($warning-invalid-declass-event-and-exemption)"/>
                       <xsl:with-param name="warning-missing-cuiControlledBy"
                                       select="string($warning-missing-cuiControlledBy)"/>
                       <xsl:with-param name="classified-by-line" select="string($classified-by-line)"/>
                       <xsl:with-param name="derivatively-classified-by-line"
                                       select="string($derivatively-classified-by-line)"/>
                       <xsl:with-param name="derived-from-line" select="string($derived-from-line)"/>
                       <xsl:with-param name="reason-line" select="string($reason-line)"/>
                       <xsl:with-param name="declassify-on-line" select="string($declassify-on-line)"/>
                       <xsl:with-param name="cui-controlled-by-line" select="string($cui-controlled-by-line)"/>
                       <xsl:with-param name="cui-decontrol-on-line" select="string($cui-decontrol-on-line)"/>
                       <xsl:with-param name="cui-controlled-by-office-line"
                                       select="string($cui-controlled-by-office-line)"/>
                       <xsl:with-param name="cui-POC-line" select="string($cui-POC-line)"/>
                       <xsl:with-param name="cui-categories-line" select="$cui-categories-line"/>
                       <xsl:with-param name="dissemVal" select="$dissemVal"/>
                       <xsl:with-param name="delimiter" select="$class-declass-delimiter"/>
                    </xsl:call-template>
                 </xsl:if>
              </xsl:template>

              <xsl:template name="ism:concat.class.declass">
<!-- **************************************************************** -->
<!-- concat.class.declass - Concatenates class/declass block content  -->
<!--                        into a delimited string.  Generates CUI   -->
<!--                        Control Block if there are CUI markings.  -->
<!-- **************************************************************** -->
<xsl:param name="warning-missing-classif"/>
                 <xsl:param name="warning-unclass-and-classified-markings"/>
                 <xsl:param name="warning-missing-classifiedBy"/>
                 <xsl:param name="warning-both-original-and-derivatively-classified"/>
                 <xsl:param name="warning-missing-classificationReason"/>
                 <xsl:param name="warning-missing-declass-instructions"/>
                 <xsl:param name="warning-missing-declass-info"/>
                 <xsl:param name="warning-invalid-declass-date-and-exemption"/>
                 <xsl:param name="warning-invalid-declass-event-and-exemption"/>
                 <xsl:param name="warning-missing-cuiControlledBy"/>
                 <xsl:param name="classified-by-line"/>
                 <xsl:param name="derivatively-classified-by-line"/>
                 <xsl:param name="derived-from-line"/>
                 <xsl:param name="reason-line"/>
                 <xsl:param name="declassify-on-line"/>
                 <xsl:param name="cui-controlled-by-line"/>
                 <xsl:param name="cui-decontrol-on-line"/>
                 <xsl:param name="cui-controlled-by-office-line"/>
                 <xsl:param name="cui-POC-line"/>
                 <xsl:param name="cui-categories-line"/>
                 <xsl:param name="dissemVal"/>
                 <xsl:param name="delimiter"/>

                 <xsl:variable name="class-declass-content">
                    <xsl:if test="$warning-missing-classif">
                       <xsl:value-of select="$delimiter"/>
                       <xsl:value-of select="$warning-missing-classif"/>
```

```
                  </xsl:if>
                  <xsl:if test="$warning-unclass-and-classified-markings">
                     <xsl:value-of select="$delimiter"/>
                     <xsl:value-of select="$warning-unclass-and-classified-markings"/>
                  </xsl:if>
                  <xsl:if test="$warning-missing-classifiedBy">
                     <xsl:value-of select="$delimiter"/>
                     <xsl:value-of select="$warning-missing-classifiedBy"/>
                  </xsl:if>

                  <!-- **************************************************************** -->
<!-- Include the "Classified by" line or the "Derived from" line.     -->
<!--                                                                  -->
<!-- NOTE: A classified document can be either an originally          -->
<!--       classified document or a derivatively classified document. -->
<!--       An originally classified document will always contain a    -->
<!--       "Classified by" line.  A derivatively classified document  -->
<!--       may (somewhat misleadingly) contain a "Classified by" line -->
<!--       if attribute @derivativelyClassifiedBy exists, and will    -->
<!--       always contain a "Derived from" line.                      -->
<!-- **************************************************************** -->
<xsl:if test="$classified-by-line or $derived-from-line">
                     <xsl:value-of select="$delimiter"/>
                     <xsl:choose>
                        <xsl:when test="$classified-by-line">
                           <xsl:text>Classified by: </xsl:text>
                           <xsl:value-of select="$classified-by-line"/>
                        </xsl:when>
                        <xsl:when test="$derived-from-line">
                           <xsl:if test="$derivatively-classified-by-line">
                              <xsl:text>Classified by: </xsl:text>
                              <xsl:value-of select="$derivatively-classified-by-line"/>
                              <xsl:value-of select="$delimiter"/>
                           </xsl:if>
                           <xsl:text>Derived from: </xsl:text>
                           <xsl:value-of select="$derived-from-line"/>
                        </xsl:when>
                     </xsl:choose>
                  </xsl:if>

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

<!-- **************************************************************** -->
<!-- Include the "Reason" line.                                       -->
<!--                                                                  -->
<!-- NOTE: For originally classified documents, the reason for the    -->
<!--       classification decision should always be specified.        -->
<!--                                                                  -->
<!--       According to ISOO Directive 1, Section 2001.22(c), for     -->
<!--       derivatively classified documents, the reason for the      -->
<!--       original classification decision, as reflected in the      -->
<!--       source document(s) or classification guide, is not         -->
<!--       required.  If included, however, it shall conform to the   -->
<!--       standards in Section 2001.21(a)(3).                        -->
```

```
  <!--                                                                  -->
  <!--       In other words, the "Reason" line can be included in the   -->
  <!--       classification/declassification block for derivatively     -->
  <!--       classified documents.                                      -->
  <!-- **************************************************************** -->
  <xsl:if test="$reason-line">
                       <xsl:value-of select="$delimiter"/>
                       <xsl:text>Reason: </xsl:text>
                       <xsl:value-of select="$reason-line"/>
                    </xsl:if>

                    <!-- **************************************************************** -->
  <xsl:if test="$warning-both-original-and-derivatively-classified">
                       <xsl:value-of select="$delimiter"/>
                       <xsl:value-of select="$warning-both-original-and-derivatively-classified"/>
                    </xsl:if>
                    <xsl:if test="$warning-missing-classificationReason">
                       <xsl:value-of select="$delimiter"/>
                       <xsl:value-of select="$warning-missing-classificationReason"/>
                    </xsl:if>
                    <xsl:if test="$warning-missing-declass-instructions">
                       <xsl:value-of select="$delimiter"/>
                       <xsl:value-of select="$warning-missing-declass-instructions"/>
                    </xsl:if>

                    <!-- **************************************************************** -->
  <!-- Include the "Declassify on" line.                                -->
  <!-- **************************************************************** -->
  <xsl:if test="$declassify-on-line">
                       <xsl:value-of select="$delimiter"/>
                       <xsl:text>Declassify on: </xsl:text>
                       <xsl:value-of select="$declassify-on-line"/>
                    </xsl:if>

                    <!-- **************************************************************** -->
  <xsl:if test="$warning-missing-declass-info">
                       <xsl:value-of select="$delimiter"/>
                       <xsl:value-of select="$warning-missing-declass-info"/>
                    </xsl:if>
                    <xsl:if test="$warning-invalid-declass-date-and-exemption">
                       <xsl:value-of select="$delimiter"/>
                       <xsl:value-of select="$warning-invalid-declass-date-and-exemption"/>
                    </xsl:if>
                    <xsl:if test="$warning-invalid-declass-event-and-exemption">
                       <xsl:value-of select="$delimiter"/>
                       <xsl:value-of select="$warning-invalid-declass-event-and-exemption"/>
                    </xsl:if>
                 </xsl:variable>

                 <xsl:value-of select="substring-after($class-declass-content, $delimiter)"/>

                 <!-- **************************************************************** -->
<!--    Output CUI block information                                  -->
<!-- **************************************************************** -->
```

```
<xsl:variable name="cui-content">
                    <xsl:if test="$warning-missing-cuiControlledBy">
                       <xsl:value-of select="$delimiter"/>
                       <xsl:value-of select="$warning-missing-cuiControlledBy"/>
                    </xsl:if>

                    <xsl:if test="$cui-controlled-by-line">
                       <xsl:value-of select="$delimiter"/>
                       <xsl:text>Controlled By: </xsl:text>
                       <xsl:value-of select="$cui-controlled-by-line"/>
                    </xsl:if>

                    <xsl:if test="$cui-controlled-by-office-line">
                       <xsl:value-of select="$delimiter"/>
                       <xsl:text>Controlled By: </xsl:text>
                       <xsl:value-of select="$cui-controlled-by-office-line"/>
                    </xsl:if>

                    <xsl:if test="$CUIRenderingRuleSet = 'DOD'">
                       <xsl:value-of select="$delimiter"/>
                       <xsl:text>CUI Category: </xsl:text>
                       <xsl:value-of select="$cui-categories-line"/>
                       <xsl:if test="$dissemVal != ''">
                          <xsl:value-of select="$delimiter"/>
                          <xsl:text>Distribution/Dissemination Control: </xsl:text>
                          <xsl:value-of select="replace($dissemVal, '/', '; ')"/>
                       </xsl:if>
                    </xsl:if>

                    <xsl:if test="$cui-POC-line">
                       <xsl:value-of select="$delimiter"/>
                       <xsl:text>POC: </xsl:text>
                       <xsl:value-of select="$cui-POC-line"/>
                    </xsl:if>

                    <xsl:if test="$cui-decontrol-on-line">
                       <xsl:value-of select="$delimiter"/>
                       <xsl:text>Decontrol On: </xsl:text>
                       <xsl:value-of select="$cui-decontrol-on-line"/>
                    </xsl:if>

                 </xsl:variable>

                 <!-- Output CUI Control Block, with delimiter if document is classified otherwise no delimiter -->
<xsl:choose>
                    <xsl:when test="$classified-by-line or $derived-from-line">
                       <xsl:value-of select="$cui-content"/>
                    </xsl:when>
                    <xsl:otherwise>
                       <xsl:value-of select="substring-after($cui-content, $delimiter)"/>
                    </xsl:otherwise>
                 </xsl:choose>
```

```
                  </xsl:template>


               </xsl:stylesheet>
               <!-- **************************************************************** --><!--                          CHANGE HISTORY                          --
><!--                                                                  --><!-- Version 1.0                                                  --
><!--                                                                  --><!--   - Unpublished                                              --
><!--                                                                  --><!-- Version 2.0                                                  --
><!--                                                                  --><!--   - Initial published version corresponding to IC ISM v2.0    --
><!--                                                                  --><!-- Version 2.0.1                                                --
><!--                                                                  --><!--   - Modified to separate the rendering of content as HTML into   --><!--     an independent
template                                     --><!--                                                                  --><!--   - Other minor modifications
--><!--                                                                  --><!-- Version 2.0.2                                                --
><!--                                                                  --><!--   - Modified so that content is not rendered into any specific   --><!--     format by this stylesheet.  Instead
the content is simply    --><!--     concatenated into a delimited text string, where each        --><!--     section of the delimited string is a line of content within  --><!--     the class/
declass block, including various warning messages. --><!--     This method leaves it up to the calling stylesheet to parse  --><!--     the delimited string and render it according to the
desired  --><!--     output.  A delimiter can be set in the calling stylesheet    --><!--     and passed into the class.declass template as a parameter by --><!--     the template call in the
calling stylesheet.  If a delimiter --><!--     is not set and/or not passed in, the delimiter is set to a   --><!--     "|" in the class.declass template in this stylesheet, and    --
><!--     the calling stylesheet MUST parse the string based on this   --><!--     delimiter.                                               --
><!--                                                                  --><!-- Version 2.1                                                  --
><!--                                                                  --><!--   - Corresponds to IC ISM 2.1 (ISM-XML 1.0).                 --
><!--                                                                  --><!--   - Modified the "Declassify on" line (declassify-on-line) to    --><!--     exclude the optional time zone
indicator which may exist in  --><!--     the @declassDate and @dateOfExemptedSource attribute values. --><!--                                                                  --><!--   -
Added a template to convert date values from "YYYY-MM-DD"    --><!--     format to "Month [D]D, YYYY" format for the @declassDate and --><!--     @dateOfExemptedSource
attributes.                              --><!--                                                                  --><!--   - Added a recursive template to include a space after each     --
><!--     comma when multiple declassification exemption markings or   --><!--     multiple type of exempted source markings are specified.     --
><!--                                                                  --><!--   - Added a warning when a 25X declassification exemption (other --><!--     than 25X1-human) is specified and a
declassification date or --><!--     declassification event is not specified.                     --><!--                                                                  --><!--   - Modified
stylesheet to account for @derivativelyClassifiedBy --><!--     attribute.                                                   --
><!--                                                                  --><!-- 2010-09-24
  - Changed the name of warning variables to be more descriptive.
  - Namespace qualified templates, except for class.declass and get.class.declass.
--><!-- 2011-01-28
  - Added convenience template with mode="ism:authority" for processing current element to generate Classification Authority Block
  - Changed namespace for qualified templates to use ISM namespace, except for class.declass and get.class.declass (preserved for compatibility).
--><!-- 2011-08-12
  - Changed logic for declass exception warnings to include 25X1-HUM and 25X2-WMD
--><!-- 2013-01-02
    Removed NU since it is no longer a classification.
    Added logic for NATO and NATO-AEA as a new exemptions.

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

2.3 - IC-ISM-DOD-Rendering.xsl

```
<!-- **************************************************************** --><!--                              UNCLASSIFIED                                                     --><!--
**************************************************************** --><!-- ****************************************************************
  INTELLIGENCE COMMUNITY TECHNICAL SPECIFICATION
  XML DATA ENCODING SPECIFICATION FOR
  INFORMATION SECURITY MARKING METADATA (ISM.XML)
  ****************************************************************
  Module:   IC-ISM-ISOO-Rendering.xsl
  Creators: Office of the Director of National Intelligence
  Intelligence Community Chief Information Officer
  **************************************************************** --><!-- **************************************************************** --><!--
DESCRIPTION                           --><!--                                                                --><!-- This stylesheet renders a security banner, portion mark and      --><!--
control/decontrol block from a document's top-level ISM attribute--><!-- values. This stylesheet is to be used if a document is following --><!-- DoD policy and has either SAP markings or CUI
markings;          --><!-- if the document has CUI, it can be either pure CUI or commingled.--><!-- This stylesheet renders the metadata in a way that is compliant  --><!-- with the
Information Security Oversight Office (ISOO)            --><!-- "Marking CUI" Handbook, V1.1, Dec. 6 2016.                       --><!--
**************************************************************** --><xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                               xmlns:xs="http://www.w3.org/2001/XMLSchema"
                               xmlns:cve="urn:us:gov:ic:cve"
                               xmlns:ism-func="urn:us:gov:ic:ism:functions"
                               version="2.0">

                  <xsl:import href="IC-ISM-SecurityBanner.xsl"/>
                  <xsl:import href="IC-ISM-PortionMark.xsl"/>
                  <xsl:import href="IC-ISM-ClassDeclass.xsl"/>

                  <xsl:output method="text"
                              encoding="UTF-8"
                              media-type="text-plain"
                              indent="no"/>
                  <!-- If including this xsl causes "Content is not allowed in prolog" the importing
  XSL is likely missing an output declaration -->

  <!-- Define variable CUIRenderingRuleSet that instructs the IC-ISM stylesheets  -->
  <!-- on what rules to use for a banner, block or portion-mark that contains     -->
  <!-- CUI markings.  In this stylesheet, CUIRenderingRuleSet is set to 'DOD',     -->
  <!-- meaning that the DOD rules should be followed for CUI markings.            -->

  <xsl:param name="CUIRenderingRuleSet" select="'DOD'"/>

                  <!-- Define variable SAPRenderingRuleSet that instructs the IC-ISM stylesheets  -->
  <!-- to use DoD rules for a banner, block or portion-mark that contains SAPs.   -->

  <xsl:param name="SAPRenderingRuleSet" select="'DOD'"/>



               </xsl:stylesheet>
               <!-- **************************************************************** --><!-- **************************************************************** --><!--
UNCLASSIFIED                                                        --><!-- **************************************************************** -->
```

2.4 - IC-ISM-Functions.xsl

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                               xmlns:xs="http://www.w3.org/2001/XMLSchema"
                               xmlns:xd="http://www.oxygenxml.com/ns/doc/xsl"
                               xmlns:ism-func="urn:us:gov:ic:ism:functions"
                               xmlns:cve="urn:us:gov:ic:cve"
                               xmlns:catt="urn:us:gov:ic:taxonomy:catt:tetragraph"
                               exclude-result-prefixes="xs xd"
                               version="2.0">
                  <xd:doc scope="stylesheet">
                     <xd:desc>
                        <xd:p>
                           <xd:b>Created on:</xd:b> Sep 22, 2019</xd:p>
                        <xd:p>
                           <xd:b>Author:</xd:b>IC-CIO</xd:p>
                        <xd:p/>
                     </xd:desc>
                  </xd:doc>

                  <xsl:param name="warn-missing-classif" select="'MISSING CLASSIFICATION MARKING'"/>
                  <xsl:param name="warn-parse-classif"
                             select="'UNABLE TO DETERMINE CLASSIFICATION MARKING'"/>
                  <xsl:param name="warn-parse-ownerproducer"
                             select="concat($warn-parse-classif, ' - MISSING OWNER/PRODUCER')"/>
                  <xsl:param name="warn-parse-relto" select="'UNABLE TO DETERMINE RELEASABILITY'"/>
                  <xsl:param name="warn-parse-displayonly"
                             select="'UNABLE TO DETERMINE DISPLAY ONLY'"/>
                  <xsl:param name="warn-parse-eyes"
                             select="'UNABLE TO DETERMINE EYES ONLY MARKINGS'"/>

                  <xsl:param name="pathToISMCATCVE" select="'../../CVE/ISMCAT/'"/>
                  <xsl:param name="pathToISMCVE" select="'../../CVE/ISM/'"/>

                  <xsl:param name="pathToISMCATTaxonomy" select="'../../Taxonomy/ISMCAT/'"/>

                  <xsl:variable name="FGIOpenCVE"
                                select="document(concat($pathToISMCATCVE, 'CVEnumISMCATFGIOpen.xml'))//cve:CVE/cve:Enumeration"/>
                  <xsl:variable name="FGIProtectedCVE"
                                select="document(concat($pathToISMCATCVE, 'CVEnumISMCATFGIProtected.xml'))//cve:CVE/cve:Enumeration"/>
                  <xsl:variable name="OwnerProducerCVE"
                                select="document(concat($pathToISMCATCVE, 'CVEnumISMCATOwnerProducer.xml'))//cve:CVE/cve:Enumeration"/>
                  <xsl:variable name="RelCVE"
                                select="document(concat($pathToISMCATCVE, 'CVEnumISMCATRelTo.xml'))//cve:CVE/cve:Enumeration"/>
                  <xsl:variable name="disseminationControlsIcrmCVE"
                                select="document(concat($pathToISMCVE, 'CVEnumISMDissemIcrm.xml'))//cve:CVE/cve:Enumeration"/>
                  <xsl:variable name="disseminationControlsCUICVE"
                                select="document(concat($pathToISMCVE, 'CVEnumISMDissemCui.xml'))//cve:CVE/cve:Enumeration"/>
                  <xsl:variable name="disseminationControlsCommingledCVE"
                                select="document(concat($pathToISMCVE, 'CVEnumISMDissemCommingled.xml'))//cve:CVE/cve:Enumeration"/>

                  <!-- Only used by Rollup 2021-02-23 When Rollup embraces CUI should go away -->
    <xsl:variable name="disseminationControlsCVE"
                                select="document(concat($pathToISMCVE, 'CVEnumISMDissem.xml'))//cve:CVE/cve:Enumeration"/>
```

```
              <xsl:variable name="NonICControlsCVE"
                            select="document(concat($pathToISMCVE, 'CVEnumISMNonIC.xml'))//cve:CVE/cve:Enumeration"/>
              <xsl:variable name="AtomicEnergyCVE"
                            select="document(concat($pathToISMCVE, 'CVEnumISMAtomicEnergyMarkings.xml'))//cve:CVE/cve:Enumeration"/>
              <xsl:variable name="nonICCVE"
                            select="document(concat($pathToISMCVE, 'CVEnumISMNonIC.xml'))//cve:CVE/cve:Enumeration"/>
              <xsl:variable name="cuiBasic"
                            select="document(concat($pathToISMCVE, 'CVEnumISMCUIBasic.xml'))//cve:CVE/cve:Enumeration"/>
              <xsl:variable name="cuiSpecified"
                            select="document(concat($pathToISMCVE, 'CVEnumISMCUISpecified.xml'))//cve:CVE/cve:Enumeration"/>
              <xsl:variable name="secondBannerLineCVE"
                            select="document(concat($pathToISMCVE, 'CVEnumISMSecondBannerLine.xml'))//cve:CVE/cve:Enumeration"/>

              <!-- Regex -->
<xsl:variable name="ACCMRegex" select="'^ACCM-[A-Z0-9\-_]{1,61}$'"/>

              <!-- nonACCM values left and right of ACCM values defined in CVEnumISMNonIC.xml -->
<xsl:variable name="nonACCMLeftSet" select="'DS'"/>
              <xsl:variable name="nonACCMRightSet" select="'XD,ND,SBU,SBU-NF,LES,LES-NF,SSI,NNPI'"/>
              <xsl:variable name="nonACCMLeftSetTok" select="tokenize($nonACCMLeftSet, ',')"/>
              <xsl:variable name="nonACCMRightSetTok" select="tokenize($nonACCMRightSet, ',')"/>

              <xsl:variable name="catt"
                            select="document(concat($pathToISMCATTaxonomy,'TetragraphTaxonomyDenormalized.xml'))"/>

              <xsl:variable name="cattMappings" select="$catt//catt:Tetragraph"/>

              <xsl:variable name="decomposableTetraElems"
                            select="$cattMappings[@decomposable[. = 'Yes' or . = 'NA']]"/>

              <xsl:variable name="decomposableTetras"
                            select="$decomposableTetraElems/catt:TetraToken/text()"/>

              <xsl:function name="ism-func:classStringForClass">
                 <xsl:param name="class"/>
                 <xsl:choose>
                    <xsl:when test="$class = 'TS'">TOP SECRET</xsl:when>
                    <xsl:when test="$class = 'S'">SECRET</xsl:when>
                    <xsl:when test="$class = 'C'">CONFIDENTIAL</xsl:when>
                    <xsl:when test="$class = 'R'">RESTRICTED</xsl:when>
                    <xsl:when test="$class = 'U'">UNCLASSIFIED</xsl:when>
                    <xsl:otherwise>
                       <xsl:value-of select="$warn-parse-classif"/>
                    </xsl:otherwise>
                 </xsl:choose>
              </xsl:function>

              <!-- ********************************************************** -->
<!-- A routine for processing SCIcontrols name tokens -->
<!-- ********************************************************** -->
<xsl:function name="ism-func:get.sci">
                 <xsl:param name="all"/>
```

```
                     <!-- Loop over all the SCI tokens -->
        <xsl:for-each select="tokenize($all, ' ')">
                        <xsl:variable name="tokenizedSCIToken" select="tokenize(current(), '-')"/>
                        <xsl:variable name="compartmentLevelCount" select="count($tokenizedSCIToken) - 1"/>
                        <xsl:choose>
                <!-- Not the first SCI and has no compartment/subcompartments add a / -->
                <xsl:when test="$compartmentLevelCount = 0 and not(position() = 1)">
                              <xsl:text>/</xsl:text>
                           </xsl:when>
                           <!-- A compartment add a - -->
                <xsl:when test="$compartmentLevelCount = 1">
                              <xsl:text>-</xsl:text>
                           </xsl:when>
                           <!-- A subcompartment add a space -->
                <xsl:when test="$compartmentLevelCount = 2">
                              <xsl:text> </xsl:text>
                           </xsl:when>
                        </xsl:choose>
                        <xsl:value-of select="$tokenizedSCIToken[last()]"/>
                     </xsl:for-each>
                  </xsl:function>

                  <xsl:function name="ism-func:sciVal">
                     <xsl:param name="sci"/>
                     <xsl:param name="nonUSControls"/>

                     <xsl:if test="$sci != ''">
                        <xsl:text>//</xsl:text>
                        <xsl:value-of select="ism-func:get.sci($sci)"/>

                        <xsl:if test="$nonUSControls and contains($nonUSControls, 'BALK')">
                           <xsl:text>/BALK</xsl:text>
                        </xsl:if>

                        <xsl:if test="$nonUSControls and contains($nonUSControls, 'BOHEMIA')">
                           <xsl:text>/BOHEMIA</xsl:text>
                        </xsl:if>
                     </xsl:if>
                  </xsl:function>

                  <xsl:function name="ism-func:AEAVal">
                     <xsl:param name="atomicEnergyMarking"/>
                     <xsl:param name="nonUSControlsLocal"/>
                     <xsl:param name="banner"/>
                     <xsl:if test="normalize-space($atomicEnergyMarking) != ''">
                        <xsl:text>//</xsl:text>
                        <xsl:value-of select="ism-func:getAEA($atomicEnergyMarking, $banner)"/>
                        <xsl:if test="$nonUSControlsLocal and contains($nonUSControlsLocal, 'ATOMAL')">
                           <xsl:text>/ATOMAL</xsl:text>
                        </xsl:if>
                     </xsl:if>
                  </xsl:function>

                  <xsl:function name="ism-func:getAEA">
```

```
                 <xsl:param name="all"/>
                 <xsl:param name="banner"/>
                 <xsl:variable name="tokenizedAEA" select="tokenize($all, ' ')"/>
                 <xsl:for-each select="$tokenizedAEA">
                    <xsl:variable name="tokenizedAEAToken" select="tokenize(current(), '-')"/>
                    <xsl:variable name="compartmentLevelCount" select="count($tokenizedAEAToken) - 1"/>
                    <xsl:variable name="currentPosition" select="position()"/>
                    <xsl:choose>
            <!-- Not the first AEA and has no compartment/subcompartments add a / -->
            <xsl:when test="$compartmentLevelCount = 0 and not(position() = 1)">
                          <xsl:text>/</xsl:text>
                       </xsl:when>
                       <!-- A compartment add a - -->
            <xsl:when test="$compartmentLevelCount = 1">
                          <xsl:text>-</xsl:text>
                       </xsl:when>
                       <!-- AEA does not really have subcompartments SIGMA's look like they are subs so add space if not the first one. -->
            <xsl:when test="$compartmentLevelCount = 2">
                <!-- First Sigma add -SG then move on -->
                <xsl:if test="not(($tokenizedAEAToken[last() - 1] = 'SG') and (tokenize($tokenizedAEA[$currentPosition - 1], '-')[last() - 1] = 'SG'))">
                             <xsl:text>-</xsl:text>
                             <xsl:choose>
                                <xsl:when test="$banner">SIGMA</xsl:when>
                                <xsl:otherwise>
                                   <xsl:value-of select="$tokenizedAEAToken[last() - 1]"/>
                                </xsl:otherwise>
                             </xsl:choose>
                          </xsl:if>
                          <xsl:text> </xsl:text>
                       </xsl:when>
                    </xsl:choose>
                    <xsl:choose>
                       <xsl:when test="$banner and $tokenizedAEAToken[last()] = 'DCNI'">DOD UCNI</xsl:when>
                       <xsl:when test="$banner and $tokenizedAEAToken[last()] = 'UCNI'">DOE UCNI</xsl:when>
                       <xsl:otherwise>
                          <xsl:value-of select="$tokenizedAEAToken[last()]"/>
                       </xsl:otherwise>
                    </xsl:choose>
                 </xsl:for-each>
              </xsl:function>

              <xsl:function name="ism-func:get.sar.name">
                 <xsl:param name="name"/>
                 <xsl:sequence select="ism-func:get.sar.name($name, 'yes')"/>
              </xsl:function>

              <!-- **************************************** -->
<!-- Full name conversion for SAR name token -->
<!-- **************************************** -->
<xsl:function name="ism-func:get.sar.name">
                 <xsl:param name="name"/>
                 <xsl:param name="abbreviate"/>
                 <!-- *********************************************************************** -->
    <!-- Set this abbreviate to "yes" to use the program identifier abbreviations. -->
```

```
<!-- Otherwise the program identifiers will be used.                             -->
<!-- *********************************************************************** -->
<xsl:variable name="SAR-val">
                <xsl:choose>
                   <xsl:when test="substring-after($name, 'SAR-') = ''">
                      <xsl:value-of select="concat('SAR-', $name)"/>
                   </xsl:when>
                   <xsl:otherwise>
                      <xsl:value-of select="substring-after($name, 'SAR-')"/>
                   </xsl:otherwise>
                </xsl:choose>
             </xsl:variable>

             <xsl:choose>
    <!-- ********************************************** -->
    <!-- The actual SAR program identifiers and program -->
    <!-- identifier abbreviations should be substituted -->
    <!-- for the placeholders here.                     -->
    <!-- ********************************************** -->
    <xsl:when test="$abbreviate = 'yes'">
                   <xsl:value-of select="ism-func:translateSARname($name)"/>
                </xsl:when>
                <xsl:otherwise>
                   <xsl:choose>
                      <xsl:when test="$name = 'ABC'">ALPHA BRAVO CHARLIE</xsl:when>
                      <xsl:when test="$name = 'DEF'">DELTA ECHO FOX</xsl:when>
                      <xsl:when test="$name = 'GHI'">GULF HOTEL INDIGO</xsl:when>
                      <xsl:otherwise>
                         <xsl:value-of select="ism-func:translateSARname($name)"/>
                      </xsl:otherwise>
                   </xsl:choose>
                </xsl:otherwise>
             </xsl:choose>
          </xsl:function>

          <xd:doc>
             <xd:desc>A function that takes a SAR value of the form SAROwner:SARmarking, and extracts the SARmarking.
    The function also replaces a single underscore _ with a space, and replaces a double underscore __
    with a single underscore.  The logic uses a temporary replacement of double underscore with double tilde ~</xd:desc>
             <xd:param name="name"/>
          </xd:doc>
          <xsl:function name="ism-func:translateSARname">
             <xsl:param name="name"/>
             <xsl:choose>
                <xsl:when test="contains($name,'_') and not(contains($name,'__'))">
                   <xsl:value-of select="substring-after(replace($name, '_', ' '),':')"/>
                </xsl:when>
                <xsl:when test="contains($name,'__') and not(contains($name,'_'))">
                   <xsl:value-of select="substring-after(replace($name, '__', '_'),':')"/>
                </xsl:when>
                <xsl:when test="contains($name,'_') and contains($name,'__')">
                   <xsl:variable name="doubleunderscoresstep1" select="replace($name,'__','~~')"/>
                   <xsl:variable name="singleunderscore"
                                 select="replace($doubleunderscoresstep1,'_',' ')"/>
                   <xsl:value-of select="substring-after(replace($singleunderscore,'~~','_'),':')"/>
                </xsl:when>
                <xsl:otherwise>
                   <xsl:value-of select="substring-after($name,':')"/>
                </xsl:otherwise>
             </xsl:choose>
          </xsl:function>

          <xsl:function name="ism-func:get.secondBannerLine.name">
             <xsl:param name="token"/>
             <xsl:param name="HVCO"/>
             <xsl:choose>
                <xsl:when test="normalize-space($token) = 'HVCO'">
                   <xsl:value-of select="'HANDLE VIA '"/>
                   <xsl:value-of select="$HVCO"/>
                   <xsl:value-of select="' CHANNELS ONLY'"/>
                </xsl:when>
                <xsl:otherwise>
                   <xsl:value-of select="$secondBannerLineCVE//cve:Term[./cve:Value = $token]/cve:Description"/>
                </xsl:otherwise>
             </xsl:choose>
          </xsl:function>

          <xsl:function name="ism-func:get.relOrDisplayString">
             <xsl:param name="countryString"/>
             <xsl:value-of select="ism-func:get.relOrDisplayString($countryString, ', ')"/>
          </xsl:function>

          <xsl:function name="ism-func:get.eyesString">
             <xsl:param name="countryString"/>
             <xsl:value-of select="ism-func:get.relOrDisplayString($countryString, '/')"/>
          </xsl:function>

          <xsl:function name="ism-func:get.relOrDisplayString">
             <xsl:param name="countryString"/>
             <xsl:param name="delimiter"/>
             <xsl:variable name="countryStringWithDelimiters">
                <xsl:value-of select="string-join(tokenize($countryString, ' '), $delimiter)"/>
             </xsl:variable>
             <!-- Deal with NATO extensions like NATO:PFP or NATO:PARTNERSHIP_FOR_PEACE-->
<xsl:value-of select="translate($countryStringWithDelimiters, '_:', '  ')"/>
          </xsl:function>

          <xd:doc>
             <xd:desc>A routine for processing cuiBasic name token values</xd:desc>
             <xd:param name="all"/>
          </xd:doc>
          <xsl:function name="ism-func:get.cuiBasic">
             <xsl:param name="all"/>
             <xsl:variable name="tokenizedCuiBasic" select="tokenize($all, ' ')"/>
             <xsl:for-each select="$tokenizedCuiBasic">
                <xsl:value-of select="current()"/>
                <!-- Add a trailing / for all but the last cuiBasic marking. -->
    <xsl:if test="position() != last()">
```

```
               <xsl:text>/</xsl:text>
            </xsl:if>
         </xsl:for-each>
      </xsl:function>

      <xd:doc>
         <xd:desc>A routine for processing cuiSpecified name token values</xd:desc>
         <xd:param name="all"/>
      </xd:doc>
      <xsl:function name="ism-func:get.cuiSpecified">
         <xsl:param name="all"/>
         <xsl:variable name="tokenizedCuiSpecified" select="tokenize($all, ' ')"/>
         <xsl:for-each select="$tokenizedCuiSpecified">
            <xsl:value-of select="concat('SP-', current())"/>
            <!-- Add a trailing / for all but the last cuiSpecified marking. -->
<xsl:if test="position() != last()">
               <xsl:text>/</xsl:text>
            </xsl:if>
         </xsl:for-each>
      </xsl:function>

      <xd:doc>
         <xd:desc>A routine for processing nonIC name token values in banners and portion marks. For
banners, the BannerMapping.xml is passed with the banner and portion marking values for
markings with different banner and portion marks. For portion marks, an empty node set
is passed.</xd:desc>
         <xd:param name="all"/>
      </xd:doc>
      <xsl:function name="ism-func:get.nonic">
         <xsl:param name="all"/>
         <xsl:param name="DissemLookup"/>
         <xsl:variable name="tokenizedNonic" select="tokenize($all, ' ')"/>
         <xsl:variable name="firstACCMValue"
                       select="$tokenizedNonic[starts-with(., 'ACCM-')][1]"/>
         <xsl:for-each select="$tokenizedNonic">
            <xsl:choose>
               <xsl:when test="$DissemLookup//BannerMap[@portion = current()]">
                  <xsl:value-of select="$DissemLookup//BannerMap[@portion = current()]/text()"/>
               </xsl:when>
               <xsl:when test="starts-with(current(), 'ACCM-')">
        <!-- Remove ACCM prefix from ACCM tokens -->
        <xsl:variable name="prefixlessACCM" select="substring-after(current(), 'ACCM-')"/>
                  <!-- Replace '_' with ' ' -->
        <xsl:if test="current() = $firstACCMValue">
                     <xsl:text>ACCM-</xsl:text>
                  </xsl:if>
                  <xsl:value-of select="translate($prefixlessACCM, '_', ' ')"/>
               </xsl:when>
               <xsl:otherwise>
                  <xsl:value-of select="current()"/>
               </xsl:otherwise>
            </xsl:choose>
            <!-- Add a trailing / for all but the last non-ic dissem control. -->
<xsl:if test="position() != last()">
```

```
           <xsl:text>/</xsl:text>
        </xsl:if>
     </xsl:for-each>
  </xsl:function>

  <xd:doc>
     <xd:desc>
        <xd:p>Returns a sort position for a given value passed based in the position in a given
CVE</xd:p>
     </xd:desc>
     <xd:param name="value"/>
     <xd:param name="cveToSortWith"/>
  </xd:doc>
  <xsl:function name="ism-func:cveSortOrder" as="xs:integer">
     <xsl:param name="value"/>
     <xsl:param name="cveToSortWith" as="node()*"/>
     <xsl:variable name="returnValue" as="xs:integer">
        <xsl:value-of select="count($cveToSortWith//cve:Term[$value = cve:Value]/preceding-sibling::*)"/>
     </xsl:variable>
     <xsl:value-of select="$returnValue"/>
  </xsl:function>

  <xd:doc>
     <xd:desc>
        <xd:p>Checks if the value is an empty string, if not calls ism-func:cveSortOrderJoin if
empty returns empty string.</xd:p>
     </xd:desc>
     <xd:param name="values"/>
     <xd:param name="cveToSortWith"/>
  </xd:doc>
  <xsl:function name="ism-func:cveSortOrderJoinWithEmptyCheck">
     <xsl:param name="values"/>
     <xsl:param name="cveToSortWith" as="node()*"/>
     <xsl:choose>
        <xsl:when test="normalize-space($values) = ''">
           <xsl:value-of select="normalize-space($values)"/>
        </xsl:when>
        <xsl:otherwise>
           <xsl:variable name="tokens"
                         select="xs:NMTOKENS(normalize-space($values))"
                         as="xs:NMTOKEN*"/>
           <xsl:value-of select="ism-func:cveSortOrderJoin($tokens, $cveToSortWith)"/>
        </xsl:otherwise>
     </xsl:choose>
  </xsl:function>

  <xd:doc>
     <xd:desc>
        <xd:p>Sorts multiple values according to a CVE passed in and returns the sorted set,
removing duplicates.</xd:p>
     </xd:desc>
     <xd:param name="values"/>
     <xd:param name="cveToSortWith"/>
  </xd:doc>
```

```
      <xsl:function name="ism-func:cveSortOrderJoin">
         <xsl:param name="values" as="xs:anyAtomicType*"/>
         <xsl:param name="cveToSortWith" as="node()*"/>
         <xsl:variable name="SortedValues" as="xs:NMTOKEN*">
            <xsl:perform-sort select="$values">
               <xsl:sort select="ism-func:cveSortOrder(., $cveToSortWith)" data-type="number"/>
            </xsl:perform-sort>
         </xsl:variable>
         <xsl:value-of select="ism-func:join(distinct-values($SortedValues))"/>
      </xsl:function>

      <xd:doc>
         <xd:desc> Returns the given sequence of $values joined into a normalized single string </xd:desc>
         <xd:param name="values"/>
      </xd:doc>
      <xsl:function name="ism-func:join" as="xs:string">
         <xsl:param name="values" as="xs:anyAtomicType*"/>

         <xsl:sequence select="normalize-space(string-join($values, ' '))"/>
      </xsl:function>

      <xd:doc>
         <xd:desc> Return a list of values as a space delimited string from a sequence of tokens that
only matches the regex </xd:desc>
         <xd:param name="attrValues"/>
         <xd:param name="regex"/>
      </xd:doc>
      <xsl:function name="ism-func:getStringFromSequenceWithOnlyRegexValues" as="xs:string">
         <xsl:param name="attrValues"/>
         <xsl:param name="regex"/>
         <xsl:variable name="StringWithOnlyRegexValues">
            <xsl:for-each select="$attrValues">
    <!-- if value does match the regex, return that value -->
    <xsl:if test="matches(current(), $regex)">
                  <xsl:value-of select="current()"/>
               </xsl:if>
               <xsl:value-of select="' '"/>
            </xsl:for-each>
         </xsl:variable>
         <xsl:value-of select="normalize-space(string($StringWithOnlyRegexValues))"/>
      </xsl:function>

      <xd:doc>
         <xd:desc> Return a list of values as a space delimited string from a sequence of tokens that
filters out anything matching the regex </xd:desc>
         <xd:param name="attrValues"/>
         <xd:param name="regex"/>
      </xd:doc>
      <xsl:function name="ism-func:getStringFromSequenceWithoutRegexValues" as="xs:string">
         <xsl:param name="attrValues" as="xs:string*"/>
         <xsl:param name="regex"/>
         <xsl:variable name="StringWithoutRegexValues">
            <xsl:for-each select="$attrValues">
    <!-- if value does not match the regex, return that value -->
```

```
<xsl:if test="not(matches(current(), $regex))">
              <xsl:value-of select="current()"/>
           </xsl:if>
           <xsl:value-of select="' '"/>
        </xsl:for-each>
     </xsl:variable>
     <xsl:value-of select="normalize-space(string($StringWithoutRegexValues))"/>
  </xsl:function>

  <xd:doc>
     <xd:desc>for running xspec tests and other debug</xd:desc>
     <xd:param name="attrValues"/>
  </xd:doc>
  <xsl:function name="ism-func:cveSortOrderDebug" as="xs:string">
     <xsl:param name="attrValues"/>
     <xsl:variable name="Sorted">
        <xsl:for-each select="$attrValues">
           <xsl:value-of select="current()"/>
           <xsl:text>:</xsl:text>
           <xsl:value-of select="xs:integer(ism-func:cveSortOrder(., $RelCVE))"/>
           <xsl:text>|</xsl:text>
        </xsl:for-each>
     </xsl:variable>
     <xsl:value-of select="$Sorted"/>
  </xsl:function>

  <xd:doc>
     <xd:desc>
        <xd:p>Sorts values from an ism:ownerProducer attribute to the right order for
banner/portion rendering.</xd:p>
     </xd:desc>
     <xd:param name="values"/>
  </xd:doc>
  <xsl:function name="ism-func:sortOwnerProducer">
     <xsl:param name="values"/>
     <xsl:value-of select="ism-func:sortCountryAndTetraWithEmptyCheck($values, $OwnerProducerCVE)"/>
  </xsl:function>

  <xd:doc>
     <xd:desc>
        <xd:p>Sorts values from an ism:sciControls attribute to the right order for
banner/portion rendering.</xd:p>
     </xd:desc>
     <xd:param name="values"/>
  </xd:doc>
  <xsl:function name="ism-func:sortSciControls">
     <xsl:param name="values"/>
     <xsl:value-of select="ism-func:sortJoin($values)"/>
  </xsl:function>

  <xd:doc>
     <xd:desc>
        <xd:p>Sorts values from an ism:sar attribute to the right order for banner/portion
rendering.</xd:p>
```

```
     </xd:desc>
     <xd:param name="values"/>
  </xd:doc>
  <xsl:function name="ism-func:sortSar">
     <xsl:param name="values"/>
     <xsl:value-of select="ism-func:sortJoin($values)"/>
  </xsl:function>

  <xd:doc>
     <xd:desc>
        <xd:p>Sorts values from an ism:atomicenergymarkings attribute to the right order for
banner/portion rendering.</xd:p>
     </xd:desc>
     <xd:param name="values"/>
  </xd:doc>
  <xsl:function name="ism-func:sortAtomicenergymarkings">
     <xsl:param name="values"/>
     <xsl:value-of select="ism-func:cveSortOrderJoinWithEmptyCheck($values, $AtomicEnergyCVE)"/>
  </xsl:function>

  <xd:doc>
     <xd:desc>
        <xd:p>Sorts values from an ism:fgiopen attribute to the right order for banner/portion
rendering.</xd:p>
     </xd:desc>
     <xd:param name="values"/>
  </xd:doc>
  <xsl:function name="ism-func:sortFGIOpen">
     <xsl:param name="values"/>
     <xsl:value-of select="ism-func:sortCountryAndTetraWithEmptyCheck($values, $FGIOpenCVE)"/>
  </xsl:function>

  <xd:doc>
     <xd:desc>
        <xd:p>Sorts values from an ism:fgiProtected attribute to the right order for
banner/portion rendering.</xd:p>
     </xd:desc>
     <xd:param name="values"/>
  </xd:doc>
  <xsl:function name="ism-func:sortFGIProtected">
     <xsl:param name="values"/>
     <xsl:value-of select="ism-func:sortCountryAndTetraWithEmptyCheck($values, $FGIProtectedCVE)"/>
  </xsl:function>

  <xd:doc>
     <xd:desc>
        <xd:p>Sorts values from an ism:dissemControls attribute to the right order for
banner/portion rendering.</xd:p>
        <xd:p>Only used by Rollup 2021-02-23 When Rollup embraces CUI should go away.</xd:p>
     </xd:desc>
     <xd:param name="values"/>
  </xd:doc>
  <xsl:function name="ism-func:sortDissemControlsPreCUI">
     <xsl:param name="values"/>
```

```
             <xsl:value-of select="ism-func:cveSortOrderJoinWithEmptyCheck($values, $disseminationControlsCVE)"/>
          </xsl:function>

          <xd:doc>
             <xd:desc>
                <xd:p>Determines the right set of allowed ism:dissemControls values based on
        ism:compliesWith. Sorts values from the ism:dissemControls attribute to the right
        order for banner/portion rendering.</xd:p>
             </xd:desc>
             <xd:param name="values"/>
             <xd:param name="compliesWith"/>
             <xd:param name="CUIandICcontrolMarkings"/>
          </xd:doc>
          <xsl:function name="ism-func:sortDissemControls">
             <xsl:param name="values"/>
             <xsl:param name="compliesWith"/>
             <xsl:param name="CUIandICcontrolMarkings"/>
             <xsl:variable name="disseminationControlsCVE">
                <xsl:choose>
                   <xsl:when test="$compliesWith = 'USA-CUI-ONLY'">
                      <xsl:copy-of select="$disseminationControlsCUICVE"/>
                   </xsl:when>
                   <xsl:when test="contains($compliesWith, 'USA-CUI')">
                      <xsl:choose>
                         <xsl:when test="$CUIandICcontrolMarkings = false()">
                            <xsl:copy-of select="$disseminationControlsCUICVE"/>
                         </xsl:when>
                         <xsl:otherwise>
                            <xsl:copy-of select="$disseminationControlsCommingledCVE"/>
                         </xsl:otherwise>
                      </xsl:choose>
                   </xsl:when>
                   <xsl:otherwise>
                      <xsl:copy-of select="$disseminationControlsIcrmCVE"/>
                   </xsl:otherwise>
                </xsl:choose>
             </xsl:variable>
             <xsl:value-of select="ism-func:cveSortOrderJoinWithEmptyCheck($values, $disseminationControlsCVE)"/>
          </xsl:function>

          <xd:doc>
             <xd:desc>Determine the set of dissemination controls that are NOT CUI limited dissem
    controls. This variable is used to determine whether to use UNCLASSIFIED or CUI in the
    banner or portion mark of an UNCLASSIFIED commingled document. If any of the
    dissemination controls is NOT one of the CUI limited dissem controls (i.e., it is a
    dissem control from the IC Markings Register and Manual that is NOT also one of the CUI
    limited dissem controls), then use UNCLASSIFIED rather than CUI at the start of the
    banner or portion mark.</xd:desc>
             <xd:param name="values"/>
          </xd:doc>
          <!--   <xsl:function name="ism-func:get.dissemNotCUI">
<xsl:param name="values"/>
<xsl:variable name="CUIdissems">
    <xsl:for-each select="$disseminationControlsCUICVE/cve:Term/cve:Value">
```

```
            <xsl:choose>
                <xsl:when test="current() = 'NOFORN'">
                    <xsl:value-of select="'NF'"/>
                </xsl:when>
                <xsl:otherwise>
                    <xsl:value-of select="current()"/>
                </xsl:otherwise>
            </xsl:choose>
            <xsl:if test="position() != last()">
                <xsl:text> </xsl:text>
            </xsl:if>
        </xsl:for-each>
    </xsl:variable>
    <xsl:variable name="dissemNotCui">
        <xsl:if test="$values != ''">
            <xsl:for-each select="tokenize($values, ' ')">
                <xsl:if test="not(contains($CUIdissems, current()))">
                    <xsl:value-of select="current()"/>
                </xsl:if>
            </xsl:for-each>
        </xsl:if>
    </xsl:variable>
    <xsl:value-of select="$dissemNotCui"/>
</xsl:function> -->

<xsl:function name="ism-func:get.dissemNotCUI">
                 <xsl:param name="values"/>
                 <xsl:variable name="tokenizedDissems" select="tokenize($values, ' ')"/>
                 <xsl:variable name="dissemNotCui">
                    <xsl:for-each select="$tokenizedDissems">
                       <xsl:if test="not($disseminationControlsCUICVE/cve:Term[cve:Value = current()])">
                          <xsl:if test="position() != 1">
                             <xsl:text> </xsl:text>
                          </xsl:if>
                          <xsl:value-of select="current()"/>
                       </xsl:if>
                    </xsl:for-each>
                 </xsl:variable>
                 <xsl:value-of select="normalize-space($dissemNotCui)"/>
              </xsl:function>

              <xd:doc>
                 <xd:desc>Determine the set of dissemination controls that ARE CUI limited dissem controls.
        This variable is used to generate a CUI dissems line in a CUI Control Block for DoD
        uses. </xd:desc>
                 <xd:param name="values"/>
              </xd:doc>
              <xsl:function name="ism-func:get.dissemCUI">
                 <xsl:param name="values"/>
                 <xsl:variable name="tokenizedDissems" select="tokenize($values, ' ')"/>
                 <xsl:variable name="dissemCui">
                    <xsl:for-each select="$tokenizedDissems">
                       <xsl:if test="$disseminationControlsCUICVE/cve:Term[cve:Value = current()]">
                          <xsl:if test="position() != 1">
```

```
                        <xsl:text> </xsl:text>
                     </xsl:if>
                     <xsl:value-of select="current()"/>
                  </xsl:if>
               </xsl:for-each>
            </xsl:variable>
            <xsl:value-of select="normalize-space($dissemCui)"/>
         </xsl:function>

         <xd:doc>
            <xd:desc>
               <xd:p>Sorts values from an ism:releaseto attribute to the right order for banner/portion
rendering.</xd:p>
            </xd:desc>
            <xd:param name="values"/>
         </xd:doc>
         <xsl:function name="ism-func:sortReleaseto">
            <xsl:param name="values"/>
            <xsl:value-of select="ism-func:sortCountryAndTetraWithEmptyCheck($values, $RelCVE)"/>
         </xsl:function>

         <xd:doc>
            <xd:desc>
               <xd:p>Sorts values from an ism:displayonly attribute to the right order for
banner/portion rendering.</xd:p>
            </xd:desc>
            <xd:param name="values"/>
         </xd:doc>
         <xsl:function name="ism-func:sortDisplayonly">
            <xsl:param name="values"/>
            <xsl:value-of select="ism-func:sortCountryAndTetraWithEmptyCheck($values, $RelCVE)"/>
         </xsl:function>

         <xd:doc>
            <xd:desc>
               <xd:p>Sorts values from an ism:nonic attribute to the right order for banner/portion
rendering.</xd:p>
            </xd:desc>
            <xd:param name="values"/>
         </xd:doc>
         <xsl:function name="ism-func:sortNonic">
            <xsl:param name="values"/>
            <xsl:value-of select="ism-func:sortNonICWithEmptyCheck($values)"/>
         </xsl:function>

         <xd:doc>
            <xd:desc>
               <xd:p>Sorts values from an ism:cuiBasic attribute alphabetically for banner/portion
rendering.</xd:p>
            </xd:desc>
            <xd:param name="values"/>
         </xd:doc>
         <xsl:function name="ism-func:sortCuiBasic">
            <xsl:param name="values"/>
            <xsl:value-of select="ism-func:sortJoin($values)"/>
         </xsl:function>

         <xd:doc>
            <xd:desc>
               <xd:p>Sorts values from an ism:cuiSpecified attribute alphabetically for banner/portion
       rendering.</xd:p>
            </xd:desc>
            <xd:param name="values"/>
         </xd:doc>
         <xsl:function name="ism-func:sortCuiSpecified">
            <xsl:param name="values"/>
            <xsl:value-of select="ism-func:sortJoin($values)"/>
         </xsl:function>

         <xd:doc>
            <xd:desc>
               <xd:p>Sorts values from an ism:secondBannerLine attribute alphabetically for
       banner/portion rendering.</xd:p>
            </xd:desc>
            <xd:param name="values"/>
         </xd:doc>
         <xsl:function name="ism-func:sortSecondBannerLine">
            <xsl:param name="values"/>
            <xsl:value-of select="ism-func:sortJoin($values)"/>
         </xsl:function>

         <xd:doc>
            <xd:desc>
               <xd:p>Checks if the value is an empty string, if not calls ism-func:sortCountryAndTetra
       if empty returns empty string.</xd:p>
            </xd:desc>
            <xd:param name="values"/>
            <xd:param name="cveToSortWith"/>
         </xd:doc>
         <xsl:function name="ism-func:sortCountryAndTetraWithEmptyCheck">
            <xsl:param name="values"/>
            <xsl:param name="cveToSortWith" as="node()*"/>
            <xsl:choose>
               <xsl:when test="normalize-space($values) = ''">
                  <xsl:value-of select="normalize-space($values)"/>
               </xsl:when>
               <xsl:otherwise>
                  <xsl:variable name="tokens"
                                select="xs:NMTOKENS(normalize-space($values))"
                                as="xs:NMTOKEN*"/>
                  <xsl:value-of select="ism-func:sortCountryAndTetra($tokens, $cveToSortWith)"/>
               </xsl:otherwise>
            </xsl:choose>
         </xsl:function>

         <xd:doc>
            <xd:desc> CVEnumISMCATFGIOpen, CVEnumISMCATFGIProtected, CVEnumISMCATOwnerProducer, and
   CVEnumISMCATRelTo need special sorting to account for NATO Nacs. All tokens up to and
```

```
including NATO go according to the CVE order All NATO:xxx go alphabetical. After any
NATO:xxx the remainder go in CVE order. </xd:desc>
         <xd:param name="attrValues"/>
         <xd:param name="cveToSortWith">
            <xd:p>The contents of the CVE that should be used for sorting.</xd:p>
         </xd:param>
      </xd:doc>
      <xsl:function name="ism-func:sortCountryAndTetra" as="xs:string">
         <xsl:param name="attrValues"/>
         <xsl:param name="cveToSortWith"/>
         <xsl:variable name="regex" select="'NATO:'"/>
         <xsl:variable name="sortedTokens" as="xs:string*">
            <xsl:variable name="beforeDistinct" as="xs:string*">
               <xsl:perform-sort select="$attrValues">
                  <xsl:sort select="xs:integer(ism-func:cveSortOrder(., $cveToSortWith))"
                            data-type="number"/>
               </xsl:perform-sort>
            </xsl:variable>
            <xsl:sequence select="distinct-values($beforeDistinct)"/>
         </xsl:variable>
         <xsl:variable name="withoutRegexValues" as="xs:NMTOKEN*">
            <xsl:variable name="StringWithout"
                          select="ism-func:getStringFromSequenceWithoutRegexValues($sortedTokens, $regex)"/>
            <xsl:choose>
               <xsl:when test="normalize-space($StringWithout) != ''">
                  <xsl:sequence select="xs:NMTOKENS(normalize-space($StringWithout))"/>
               </xsl:when>
               <xsl:otherwise/>
            </xsl:choose>
         </xsl:variable>
         <xsl:choose>
            <xsl:when test="count($sortedTokens) = count($withoutRegexValues)">
               <xsl:value-of select="ism-func:join($sortedTokens)"/>
            </xsl:when>
            <xsl:otherwise>
               <xsl:variable name="nacSortOrder"
                             select="ism-func:cveSortOrder('NATO', $cveToSortWith)"/>
               <xsl:variable name="NacValues" as="xs:NMTOKEN*">
                  <xsl:perform-sort select="xs:NMTOKENS(ism-func:getStringFromSequenceWithOnlyRegexValues($attrValues, $regex))">
                     <xsl:sort select="." order="ascending"/>
                  </xsl:perform-sort>
               </xsl:variable>
               <xsl:variable name="SortedValuesWithoutNac" as="xs:NMTOKEN*">
                  <xsl:perform-sort select="$withoutRegexValues">
                     <xsl:sort select="ism-func:cveSortOrder(., $cveToSortWith)" data-type="number"/>
                  </xsl:perform-sort>
               </xsl:variable>
               <xsl:variable name="beforeNAC"
                             select="$SortedValuesWithoutNac[ism-func:cveSortOrder(., $cveToSortWith) &lt; $nacSortOrder]"
                             as="xs:NMTOKEN*"/>
               <xsl:variable name="afterNAC"
                             select="$SortedValuesWithoutNac[ism-func:cveSortOrder(., $cveToSortWith) &gt; $nacSortOrder]"
                             as="xs:NMTOKEN*"/>
               <xsl:value-of select="($beforeNAC, $NacValues, $afterNAC)"/>
            </xsl:otherwise>
         </xsl:choose>
      </xsl:function>

      <xd:doc>
         <xd:desc>
            <xd:p>Checks if the value is an empty string, if not calls ism-func:sortNonIC if empty
    returns empty string.</xd:p>
         </xd:desc>
         <xd:param name="values"/>
      </xd:doc>
      <xsl:function name="ism-func:sortNonICWithEmptyCheck">
         <xsl:param name="values"/>
         <xsl:choose>
            <xsl:when test="normalize-space($values) = ''">
               <xsl:value-of select="normalize-space($values)"/>
            </xsl:when>
            <xsl:otherwise>
               <xsl:variable name="tokens"
                             select="xs:NMTOKENS(normalize-space($values))"
                             as="xs:NMTOKEN*"/>
               <xsl:value-of select="ism-func:sortNonIC($tokens)"/>
            </xsl:otherwise>
         </xsl:choose>
      </xsl:function>

      <xd:doc>
         <xd:desc> CVEnumISMNonIC needs special sorting to account for ACCMs. The order is DS if
present followed by all the ACCM-xxx tokens in alphabetical order followed by the
remaining CVEnumISMNonIC values. </xd:desc>
         <xd:param name="attrValues"/>
      </xd:doc>
      <xsl:function name="ism-func:sortNonIC" as="xs:string">
         <xsl:param name="attrValues"/>
         <xsl:variable name="regex" select="'ACCM-'"/>
         <xsl:variable name="sortedTokens" as="xs:string*">
            <xsl:variable name="beforeDistinct" as="xs:string*">
               <xsl:perform-sort select="$attrValues">
                  <xsl:sort select="xs:integer(ism-func:cveSortOrder(., $NonICControlsCVE))"
                            data-type="number"/>
               </xsl:perform-sort>
            </xsl:variable>
            <xsl:sequence select="distinct-values($beforeDistinct)"/>
         </xsl:variable>
         <xsl:variable name="withoutRegexValues" as="xs:NMTOKEN*">
            <xsl:variable name="StringWithout"
                          select="ism-func:getStringFromSequenceWithoutRegexValues($sortedTokens, $regex)"/>
            <xsl:choose>
               <xsl:when test="normalize-space($StringWithout) != ''">
                  <xsl:sequence select="xs:NMTOKENS(normalize-space($StringWithout))"/>
               </xsl:when>
               <xsl:otherwise/>
            </xsl:choose>
         </xsl:variable>
         <xsl:choose>
            <xsl:when test="count($sortedTokens) = count($withoutRegexValues)">
               <xsl:value-of select="ism-func:join($sortedTokens)"/>
            </xsl:when>
            <xsl:otherwise>
               <xsl:variable name="ds-SortOrder"
                             select="ism-func:cveSortOrder('DS', $NonICControlsCVE)"/>
               <xsl:variable name="accmValues" as="xs:NMTOKEN*">
                  <xsl:perform-sort select="xs:NMTOKENS(ism-func:getStringFromSequenceWithOnlyRegexValues($attrValues, $regex))">
                     <xsl:sort select="." order="ascending"/>
                  </xsl:perform-sort>
               </xsl:variable>
               <xsl:variable name="SortedValuesWithoutACCM" as="xs:NMTOKEN*">
                  <xsl:perform-sort select="$withoutRegexValues">
                     <xsl:sort select="ism-func:cveSortOrder(., $NonICControlsCVE)" data-type="number"/>
                  </xsl:perform-sort>
               </xsl:variable>
               <xsl:variable name="beforeAccm"
                             select="$SortedValuesWithoutACCM[ism-func:cveSortOrder(., $NonICControlsCVE) &lt;= $ds-SortOrder]"
                             as="xs:NMTOKEN*"/>
               <xsl:variable name="afterACCM"
                             select="$SortedValuesWithoutACCM[ism-func:cveSortOrder(., $NonICControlsCVE) &gt; $ds-SortOrder]"
                             as="xs:NMTOKEN*"/>
               <xsl:value-of select="($beforeAccm, $accmValues, $afterACCM)"/>
            </xsl:otherwise>
         </xsl:choose>
      </xsl:function>

      <xd:doc>
         <xd:desc>
            <xd:p>Sorts values alphabetically and string joins with spaces.</xd:p>
         </xd:desc>
         <xd:param name="values"/>
      </xd:doc>
      <xsl:function name="ism-func:sortJoin">
         <xsl:param name="values"/>
         <xsl:variable name="tokens">
            <xsl:perform-sort select="tokenize(normalize-space($values), ' ')">
               <xsl:sort select="." order="ascending"/>
            </xsl:perform-sort>
         </xsl:variable>
         <xsl:value-of select="string-join($tokens, ' ')"/>
      </xsl:function>

      <xd:doc>
         <xd:desc> Recursively remove all decomposable tetragraphs in the given $relTo string
and replace them with their constituent countries. Note: Does not include USA </xd:desc>
         <xd:param name="relTo"/>
      </xd:doc>
      <xsl:function name="ism-func:expandDecomposableTetras" as="xs:string*">
         <xsl:param name="relTo" as="xs:string"/>

         <xsl:variable name="expandedTetras">
            <xsl:choose>
```

```
                   <xsl:when test="ism-func:containsDecomposableTetra($relTo)">
                      <xsl:variable name="currTetra"
                                    select="ism-func:tokenize($relTo)[. = $decomposableTetras][1]"/>
                      <xsl:variable name="currTetraCountries"
                                    select="ism-func:join(ism-func:getCountriesForTetra($currTetra))"/>
                      <xsl:variable name="expandCurrTetra"
                                    select="replace(ism-func:padValue($relTo), ism-func:padValue($currTetra), ism-func:padValue($currTetraCountries))"/>

                      <xsl:value-of select="ism-func:expandDecomposableTetras($expandCurrTetra)"/>
                   </xsl:when>

                   <xsl:otherwise>
                      <xsl:value-of select="normalize-space($relTo)"/>
                   </xsl:otherwise>
                </xsl:choose>
             </xsl:variable>

             <xsl:sequence select="distinct-values(ism-func:tokenize($expandedTetras))[. != 'USA']"/>
          </xsl:function>

          <xd:doc>
             <xd:desc> Returns true if the given $relTo string (e.g. 'USA CAN GBR') contains any
    tetragraphs that can be decomposed into its constituent countries  </xd:desc>
             <xd:param name="relTo"/>
          </xd:doc>
          <xsl:function name="ism-func:containsDecomposableTetra" as="xs:boolean">
             <xsl:param name="relTo" as="xs:string?"/>

             <xsl:sequence select="normalize-space($relTo) and ism-func:containsAnyOfTheTokens($relTo, $decomposableTetras)"/>
          </xsl:function>

          <xd:doc>
             <xd:desc>
    Returns true if any token in the attribute value matches at least one token in the provided list.
</xd:desc>
             <xd:param name="attribute"/>
             <xd:param name="tokenList"/>
          </xd:doc>
          <xsl:function name="ism-func:containsAnyOfTheTokens" as="xs:boolean">
             <xsl:param name="attribute"/>
             <xsl:param name="tokenList" as="xs:string*"/>
             <xsl:sequence select="some $attrToken in tokenize(normalize-space(string($attribute)), ' ') satisfies $attrToken = $tokenList"/>
          </xsl:function>

          <xd:doc>
             <xd:desc> Returns the sequence of country codes that correspond to the given $tetra </xd:desc>
             <xd:param name="tetra"/>
          </xd:doc>
          <xsl:function name="ism-func:getCountriesForTetra" as="xs:string*">
             <xsl:param name="tetra" as="xs:string"/>

             <xsl:sequence select="$decomposableTetraElems[catt:TetraToken/text() = $tetra]/catt:Membership/*/text()"/>
          </xsl:function>
```

```
   <xd:doc>
      <xd:desc> Returns normalized $value with a preceding and subsequent space (' ') character </xd:desc>
      <xd:param name="value"/>
   </xd:doc>
   <xsl:function name="ism-func:padValue" as="xs:string">
      <xsl:param name="value" as="xs:string?"/>

      <xsl:value-of select="concat(' ', normalize-space($value), ' ')"/>
   </xsl:function>

   <xd:doc>
      <xd:desc> Returns the given $value with its values broken into tokens using whitespace as delimiters </xd:desc>
      <xd:param name="value"/>
   </xd:doc>
   <xsl:function name="ism-func:tokenize" as="xs:string*">
      <xsl:param name="value" as="xs:string?"/>

      <xsl:sequence select="tokenize(normalize-space($value), ' ')"/>
   </xsl:function>

</xsl:stylesheet>
```

2.5 - IC-ISM-ISOO-Rendering.xsl

```
<!-- ****************************************************************** --><!--                                   UNCLASSIFIED                                                      --><!--
****************************************************************** --><!-- ******************************************************************
  INTELLIGENCE COMMUNITY TECHNICAL SPECIFICATION
  XML DATA ENCODING SPECIFICATION FOR
  INFORMATION SECURITY MARKING METADATA (ISM.XML)
  ******************************************************************
  Module:   IC-ISM-ISOO-Rendering.xsl
  Creators: Office of the Director of National Intelligence
  Intelligence Community Chief Information Officer
  ****************************************************************** --><!-- ****************************************************************** --><!--
DESCRIPTION                           --><!--                                                                        --><!-- This stylesheet renders a security banner, portion mark and      --><!--
control/decontrol block from a document's top-level ISM attribute--><!-- values. This stylesheet is to be used if a document has CUI      --><!-- markings; the document can be either pure CUI
or commingled.     --><!-- This stylesheet renders the metadata in a way that is compliant  --><!-- with the Information Security Oversight Office (ISOO)           --><!-- "Marking CUI"
Handbook, V1.1, Dec. 6 2016.                    --><!-- ****************************************************************** --><xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                               xmlns:xs="http://www.w3.org/2001/XMLSchema"
                               xmlns:cve="urn:us:gov:ic:cve"
                               xmlns:ism-func="urn:us:gov:ic:ism:functions"
                               version="2.0">

                  <xsl:import href="IC-ISM-SecurityBanner.xsl"/>
                  <xsl:import href="IC-ISM-PortionMark.xsl"/>
                  <xsl:import href="IC-ISM-ClassDeclass.xsl"/>

                  <xsl:output method="text"
                              encoding="UTF-8"
                              media-type="text-plain"
                              indent="no"/>
                  <!-- If including this xsl causes "Content is not allowed in prolog" the importing
  XSL is likely missing an output declaration -->

  <!-- Define variable CUIRenderingRuleSet that instructs the IC-ISM stylesheets  -->
  <!-- on what rules to use for a banner, block or portion-mark that contains     -->
  <!-- CUI markings.  In this stylesheet, CUIRenderingRuleSet is set to 'ISOO',   -->
  <!-- meaning that the ISOO rules in the CUI Marking Handbook should be followed.-->

  <xsl:param name="CUIRenderingRuleSet" select="'ISOO'"/>




               </xsl:stylesheet>
               <!-- ****************************************************************** --><!-- ****************************************************************** --><!--
UNCLASSIFIED                                                      --><!-- ****************************************************************** -->
```

2.6 - IC-ISM-PortionMark.xsl

```
<!-- **************************************************************** --><!--                           UNCLASSIFIED                         --><!--
**************************************************************** --><!-- ****************************************************************
  INTELLIGENCE COMMUNITY TECHNICAL SPECIFICATION
  XML DATA ENCODING SPECIFICATION FOR
  INFORMATION SECURITY MARKING METADATA (ISM.XML)
  ****************************************************************
  Module:   IC-ISM-PortionMark.xsl
  Creators: Office of the Director of National Intelligence
  Intelligence Community Chief Information Officer
  **************************************************************** --><!-- **************************************************************** --><!--
INTRODUCTION                          --><!-- **************************************************************** --><!-- This XSLT file is one component of the ISM.XML Data Encoding
  Specification (DES). Please see the document titled
  XML DATA ENCODING SPECIFICATION FOR INFORMATION SECURITY MARKING METADATA
  for a complete description of the encoding as well as list
  of all components.

  It is envisioned that this XSLT or its components, as well as other
  parts of the DES may be overridden for localized implementations.
  Therefore, permission to use, copy, modify and distribute this XSLT
  and the other parts of the DES for any purpose is hereby
  granted in perpetuity.

  Please reference the preceding two paragraphs in all copies or
  variations. The developers make no representation about the
  suitability of the schema or DES for any purpose. It is provided
  "as is" without expressed or implied warranty.

  If you modify this XSLT in any way label it
  as a variant of ISM.XML.

  Please direct all questions, bug reports,or suggestions for changes
  to the points of contact identified in the document referenced above.
--><!-- **************************************************************** --><!--                             DESCRIPTION                          --
><!--                                                                  --><!-- This XSLT 2 stylesheet renders a portion marking from the        --><!-- ISM attribute values of a portion-level
element.  The rendered   --><!-- marking is compliant with the IC Register and Manual guidelines  --><!-- as of the 2019-AUG release of the manual                         --><!--
**************************************************************** --><xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                               xmlns:xs="http://www.w3.org/2001/XMLSchema"
                               xmlns:ism-func="urn:us:gov:ic:ism:functions"
                               version="2.0">

                  <xsl:import href="IC-ISM-Functions.xsl"/>

                  <xsl:output method="text"
                              encoding="UTF-8"
                              media-type="text-plain"
                              indent="no"/>
                  <!-- If including this xsl causes "Content is not allowed in prolog" the importing
  XSL is likely missing an output declaration -->

  <xsl:param name="warn-missing-classif" select="'MISSING CLASSIFICATION MARKING'"/>
                  <xsl:param name="warn-parse-classif"
                             select="'UNABLE TO DETERMINE CLASSIFICATION MARKING'"/>
```

```
                <xsl:param name="warn-parse-ownerproducer"
                           select="concat($warn-parse-classif, ' - MISSING OWNER/PRODUCER')"/>
                <xsl:param name="warn-parse-relto" select="'UNABLE TO DETERMINE RELEASABILITY'"/>
                <xsl:param name="warn-parse-displayonly"
                           select="'UNABLE TO DETERMINE DISPLAY ONLY'"/>
                <xsl:param name="warn-parse-eyes"
                           select="'UNABLE TO DETERMINE EYES ONLY MARKINGS'"/>

                <xsl:param name="CUIRenderingRuleSet" select="''"/>
                <xsl:param name="SAPRenderingRuleSet" select="''"/>

                <!--***********************************************-->
<!-- Mode for generating the CAPCO portion mark-->
<!--***********************************************-->
<xsl:template match="*[@ism:*]" mode="ism:portionmark">
                   <xsl:param name="overalldissem"/>
                   <xsl:param name="overallreleaseto"/>
                   <xsl:call-template name="get.portionmark">
                      <xsl:with-param name="overalldissem" select="$overalldissem"/>
                      <xsl:with-param name="overallreleaseto" select="$overallreleaseto"/>
                   </xsl:call-template>
                </xsl:template>

                <!-- **************************************************************** -->
<!-- portionmark - renders the security portion marking for each of   -->
<!--              the document's portion level elements.             -->
<!-- **************************************************************** -->
<!-- **************************************************************** -->
<!-- NOTE: The "overalldissem" and "overallreleaseto" parameters are  -->
<!--       used to compare the document-level "REL TO" or "EYES ONLY" -->
<!--       dissemination controls to the corresponding portion-level  -->
<!--       dissemination controls (as specified in the "dissem" and   -->
<!--       "releaseto" parameters).                                   -->
<!--                                                                  -->
<!--       As per IC guidelines, "REL TO" and "EYES ONLY" portion     -->
<!--       markings can be abbreviated when they would otherwise be   -->
<!--       identical to the corresponding document-level markings.    -->
<!--                                                                  -->
<!--       The "overalldissem" and "overallreleaseto" parameters are  -->
<!--       not required.  However, if the parameters are not passed  -->
<!--       into the template, a comparison can not be made, in which  -->
<!--       case the full "REL TO" or "EYES ONLY" dissemination        -->
<!--       control markings will be rendered for the portion even     -->
<!--       when the portion-level and document-level dissemination    -->
<!--       control markings are the same.                             -->
<!-- **************************************************************** -->
<xsl:template name="portionmark">
                   <xsl:param name="class"/>
                   <xsl:param name="ownerproducer"/>
                   <xsl:param name="joint"/>
                   <xsl:param name="sci"/>
                   <xsl:param name="sar"/>
                   <xsl:param name="atomicenergymarkings"/>
                   <xsl:param name="fgiopen"/>
```

```
                 <xsl:param name="fgiprotect"/>
                 <xsl:param name="dissem"/>
                 <xsl:param name="releaseto"/>
                 <xsl:param name="displayonly"/>
                 <xsl:param name="cuiBasic"/>
                 <xsl:param name="cuiSpecified"/>
                 <xsl:param name="nonic"/>
                 <xsl:param name="nonuscontrols"/>
                 <xsl:param name="compliesWith"/>
                 <xsl:param name="overalldissem"/>
                 <xsl:param name="overallreleaseto"/>

                 <!-- **** Normalize all of the parameters. **** -->
<xsl:variable name="n-class" select="normalize-space($class)"/>
                 <xsl:variable name="n-joint" select="normalize-space($joint)"/>

                 <!-- Sort ownerproducer Based on CVE -->
<xsl:variable name="n-ownerproducer">
                    <xsl:value-of select="ism-func:sortOwnerProducer($ownerproducer)"/>
                 </xsl:variable>


                 <!-- Sort SCI alphabetically -->
<xsl:variable name="n-sci">
                    <xsl:value-of select="ism-func:sortSciControls($sci)"/>
                 </xsl:variable>

                 <!-- Sort atomicenergymarkings Based on CVE -->
<!-- Requires 2020-JUN or later CVE with regex replaced by actual values.  -->
<xsl:variable name="n-atomicenergymarkings">
                    <xsl:value-of select="ism-func:sortAtomicenergymarkings($atomicenergymarkings)"/>
                 </xsl:variable>

                 <!-- Sort fgiopen Based on CVE -->
<xsl:variable name="n-fgiopen">
                    <xsl:value-of select="ism-func:sortFGIOpen($fgiopen)"/>
                 </xsl:variable>

                 <!-- Sort fgiprotect Based on CVE -->
<!-- Should not matter since any protected renders as just FGI. -->
<xsl:variable name="n-fgiprotect">
                    <xsl:value-of select="ism-func:sortFGIProtected($fgiprotect)"/>
                 </xsl:variable>

                 <!-- **** Determine the set of dissemination controls that are not CUI limited dissem controls -->
<xsl:variable name="dissemsNotCui">
                    <xsl:if test="$dissem != ''">
                       <xsl:value-of select="ism-func:get.dissemNotCUI($dissem)"/>
                    </xsl:if>
                 </xsl:variable>

                 <!-- Variable to determine if there are any IC Register-specific control markings that are not CUI limited dissem controls -->
<xsl:variable name="CUIandICcontrolMarkings">
                    <xsl:choose>
```

```
                           <xsl:when test="             ($cuiBasic != '' or $cuiSpecified != '') and ($n-class = '' or $n-class = 'U') and             (string($sci) = '' and string($sar) = ''
and string($atomicenergymarkings) = '' and             string($fgiopen) = '' and string($fgiprotect) = '' and string($nonic) = '' and string($dissemsNotCui) = '')">
                              <xsl:value-of select="false()"/>
                           </xsl:when>
                           <xsl:otherwise>
                              <xsl:value-of select="true()"/>
                           </xsl:otherwise>
                        </xsl:choose>
                     </xsl:variable>

                     <!-- Sort Dissem Based on CVE -->
    <xsl:variable name="n-dissem">
                        <xsl:variable name="sortedDissem"
                                      select="ism-func:sortDissemControls($dissem, $compliesWith, $CUIandICcontrolMarkings)"/>
                        <xsl:value-of select="replace(normalize-space($sortedDissem), 'OC OC-USGOV', 'OC-USGOV')"/>
                     </xsl:variable>

                     <!-- Sort RelTo Based on CVE -->
    <xsl:variable name="n-releaseto">
                        <xsl:value-of select="ism-func:sortReleaseto($releaseto)"/>
                     </xsl:variable>

                     <!-- Sort DisplayOnly Based on CVE -->
    <xsl:variable name="n-displayonly">
                        <xsl:value-of select="ism-func:sortDisplayonly($displayonly)"/>
                     </xsl:variable>

                     <!-- Sort NonIC Based on CVE -->
    <xsl:variable name="n-nonic">
                        <xsl:value-of select="ism-func:sortNonic($nonic)"/>
                     </xsl:variable>

                     <!-- Sort cuiBasic alphabetically -->
    <xsl:variable name="n-cuiBasic">
                        <xsl:value-of select="ism-func:sortCuiBasic($cuiBasic)"/>
                     </xsl:variable>

                     <!-- Sort cuiSpecified alphabetically -->
    <xsl:variable name="n-cuiSpecified">
                        <xsl:value-of select="ism-func:sortCuiSpecified($cuiSpecified)"/>
                     </xsl:variable>

                     <xsl:variable name="n-nonuscontrls" select="normalize-space($nonuscontrols)"/>
                     <xsl:variable name="n-overalldissem" select="normalize-space($overalldissem)"/>

                     <!-- Sort overallreleaseto Based on CVE -->
    <xsl:variable name="n-overallreleaseto">
                        <xsl:value-of select="ism-func:sortReleaseto($overallreleaseto)"/>
                     </xsl:variable>

                     <!-- **** Determine the classification marking **** -->
    <xsl:variable name="classVal">
                        <xsl:choose>
                           <xsl:when test="$n-class != ''">
```

```
                  <xsl:choose>
<!-- Multiple Owner Producers JOINT true -->
<xsl:when test="$n-ownerproducer = ''">
                        <xsl:value-of select="$warn-parse-ownerproducer"/>
                     </xsl:when>
                     <xsl:when test="contains($n-ownerproducer, ' ') and ($n-fgiprotect = '') and $joint = 'true'">
                        <xsl:choose>
                           <xsl:when test="not($n-class = ('TS', 'S', 'C', 'R', 'U'))">
                              <xsl:value-of select="$warn-parse-classif"/>
                           </xsl:when>
                           <xsl:otherwise>
                              <xsl:text>//JOINT </xsl:text>
                              <xsl:value-of select="$n-class"/>
                              <xsl:text> </xsl:text>
                              <xsl:value-of select="$n-ownerproducer"/>
                           </xsl:otherwise>
                        </xsl:choose>
                     </xsl:when>

                     <!-- Multiple Owner Producers JOINT false USA not one of the producers -->
<xsl:when test="contains($n-ownerproducer, ' ') and not(contains($n-ownerproducer, 'USA')) and ($n-fgiprotect = '') and $joint != 'true'">
                        <xsl:choose>
                           <xsl:when test="not($n-class = ('TS', 'S', 'C', 'R', 'U'))">
                              <xsl:value-of select="$warn-parse-classif"/>
                           </xsl:when>
                           <xsl:otherwise>
                              <xsl:text>//</xsl:text>
                              <xsl:value-of select="$n-ownerproducer"/>
                              <xsl:text> </xsl:text>
                              <xsl:value-of select="$n-class"/>
                           </xsl:otherwise>
                        </xsl:choose>
                     </xsl:when>

                     <!-- Multiple Owner Producers JOINT false USA one of the producers invalid state-->
<xsl:when test="contains($n-ownerproducer, ' ') and contains($n-ownerproducer, 'USA') and ($n-fgiprotect = '') and $joint != 'true'">
                        <xsl:value-of select="$warn-parse-classif"/>
                     </xsl:when>

                     <xsl:when test="($n-ownerproducer = 'USA') and ($n-fgiopen != 'UNKNOWN')">
                        <xsl:choose>
                           <xsl:when test="not($n-class = ('TS', 'S', 'C', 'U'))">
                              <xsl:value-of select="$warn-parse-classif"/>
                           </xsl:when>
                           <xsl:otherwise>
                              <xsl:value-of select="$n-class"/>
                           </xsl:otherwise>
                        </xsl:choose>
                     </xsl:when>
                     <xsl:when test="$n-ownerproducer = 'NATO'">
                        <xsl:choose>
                           <xsl:when test="$n-class = 'TS'">
                              <xsl:text>//CTS</xsl:text>
                           </xsl:when>
```

```
         <xsl:when test="$n-class = ('S', 'C', 'R', 'U')">
            <xsl:text>//N</xsl:text>
            <xsl:value-of select="$n-class"/>
         </xsl:when>
         <xsl:otherwise>
            <xsl:value-of select="$warn-parse-classif"/>
         </xsl:otherwise>
      </xsl:choose>
      <xsl:if test="$n-nonuscontrls">
         <xsl:text>//</xsl:text>
         <xsl:value-of select="translate($n-nonuscontrls, ' ', '/')"/>
      </xsl:if>
   </xsl:when>
   <xsl:when test="starts-with($n-ownerproducer, 'NATO:')">
      <xsl:variable name="natoNacString">
         <xsl:call-template name="ism:get.nato.nac.portion">
            <xsl:with-param name="source" select="$n-ownerproducer"/>
         </xsl:call-template>
      </xsl:variable>
      <xsl:choose>
         <xsl:when test="$n-class = ('S', 'C', 'R', 'U')">
            <xsl:value-of select="concat('//N', $natoNacString, $n-class)"/>
         </xsl:when>
         <xsl:otherwise>
            <xsl:value-of select="$warn-parse-classif"/>
         </xsl:otherwise>
      </xsl:choose>
      <xsl:if test="$n-nonuscontrls">
         <xsl:text>//</xsl:text>
         <xsl:value-of select="translate($n-nonuscontrls, ' ', '/')"/>
      </xsl:if>
   </xsl:when>

   <xsl:otherwise>
      <xsl:choose>
         <xsl:when test="not($n-class = ('TS', 'S', 'C', 'R', 'U'))">
            <xsl:value-of select="$warn-parse-classif"/>
         </xsl:when>
         <xsl:otherwise>
            <xsl:text>//</xsl:text>
            <xsl:choose>
               <xsl:when test="($n-fgiprotect != '') or ($n-fgiopen = 'UNKNOWN')">
                  <xsl:text>FGI</xsl:text>
               </xsl:when>
               <xsl:otherwise>
                  <xsl:value-of select="$n-ownerproducer"/>
               </xsl:otherwise>
            </xsl:choose>
            <xsl:text> </xsl:text>
            <xsl:value-of select="$n-class"/>
         </xsl:otherwise>
      </xsl:choose>
   </xsl:otherwise>
</xsl:choose>
```

```
                       </xsl:when>
                       <xsl:otherwise>
                          <xsl:value-of select="$warn-missing-classif"/>
                       </xsl:otherwise>
                    </xsl:choose>
                 </xsl:variable>

                 <!-- **** Determine the SCI marking **** -->
<xsl:variable name="sciVal">
                    <xsl:value-of select="ism-func:sciVal($n-sci, $n-nonuscontrls)"/>
                 </xsl:variable>

                 <!-- **** Determine AtomicEnergyMarking ****-->
<xsl:variable name="atomicEnergyVal">
                    <xsl:value-of select="ism-func:AEAVal($n-atomicenergymarkings, $n-nonuscontrls, false())"/>
                 </xsl:variable>

                 <!-- **** Determine the SAR marking **** -->
<xsl:variable name="sarVal">
                    <xsl:if test="$sar != ''">
                       <xsl:text>//SAR-</xsl:text>
                       <xsl:call-template name="ism:get.sar.pm">
                          <xsl:with-param name="all" select="$sar"/>
                       </xsl:call-template>
                    </xsl:if>
                 </xsl:variable>

                 <!-- **** Determine the dissemination marking **** -->
<xsl:variable name="dissemVal">
                    <xsl:if test="$n-dissem != ''">
                       <xsl:variable name="val" select="$n-dissem"/>
                       <xsl:text>//</xsl:text>
                       <xsl:call-template name="ism:get.dissem.pm">
                          <xsl:with-param name="all" select="$val"/>
                          <xsl:with-param name="relto" select="$n-releaseto"/>
                          <xsl:with-param name="displayonly" select="$n-displayonly"/>
                          <xsl:with-param name="overalldissem" select="$n-overalldissem"/>
                          <xsl:with-param name="overallrelto" select="$n-overallreleaseto"/>
                       </xsl:call-template>
                    </xsl:if>
                 </xsl:variable>

                 <!-- **** Determine the non-IC marking **** -->
<xsl:variable name="nonicVal">
                    <xsl:if test="$n-nonic != ''">
                       <xsl:variable name="val" select="$n-nonic"/>
                       <xsl:variable name="DissemLookup" select="()"/>
                       <xsl:text>//</xsl:text>
                       <xsl:value-of select="ism-func:get.nonic($val, $DissemLookup)"/>
                    </xsl:if>
                 </xsl:variable>


                 <!-- **** Determine the FGI marking **** -->
```

```
    <xsl:variable name="fgiVal">
      <!-- ****************************************************************************************************** -->
      <!-- FGI markings are only used when foreign government information is included in a US controlled document, -->
      <!-- or when the document is jointly controlled and 'USA' is an owner/producer and a non-US owner/producer   -->
      <!-- is protected.                                                                                           -->
      <!-- ****************************************************************************************************** -->
      <xsl:if test="(($n-ownerproducer = 'USA') or (contains($n-ownerproducer, 'USA') and $n-fgiprotect != ''))">
                           <xsl:choose>
                              <xsl:when test="(($n-fgiopen != '') and (not(contains($n-fgiopen, 'UNKNOWN'))) and ($n-fgiprotect = ''))">

                                 <xsl:text>//FGI </xsl:text>
                                 <xsl:value-of select="translate($n-fgiopen, '_:', '  ')"/>
                                 <xsl:if test="$n-nonuscontrls">
                                    <xsl:variable name="nonatocontrls">
                                       <xsl:value-of select="                     translate(                     normalize-space(translate(translate(translate($n-nonuscontrls, 'BALK', ' '),
'BOHEMIA', ' '), 'ATOMAL', ' ')),                     ' ', '/')"/>
                                    </xsl:variable>
                                    <xsl:if test="$nonatocontrls">
                                       <xsl:value-of select="$nonatocontrls"/>
                                    </xsl:if>
                                 </xsl:if>
                              </xsl:when>
                              <xsl:when test="(($n-fgiprotect != '') or (contains($n-fgiopen, 'UNKNOWN')))">
            <!-- *************************************************************** -->
            <!-- Display the generic FGI marking when the document:              -->
            <!--                                                                 -->
            <!--   1.  contains some FGI from a protected source(s)              -->
            <!--   2.  contains some FGI from an unknown source(s)               -->
            <!--                                                                 -->
            <!-- *************************************************************** -->
            <xsl:text>//FGI</xsl:text>
                              </xsl:when>
                           </xsl:choose>
                        </xsl:if>
                     </xsl:variable>


                     <!-- **** Determine the cuiBasic marking **** -->
    <xsl:variable name="cuiBasicVal">
                        <xsl:if test="$n-cuiBasic != ''">
                           <xsl:choose>
                              <xsl:when test="$n-cuiSpecified != ''">
                                 <xsl:text>/</xsl:text>
                              </xsl:when>
                              <xsl:otherwise>
                                 <xsl:text>//</xsl:text>
                              </xsl:otherwise>
                           </xsl:choose>
                           <xsl:value-of select="ism-func:get.cuiBasic($n-cuiBasic)"/>
                        </xsl:if>
                     </xsl:variable>

                     <!-- **** Determine the cuiSpecified marking **** -->
    <xsl:variable name="cuiSpecifiedVal">
```

```
                        <xsl:if test="$n-cuiSpecified != ''">
                           <xsl:text>//</xsl:text>
                           <xsl:value-of select="ism-func:get.cuiSpecified($n-cuiSpecified)"/>
                        </xsl:if>
                     </xsl:variable>

                     <!-- **** Output the values as a single string **** -->
    <xsl:choose>
                        <xsl:when test="           ($cuiBasicVal != '' or $cuiSpecified != '') and ($n-class = '' or $n-class = 'U') and           ($sciVal = '' and $sarVal = '' and
$atomicEnergyVal = '' and $fgiVal = '' and $nonicVal = '' and $dissemsNotCui = '')">
                           <xsl:text>CUI</xsl:text>
                        </xsl:when>
                        <xsl:otherwise>
                           <xsl:value-of select="$classVal"/>
                        </xsl:otherwise>
                     </xsl:choose>
                     <xsl:value-of select="$sciVal"/>
                     <xsl:value-of select="$sarVal"/>
                     <xsl:value-of select="$atomicEnergyVal"/>
                     <xsl:value-of select="$fgiVal"/>
                     <xsl:if test="         ($cuiBasicVal != '' or $cuiSpecified != '') and         (($sciVal != '' or $sarVal != '' or $atomicEnergyVal != '' or $fgiVal != '' or
$dissemsNotCui != '')         or ($n-class != '' and $n-class != 'U'))">
      <!-- or (($n-class = '' or $n-class = 'U') and $CUIRenderingRuleSet = 'DOD'))"> -->
      <xsl:text>//CUI</xsl:text>
                     </xsl:if>
                     <xsl:value-of select="$cuiSpecifiedVal"/>
                     <xsl:value-of select="$cuiBasicVal"/>
                     <xsl:value-of select="$dissemVal"/>
                     <xsl:value-of select="$nonicVal"/>

                  </xsl:template>



                  <!-- ******************************************************************* -->
  <!-- A  routine for processing disseminationControl name tokens -->
  <!-- ******************************************************************* -->
  <xsl:template name="ism:get.dissem.pm">
                     <xsl:param name="all"/>
                     <xsl:param name="relto"/>
                     <xsl:param name="displayonly"/>
                     <xsl:param name="overalldissem"/>
                     <xsl:param name="overallrelto"/>

                     <xsl:for-each select="tokenize($all, ' ')">
      <!-- The dissemination control EXEMPT_FROM_ICD501_DISCOVERY is not rendered -->
      <xsl:if test="not(current() = 'EXEMPT_FROM_ICD501_DISCOVERY')">
        <!-- Add a preceding / for all but the first dissem control. -->
        <xsl:if test="position() != 1">
                              <xsl:text>/</xsl:text>
                           </xsl:if>
                           <xsl:call-template name="ism:get.dissem.names">
                              <xsl:with-param name="name" select="current()"/>
                              <xsl:with-param name="rel" select="$relto"/>
```

```
                            <xsl:with-param name="displayonly" select="$displayonly"/>
                            <xsl:with-param name="overalldissem" select="$overalldissem"/>
                            <xsl:with-param name="overallrelto" select="$overallrelto"/>
                         </xsl:call-template>
                      </xsl:if>
                   </xsl:for-each>
                </xsl:template>

                <!-- **************************************************** -->
<!-- Determine releasableTo name tokens for REL and EYES -->
<!-- **************************************************** -->
<xsl:template name="ism:get.dissem.names">
                   <xsl:param name="name"/>
                   <xsl:param name="rel"/>
                   <xsl:param name="displayonly"/>
                   <xsl:param name="overalldissem"/>
                   <xsl:param name="overallrelto"/>

                   <xsl:choose>
                      <xsl:when test="$name = 'REL'">
                         <xsl:choose>
                            <xsl:when test="($rel != '')">
                               <xsl:choose>
                                  <xsl:when test="(contains($overalldissem, 'REL') and ($overallrelto = $rel))">
                                     <xsl:text>REL</xsl:text>
                                  </xsl:when>
                                  <xsl:otherwise>
                                     <xsl:text>REL TO </xsl:text>
                                     <xsl:value-of select="ism-func:get.relOrDisplayString($rel)"/>
                                  </xsl:otherwise>
                               </xsl:choose>
                            </xsl:when>
                            <xsl:otherwise>
                               <xsl:value-of select="$warn-parse-relto"/>
                            </xsl:otherwise>
                         </xsl:choose>
                      </xsl:when>
                      <xsl:when test="$name = 'EYES'">
                         <xsl:choose>
                            <xsl:when test="($rel != '')">
                               <xsl:choose>
                                  <xsl:when test="(contains($overalldissem, 'EYES') and ($overallrelto = $rel))">
                                     <xsl:text>EYES</xsl:text>
                                  </xsl:when>
                                  <xsl:otherwise>
                                     <xsl:value-of select="ism-func:get.eyesString($rel)"/>
                                     <xsl:text> EYES ONLY</xsl:text>
                                  </xsl:otherwise>
                               </xsl:choose>
                            </xsl:when>
                            <xsl:otherwise>
                               <xsl:value-of select="$warn-parse-eyes"/>
                            </xsl:otherwise>
                         </xsl:choose>
                      </xsl:when>
                      <xsl:when test="$name = 'DISPLAYONLY'">
                         <xsl:text>DISPLAY ONLY </xsl:text>
                         <xsl:choose>
                            <xsl:when test="($displayonly != '')">
                               <xsl:value-of select="ism-func:get.relOrDisplayString($displayonly)"/>
                            </xsl:when>
                            <xsl:otherwise>
                               <xsl:value-of select="$warn-parse-displayonly"/>
                            </xsl:otherwise>
                         </xsl:choose>
                      </xsl:when>
                      <xsl:otherwise>
                         <xsl:value-of select="$name"/>
                      </xsl:otherwise>
                   </xsl:choose>

                </xsl:template>

                <!-- ************************************************** -->
<!-- A routine for processing SAR name tokens -->
<!-- ************************************************** -->
<xsl:template name="ism:get.sar.pm">
                   <xsl:param name="all"/>

                   <!-- Create tokenized SAR variable.                              -->
  <xsl:variable name="tokenizedSARinitial" select="tokenize($all, ' ')"/>
                   <!-- We need to throw away the metadata for SAR owners and any required classification levels.
       First throw away any classification levels.  Second, get the unique values.  Example if
       a portion has DOD:TS:SAP1 and another portion has SAR-DOD:C:SAP1 then both will appear in the banner metadata
       (the ISM resource element).  We need to collapse down first to get SAR-DOD:SAP1 DOD:SAP1, then get the
       unique tokens which will be a single token SAR-DOD:SAP1.  Then throw away the owner DOD: because all
       we want to render is the SAP marking SAP1.  -->
  <!-- Create a STRING variable without any classification substrings -->
  <xsl:variable name="SARnoClassification">
                      <xsl:for-each select="$tokenizedSARinitial">
                         <xsl:if test="not(position() = 1)">
                            <xsl:text> </xsl:text>
                         </xsl:if>
                         <xsl:choose>
        <!-- does token have two : characters.  If so, throw away the classification
          string, which is between the two : characters, and also throw away the SAR- prefix.
          Otherwise, just take the entire token minus the SAR- prefix. Note will add back SAR- prefix
          as needed when doing the final rendering.  -->
        <xsl:when test="contains(substring-after(., ':'), ':')">
                               <xsl:value-of select="concat(substring-before(substring-after(.,'SAR-'), ':'), ':', substring-after(substring-after(., ':'), ':'))"/>
                            </xsl:when>
                            <xsl:otherwise>
                               <xsl:value-of select="substring-after(.,'SAR-')"/>
                            </xsl:otherwise>
                         </xsl:choose>
                      </xsl:for-each>
                   </xsl:variable>
                   <!-- Get the unique values of the form SAROwner:SAPMarking -->
```

```
<xsl:variable name="tokenizedSARwithOwner"
                               select="distinct-values(tokenize($SARnoClassification))"/>

                 <!-- Convert sequence to string for sorting -->
<xsl:variable name="stringSARwithOwner">
                    <xsl:for-each select="$tokenizedSARwithOwner">
                       <xsl:if test="not(position()=1)">
                          <xsl:text> </xsl:text>
                       </xsl:if>
                       <xsl:value-of select="."/>
                    </xsl:for-each>
                 </xsl:variable>

                 <!-- Sort SAR without classifications alphabetically.          -->
<!-- Note we cannot sort the SARs until we have eliminated any -->
<!-- classification requirements in the marking                -->
<xsl:variable name="n-sar">
                    <xsl:value-of select="ism-func:sortSar($stringSARwithOwner)"/>
                 </xsl:variable>

                 <!-- Tokenize again -->
<xsl:variable name="tokenizedSAR" select="tokenize($n-sar)"/>

                 <!-- Loop over all the SAR tokens -->
<xsl:for-each select="$tokenizedSAR">
                    <xsl:variable name="tokenizedSARToken" select="tokenize(current(), '-')"/>
                    <!-- In non-DoD SARs, a dash signifies a compartment and two dashes signify a subcompartment.
           In DOD SARs, there are no compartments or subcompartments so always set $compartmentLevelCount to zero -->
  <xsl:variable name="compartmentLevelCount">
                       <xsl:choose>
                          <xsl:when test="substring-before(.,':')='DOD'">
                             <xsl:value-of select="0"/>
                          </xsl:when>
                          <xsl:otherwise>
                             <xsl:value-of select="count($tokenizedSARToken) - 1"/>
                          </xsl:otherwise>
                       </xsl:choose>
                    </xsl:variable>
                    <!--  Now get an appropriate separator (dash, slash or space) to go before the SAP marking value -->
  <xsl:choose>
    <!-- Not the first SAR and has no compartment/subcompartments add a / -->
    <xsl:when test="$compartmentLevelCount = 0 and not(position() = 1)">
                          <xsl:choose>
                             <xsl:when test="$SAPRenderingRuleSet = 'DOD'">
                                <xsl:text>/SAR-</xsl:text>
                             </xsl:when>
                             <xsl:otherwise>
                                <xsl:text>/</xsl:text>
                             </xsl:otherwise>
                          </xsl:choose>
                       </xsl:when>
                       <!-- A compartment add a - -->
    <xsl:when test="$compartmentLevelCount = 1">
                          <xsl:text>-</xsl:text>
```

```
                         </xsl:when>
                         <!-- A subcompartment add a space -->
      <xsl:when test="$compartmentLevelCount = 2">
                            <xsl:text> </xsl:text>
                         </xsl:when>
                      </xsl:choose>
                      <!-- Now generate the rendered value.  If no compartments/subcompartments, send the current $tokenizedSAR
             token to ism-func:get.sar.name.  If there are compartments/subcompartments, then we need to send
             the last part of the marking after the last dash, which is $tokenizedSARToken[last()], but we need to add back
             the SAR owner in front of the last marking -->
    <xsl:choose>
                         <xsl:when test="$compartmentLevelCount = 0">
                            <xsl:value-of select="ism-func:get.sar.name(.)"/>
                         </xsl:when>
                         <xsl:otherwise>
        <!-- For compartments and subcompartments, we need to add back the SAR owner followed by colon,
              just before the SAR marking value -->
        <xsl:variable name="SARstringWithOwner"
                                          select="concat(substring-before(.,':'),':',$tokenizedSARToken[last()])"/>
                            <xsl:value-of select="ism-func:get.sar.name($SARstringWithOwner)"/>
                         </xsl:otherwise>
                      </xsl:choose>
                   </xsl:for-each>
                </xsl:template>



                <!-- ************************************************** -->
<!-- A routine for processing atomicEnergyMarking tokens -->
<!-- ************************************************** -->
<xsl:template name="ism:get.atomicEnergyMarking.pm">
                   <xsl:param name="all"/>
                   <xsl:value-of select="ism-func:getAEA($all, false())"/>
                </xsl:template>

                <!-- ********************************************************** -->
<!-- A generic template for getting a complete portion marking  -->
<!--                                                            -->
<!-- This template can be called without any parameters from    -->
<!-- any stylesheet when the element for which a portion        -->
<!-- marking is required is the current node.  When this        -->
<!-- template is called, the output will include parentheses    -->
<!-- and a space after the portion marking.                     -->
<!-- ********************************************************** -->

<!-- **************************************************************** -->
<!-- NOTE: The "overalldissem" and "overallreleaseto" parameters are  -->
<!--       used to compare the document-level "REL TO" or "EYES ONLY" -->
<!--       dissemination controls to the corresponding portion-level  -->
<!--       dissemination controls (as specified in the "dissem" and   -->
<!--       "releaseto" parameters).                                   -->
<!--                                                                  -->
<!--       As per IC guidelines, "REL TO" and "EYES ONLY" portion     -->
<!--       markings can be abbreviated when they would otherwise be   -->
```

```
<!--       identical to the corresponding document-level markings.    -->
<!--                                                                                                                          -->
<!--       The "overalldissem" and "overallreleaseto" parameters are  -->
<!--       not required.  However, if the parameters are not passed   -->
<!--       into the template, a comparison can not be made, in which  -->
<!--       case the full "REL TO" or "EYES ONLY" dissemination        -->
<!--       control markings will be rendered for the portion even     -->
<!--       when the portion-level and document-level dissemination    -->
<!--       control markings are the same.                             -->
<!-- **************************************************************** -->
<xsl:template name="get.portionmark">
                   <xsl:param name="overalldissem"/>
                   <xsl:param name="overallreleaseto"/>

                   <xsl:text>(</xsl:text>
                   <xsl:call-template name="portionmark">
                      <xsl:with-param name="class" select="./@ism:classification"/>
                      <xsl:with-param name="ownerproducer" select="./@ism:ownerProducer"/>
                      <xsl:with-param name="joint" select="./@ism:joint"/>
                      <xsl:with-param name="sci" select="./@ism:SCIcontrols"/>
                      <xsl:with-param name="sar" select="./@ism:SARIdentifier"/>
                      <xsl:with-param name="atomicenergymarkings" select="./@ism:atomicEnergyMarkings"/>
                      <xsl:with-param name="fgiopen" select="./@ism:FGIsourceOpen"/>
                      <xsl:with-param name="fgiprotect" select="./@ism:FGIsourceProtected"/>
                      <xsl:with-param name="dissem" select="./@ism:disseminationControls"/>
                      <xsl:with-param name="releaseto" select="./@ism:releasableTo"/>
                      <xsl:with-param name="displayonly" select="./@ism:displayOnlyTo"/>
                      <xsl:with-param name="nonic" select="./@ism:nonICmarkings"/>
                      <xsl:with-param name="cuiBasic" select="./@ism:cuiBasic"/>
                      <xsl:with-param name="cuiSpecified" select="./@ism:cuiSpecified"/>
                      <xsl:with-param name="nonuscontrols" select="./@ism:nonUSControls"/>
                      <xsl:with-param name="compliesWith" select="./@ism:compliesWith"/>
                      <xsl:with-param name="overalldissem" select="$overalldissem"/>
                      <xsl:with-param name="overallreleaseto" select="$overallreleaseto"/>
                   </xsl:call-template>
                   <xsl:text>) </xsl:text>

                </xsl:template>

                <xsl:template name="get.portionmark.wxs">
                   <xsl:param name="overalldissem"/>
                   <xsl:param name="overallreleaseto"/>

                   <xsl:call-template name="get.portionmark">
                      <xsl:with-param name="overalldissem" select="$overalldissem"/>
                      <xsl:with-param name="overallreleaseto" select="$overallreleaseto"/>
                   </xsl:call-template>

                </xsl:template>

                <!-- ************************************************************ -->
<!-- Get the NATO NAC string                                      -->
<!-- ************************************************************ -->
<xsl:template name="ism:get.nato.nac.portion">
```

```
                    <xsl:param name="source"/>
                    <xsl:value-of select="document('nacs.xml')//nacs/nac[@name = substring-after($source, ':')]/@portion"/>
                 </xsl:template>


              </xsl:stylesheet>
              <!-- **************************************************************** --><!--                            UNCLASSIFIED                          --><!--
**************************************************************** -->
```

2.7 - IC-ISM-SecurityBanner.xsl

```
<!-- **************************************************************** --><!--                        UNCLASSIFIED                                                   --><!--
**************************************************************** --><!-- ****************************************************************
  INTELLIGENCE COMMUNITY TECHNICAL SPECIFICATION
  XML DATA ENCODING SPECIFICATION FOR
  INFORMATION SECURITY MARKING METADATA (ISM.XML)
  ****************************************************************
  Module:   IC-ISM-SecurityBanner.xsl
  Creators: Office of the Director of National Intelligence
  Intelligence Community Chief Information Officer
  **************************************************************** --><!-- **************************************************************** --><!--
DESCRIPTION                           --><!--                                                                        --><!-- This stylesheet renders a security banner marking from a        --><!--
document's top-level ISM attribute values. The rendered          --><!-- marking is compliant with the IC Register and Manual guidelines  --><!-- as of the 2019-AUG release of the
manual                         --><!-- **************************************************************** --><xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                               xmlns:xs="http://www.w3.org/2001/XMLSchema"
                               xmlns:cve="urn:us:gov:ic:cve"
                               xmlns:ism-func="urn:us:gov:ic:ism:functions"
                               version="2.0">

                  <xsl:import href="IC-ISM-Functions.xsl"/>

                  <xsl:output method="text"
                              encoding="UTF-8"
                              media-type="text-plain"
                              indent="no"/>
                  <!-- If including this xsl causes "Content is not allowed in prolog" the importing
  XSL is likely missing an output declaration -->

  <xsl:param name="warn-missing-classif" select="'MISSING CLASSIFICATION MARKING'"/>
                  <xsl:param name="warn-parse-classif"
                             select="'UNABLE TO DETERMINE CLASSIFICATION MARKING'"/>
                  <xsl:param name="warn-parse-ownerproducer"
                             select="concat($warn-parse-classif, ' - MISSING OWNER/PRODUCER')"/>
                  <xsl:param name="warn-parse-relto" select="'UNABLE TO DETERMINE RELEASABILITY'"/>
                  <xsl:param name="warn-parse-displayonly"
                             select="'UNABLE TO DETERMINE DISPLAY ONLY'"/>
                  <xsl:param name="warn-parse-eyes"
                             select="'UNABLE TO DETERMINE EYES ONLY MARKINGS'"/>

                  <xsl:variable name="DissemLookup" select="document('./BannerMapping.xml')"/>

                  <xsl:param name="CUIRenderingRuleSet" select="''"/>
                  <xsl:param name="SAPRenderingRuleSet" select="''"/>



                  <!--***********************************************-->
  <!-- Mode for generating the CAPCO banner -->
  <!--***********************************************-->
  <xsl:template match="*[@ism:*]" mode="ism:banner">
                     <xsl:param name="portionelement"/>
                     <xsl:param name="overalldissem"/>
                     <xsl:param name="overallreleaseto"/>
```

```
                   <xsl:param name="documentdate" select="20090331"/>
                   <xsl:call-template name="get.security.banner">
                      <xsl:with-param name="portionelement" select="$portionelement"/>
                      <xsl:with-param name="overalldissem" select="$overalldissem"/>
                      <xsl:with-param name="overallreleaseto" select="$overallreleaseto"/>
                      <xsl:with-param name="documentdate" select="$documentdate"/>
                   </xsl:call-template>
                </xsl:template>

                <!-- **************************************************************** -->
<!-- Convenience template that will invoke security.banner template
    and set the parameters with the current element's ISM attributes-->
<!-- **************************************************************** -->
<xsl:template name="get.security.banner">
                   <xsl:param name="portionelement"/>
                   <xsl:param name="overalldissem"/>
                   <xsl:param name="overallreleaseto"/>
                   <xsl:param name="documentdate" select="20090331"/>
                   <xsl:call-template name="security.banner">
                      <xsl:with-param name="class" select="./@ism:classification"/>
                      <xsl:with-param name="ownerproducer" select="./@ism:ownerProducer"/>
                      <xsl:with-param name="joint" select="./@ism:joint"/>
                      <xsl:with-param name="sci" select="./@ism:SCIcontrols"/>
                      <xsl:with-param name="atomicenergymarkings" select="./@ism:atomicEnergyMarkings"/>
                      <xsl:with-param name="sar" select="./@ism:SARIdentifier"/>
                      <xsl:with-param name="fgiopen" select="./@ism:FGIsourceOpen"/>
                      <xsl:with-param name="fgiprotect" select="./@ism:FGIsourceProtected"/>
                      <xsl:with-param name="dissem" select="./@ism:disseminationControls"/>
                      <xsl:with-param name="releaseto" select="./@ism:releasableTo"/>
                      <xsl:with-param name="displayonly" select="./@ism:displayOnlyTo"/>
                      <xsl:with-param name="cuiBasic" select="./@ism:cuiBasic"/>
                      <xsl:with-param name="cuiSpecified" select="./@ism:cuiSpecified"/>
                      <xsl:with-param name="nonic" select="./@ism:nonICmarkings"/>
                      <xsl:with-param name="nonuscontrols" select="./@ism:nonUSControls"/>
                      <xsl:with-param name="secondBannerLine" select="./@ism:secondBannerLine"/>
                      <xsl:with-param name="handleViaChannels" select="./@ism:handleViaChannels"/>
                      <xsl:with-param name="declassdate" select="./@ism:declassDate"/>
                      <xsl:with-param name="declassexception" select="./@ism:declassException"/>
                      <xsl:with-param name="declassevent" select="./@ism:declassEvent"/>
                      <xsl:with-param name="typeofexemptedsource" select="./@ism:typeOfExemptedSource"/>
                      <xsl:with-param name="declassmanualreview" select="./@ism:declassManualReview"/>
                      <xsl:with-param name="compliesWith" select="./@ism:compliesWith"/>
                      <xsl:with-param name="portionelement" select="$portionelement"/>
                      <xsl:with-param name="overalldissem" select="$overalldissem"/>
                      <xsl:with-param name="overallreleaseto" select="$overallreleaseto"/>
                      <xsl:with-param name="documentdate" select="$documentdate"/>
                   </xsl:call-template>
                </xsl:template>

                <!-- **************************************************************** -->
<!--                                                                  -->
<!--                     security.banner template                     -->
<!--                                                                  -->
<!-- NOTE: The "portionelement" parameter should be specified ONLY    -->
```

```
<!--       when the "security.banner" template is called to render a  -->
<!--       "banner style" security marking as needed for an element   -->
<!--       at the portion-level such as a table or figure.             -->
<!--                                                                   -->
<!--       The parameter can be specified in a calling stylesheet in  -->
<!--       any of the following ways:                                  -->
<!--                                                                   -->
<!--       <xsl:with-param name="portionelement" select="1"/>          -->
<!--       <xsl:with-param name="portionelement" select="'y'"/>        -->
<!--       <xsl:with-param name="portionelement" select="'Y'"/>        -->
<!--       <xsl:with-param name="portionelement" select="'yes'"/>      -->
<!--       <xsl:with-param name="portionelement" select="'Yes'"/>      -->
<!--       <xsl:with-param name="portionelement" select="'YES'"/>      -->
<!--       <xsl:with-param name="portionelement" select="true()"/>     -->
<!--       <xsl:with-param name="portionelement">1</xsl:with-param>    -->
<!--       <xsl:with-param name="portionelement">y</xsl:with-param>    -->
<!--       <xsl:with-param name="portionelement">yes</xsl:with-param> -->
<!--       <xsl:with-param name="portionelement">Yes</xsl:with-param> -->
<!--       <xsl:with-param name="portionelement">YES</xsl:with-param> -->
<!--                                                                   -->
<!-- NOTE: The "overalldissem" and "overallreleaseto" parameters are  -->
<!--       used to compare the document-level "REL TO" or "EYES ONLY" -->
<!--       dissemination controls to the corresponding portion-level  -->
<!--       dissemination controls (as specified in the "dissem" and   -->
<!--       "releaseto" parameters).                                    -->
<!--                                                                   -->
<!--       As per CAPCO guidelines, "REL TO" and "EYES ONLY" portion  -->
<!--       markings can be abbreviated when they would otherwise be   -->
<!--       identical to the corresponding document-level markings.    -->
<!--                                                                   -->
<!--       The "overalldissem" and "overallreleaseto" parameters are  -->
<!--       not required.  However, if the parameters are not passed   -->
<!--       into the template, a comparison can not be made, in which  -->
<!--       case the full "REL TO" or "EYES ONLY" dissemination        -->
<!--       control markings will be rendered for the portion even     -->
<!--       when the portion-level and document-level dissemination    -->
<!--       control markings are the same.                              -->
<!--                                                                   -->
<!-- ***************************************************************** -->
<xsl:template name="security.banner">
                  <xsl:param name="class"/>
                  <xsl:param name="ownerproducer"/>
                  <xsl:param name="joint"/>
                  <xsl:param name="sci"/>
                  <xsl:param name="sar"/>
                  <xsl:param name="atomicenergymarkings"/>
                  <xsl:param name="fgiopen"/>
                  <xsl:param name="fgiprotect"/>
                  <xsl:param name="dissem"/>
                  <xsl:param name="releaseto"/>
                  <xsl:param name="displayonly"/>
                  <xsl:param name="cuiBasic"/>
                  <xsl:param name="cuiSpecified"/>
                  <xsl:param name="nonic"/>
```

```
                 <xsl:param name="nonuscontrols"/>
                 <xsl:param name="secondBannerLine"/>
                 <xsl:param name="handleViaChannels"/>
                 <xsl:param name="declassdate"/>
                 <xsl:param name="declassexception"/>
                 <xsl:param name="declassevent"/>
                 <xsl:param name="typeofexemptedsource"/>
                 <xsl:param name="declassmanualreview"/>
                 <xsl:param name="compliesWith"/>
                 <xsl:param name="portionelement"/>
                 <xsl:param name="overalldissem"/>
                 <xsl:param name="overallreleaseto"/>
                 <xsl:param name="documentdate" select="20090331"/>

                 <!-- **** Normalize all of the parameters. **** -->
<xsl:variable name="n-class" select="normalize-space($class)"/>
                 <xsl:variable name="n-joint" select="normalize-space($joint)"/>


                 <!-- Sort ownerproducer Based on CVE -->
<xsl:variable name="n-ownerproducer">
                    <xsl:value-of select="ism-func:sortOwnerProducer($ownerproducer)"/>
                 </xsl:variable>


                 <!-- Sort SCI alphabetically -->
<xsl:variable name="n-sci">
                    <xsl:value-of select="ism-func:sortSciControls($sci)"/>
                 </xsl:variable>

                 <!-- Sort atomicenergymarkings Based on CVE -->
<!-- Requires 2020-JUN or later CVE with regex replaced by actual values.  -->
<xsl:variable name="n-atomicenergymarkings">
                    <xsl:value-of select="ism-func:sortAtomicenergymarkings($atomicenergymarkings)"/>
                 </xsl:variable>

                 <!-- Sort fgiopen Based on CVE -->
<xsl:variable name="n-fgiopen">
                    <xsl:value-of select="ism-func:sortFGIOpen($fgiopen)"/>
                 </xsl:variable>

                 <!-- Sort fgiprotect Based on CVE -->
<!-- Should not matter since any protected renders as just FGI. -->
<xsl:variable name="n-fgiprotect">
                    <xsl:value-of select="ism-func:sortFGIProtected($fgiprotect)"/>
                 </xsl:variable>

                 <!-- **** Determine the set of dissemination controls that are not CUI limited dissem controls -->
<xsl:variable name="dissemsNotCui">
                    <xsl:if test="$dissem != ''">
                       <xsl:value-of select="ism-func:get.dissemNotCUI($dissem)"/>
                    </xsl:if>
                 </xsl:variable>
```

```
                     <!-- Variable to determine if there are any IC Register-specific control markings that are not CUI limited dissem controls -->
    <xsl:variable name="CUIandICcontrolMarkings">
                        <xsl:choose>
                           <xsl:when test="             ($cuiBasic != '' or $cuiSpecified != '') and ($n-class = '' or $n-class = 'U') and             (string($sci) = '' and string($sar) = ''
and string($atomicenergymarkings) = '' and             string($fgiopen) = '' and string($fgiprotect) = '' and string($nonic) = '' and string($dissemsNotCui) = '')">
                              <xsl:value-of select="false()"/>
                           </xsl:when>
                           <xsl:otherwise>
                              <xsl:value-of select="true()"/>
                           </xsl:otherwise>
                        </xsl:choose>
                     </xsl:variable>

                     <!-- Sort Dissem Based on CVE and on the value of the document's ism:compliesWith -->
    <xsl:variable name="n-dissem">
                        <xsl:variable name="sortedDissem"
                                      select="ism-func:sortDissemControls($dissem, $compliesWith, $CUIandICcontrolMarkings)"/>
                        <xsl:value-of select="replace(normalize-space($sortedDissem), 'OC OC-USGOV', 'ORCON-USGOV')"/>
                     </xsl:variable>

                     <!-- Sort RelTo Based on CVE -->
    <xsl:variable name="n-releaseto">
                        <xsl:value-of select="ism-func:sortReleaseto($releaseto)"/>
                     </xsl:variable>

                     <!-- Sort DisplayOnly Based on CVE -->
    <xsl:variable name="n-displayonly">
                        <xsl:value-of select="ism-func:sortDisplayonly($displayonly)"/>
                     </xsl:variable>


                     <!-- Sort NonIC Based on CVE -->
    <xsl:variable name="n-nonic">
                        <xsl:value-of select="ism-func:sortNonic($nonic)"/>
                     </xsl:variable>

                     <xsl:variable name="n-nonuscontrls" select="normalize-space($nonuscontrols)"/>
                     <xsl:variable name="n-overalldissem" select="normalize-space($overalldissem)"/>


                     <!-- Sort overallreleaseto Based on CVE -->
    <xsl:variable name="n-overallreleaseto">
                        <xsl:value-of select="ism-func:sortReleaseto($overallreleaseto)"/>
                     </xsl:variable>

                     <!-- Sort cuiBasic alphabetically -->
    <xsl:variable name="n-cuiBasic">
                        <xsl:value-of select="ism-func:sortCuiBasic($cuiBasic)"/>
                     </xsl:variable>

                     <!-- Sort cuiSpecified alphabetically -->
    <xsl:variable name="n-cuiSpecified">
                        <xsl:value-of select="ism-func:sortCuiSpecified($cuiSpecified)"/>
                     </xsl:variable>
```

```
                 <!-- Sort secondBannerLine alphabetically -->
<xsl:variable name="n-secondBannerLine">
                    <xsl:value-of select="ism-func:sortSecondBannerLine($secondBannerLine)"/>
                 </xsl:variable>


                 <xsl:variable name="n-declassdate" select="normalize-space($declassdate)"/>
                 <xsl:variable name="n-declassexception" select="normalize-space($declassexception)"/>
                 <xsl:variable name="n-declassevent" select="normalize-space($declassevent)"/>
                 <xsl:variable name="n-typeofexemptedsource"
                               select="normalize-space($typeofexemptedsource)"/>
                 <xsl:variable name="n-declassmanualreview"
                               select="normalize-space($declassmanualreview)"/>



                 <xsl:variable name="isaportion">
                    <xsl:choose>
                       <xsl:when test="translate(normalize-space($portionelement), 'YES', 'yes') = 'y'">1</xsl:when>
                       <xsl:when test="translate(normalize-space($portionelement), 'YES', 'yes') = 'yes'">1</xsl:when>
                       <xsl:when test="$portionelement castable as xs:integer and number($portionelement) = 1">1</xsl:when>
                       <xsl:otherwise>0</xsl:otherwise>
                    </xsl:choose>
                 </xsl:variable>
                 <xsl:variable name="portion" select="number($isaportion)"/>

                 <!-- **** Determine the classification marking **** -->
<xsl:variable name="classVal">
                    <xsl:choose>
                       <xsl:when test="$n-class != ''">
                          <xsl:choose>
                             <xsl:when test="$n-ownerproducer = ''">
                                <xsl:value-of select="$warn-parse-ownerproducer"/>
                             </xsl:when>
                             <xsl:when test="contains($n-ownerproducer, ' ') and $n-joint = 'true'">
                                <xsl:choose>
                                   <xsl:when test="$portion and $n-fgiprotect != ''">//FGI </xsl:when>
                                   <xsl:otherwise>
                                      <xsl:if test="$n-fgiprotect = ''">
                                         <xsl:text>//</xsl:text>
                                         <xsl:text>JOINT </xsl:text>
                                      </xsl:if>
                                   </xsl:otherwise>
                                </xsl:choose>
                                <xsl:value-of select="ism-func:classStringForClass($n-class)"/>
                                <xsl:if test="not($portion) and $n-fgiprotect = ''">
                                   <xsl:text> </xsl:text>
                                   <xsl:value-of select="$n-ownerproducer"/>
                                </xsl:if>
                             </xsl:when>
                             <xsl:when test="contains($n-ownerproducer, ' ') and $n-joint != 'true'">
                                <xsl:choose>
```

```
                         <xsl:when test="$portion and $n-fgiprotect != ''">//FGI </xsl:when>
                         <xsl:otherwise>
                            <xsl:if test="$n-fgiprotect = ''">
                               <xsl:text>//</xsl:text>
                            </xsl:if>
                         </xsl:otherwise>
                      </xsl:choose>
                      <xsl:value-of select="$n-ownerproducer"/>
                      <xsl:text> </xsl:text>
                      <xsl:value-of select="ism-func:classStringForClass($n-class)"/>
                   </xsl:when>
                   <xsl:when test="(($n-ownerproducer = 'USA') and not($portion and $n-fgiopen = 'UNKNOWN'))">
<!-- **** When owner/producer is 'USA', unless this is a portion-level element and FGI source is 'UNKNOWN' **** -->
<xsl:value-of select="ism-func:classStringForClass($n-class)"/>
                   </xsl:when>
                   <xsl:when test="$n-ownerproducer = 'NATO'">
                      <xsl:choose>
                         <xsl:when test="$n-class = 'TS'">
                            <xsl:value-of select="concat('//COSMIC ', ism-func:classStringForClass($n-class))"/>
                         </xsl:when>
                         <xsl:when test="$n-class = ('S', 'C', 'R', 'U')">
                            <xsl:value-of select="concat('//NATO ', ism-func:classStringForClass($n-class))"/>
                         </xsl:when>
                         <xsl:otherwise>
                            <xsl:value-of select="$warn-parse-classif"/>
                         </xsl:otherwise>
                      </xsl:choose>
                      <xsl:if test="$n-nonuscontrls">
                         <xsl:text>//</xsl:text>
                         <xsl:value-of select="replace($n-nonuscontrls, ' ', '/')"/>
                      </xsl:if>
                   </xsl:when>
                   <xsl:when test="starts-with($n-ownerproducer, 'NATO:')">
                      <xsl:variable name="natoNacString">
                         <xsl:call-template name="ism:get.nato.nac">
                            <xsl:with-param name="source" select="$n-ownerproducer"/>
                         </xsl:call-template>
                      </xsl:variable>
                      <xsl:choose>
                         <xsl:when test="$n-class = ('S', 'C', 'R', 'U')">
                            <xsl:value-of select="concat('//NATO ', $natoNacString, ' ', ism-func:classStringForClass($n-class))"/>
                         </xsl:when>
                         <xsl:otherwise>
                            <xsl:value-of select="$warn-parse-classif"/>
                         </xsl:otherwise>
                      </xsl:choose>
                      <xsl:if test="$n-nonuscontrls">
                         <xsl:text>//</xsl:text>
                         <xsl:value-of select="replace($n-nonuscontrls, ' ', '/')"/>
                      </xsl:if>
                   </xsl:when>
                   <xsl:otherwise>
                      <xsl:choose>
                         <xsl:when test="not($n-class = ('TS', 'S', 'C', 'R', 'U'))">
```

```
                                      <xsl:value-of select="$warn-parse-classif"/>
                                   </xsl:when>
                                   <xsl:otherwise>
                                      <xsl:text>//</xsl:text>
                                      <xsl:choose>
                                         <xsl:when test="($n-ownerproducer = 'USA' and $portion and $n-fgiopen = 'UNKNOWN') or ($portion and $n-fgiprotect != '')">
                                            <xsl:text>FGI</xsl:text>
                                         </xsl:when>
                                         <xsl:otherwise>
                                            <xsl:value-of select="$n-ownerproducer"/>
                                         </xsl:otherwise>
                                      </xsl:choose>
                                      <xsl:text> </xsl:text>
                                      <xsl:value-of select="ism-func:classStringForClass($n-class)"/>
                                   </xsl:otherwise>
                                </xsl:choose>
                             </xsl:otherwise>
                          </xsl:choose>
                       </xsl:when>
                       <xsl:otherwise>
                          <xsl:value-of select="$warn-missing-classif"/>
                       </xsl:otherwise>
                    </xsl:choose>
                 </xsl:variable>

                 <!-- **** Determine the SCI marking **** -->
<xsl:variable name="sciVal">
                    <xsl:value-of select="ism-func:sciVal($n-sci, $n-nonuscontrls)"/>
                 </xsl:variable>

                 <!-- **** Determine the SAR marking **** -->
<xsl:variable name="sarVal">
                    <xsl:if test="$sar != ''">
                       <xsl:text>//SAR-</xsl:text>
                       <xsl:call-template name="ism:get.sar.banner">
                          <xsl:with-param name="all" select="$sar"/>
                       </xsl:call-template>
                    </xsl:if>
                 </xsl:variable>


                 <!-- **** Determine AtomicEnergyMarkings ****-->
<xsl:variable name="atomicEnergyVal">
                    <xsl:value-of select="ism-func:AEAVal($n-atomicenergymarkings, $n-nonuscontrls, true())"/>
                 </xsl:variable>


                 <!-- **** Determine the FGI marking **** -->
<xsl:variable name="fgiVal">
  <!-- ******************************************************************************************************* -->
  <!-- FGI markings are only used when foreign government information is included in a US controlled document, -->
  <!-- or when the document is jointly controlled and 'USA' is an owner/producer and a non-US owner/producer   -->
  <!-- is protected.                                                                                           -->
  <!-- ******************************************************************************************************* -->
```

```
      <xsl:if test="(($n-ownerproducer = 'USA') or (contains($n-ownerproducer, 'USA') and $n-fgiprotect != '')) and not($portion)">
                           <xsl:choose>
                              <xsl:when test="(($n-fgiopen != '') and (not(contains($n-fgiopen, 'UNKNOWN'))) and ($n-fgiprotect = ''))">
                                 <xsl:text>//FGI </xsl:text>
                                 <xsl:value-of select="translate($n-fgiopen, ':_', '  ')"/>
                                 <xsl:if test="$n-nonuscontrls">
                                    <xsl:variable name="nonatocontrls">
                                       <xsl:value-of select="                     replace(                     normalize-space(replace(replace(replace($n-nonuscontrls, 'BALK', ' '), 'BOHEMIA',
' '), 'ATOMAL', ' ')),                     ' ', '/')"/>
                                    </xsl:variable>
                                    <xsl:if test="$nonatocontrls">
                                       <xsl:value-of select="$nonatocontrls"/>
                                    </xsl:if>
                                 </xsl:if>
                              </xsl:when>
                              <xsl:when test="(($n-fgiprotect != '') or (contains($n-fgiopen, 'UNKNOWN')))">
            <!-- *************************************************************** -->
            <!-- Display the generic FGI marking when the document:               -->
            <!--                                                                  -->
            <!--   1.  contains some FGI from a protected source(s)              -->
            <!--   2.  contains some FGI from an unknown source(s)               -->
            <!--                                                                  -->
            <!-- *************************************************************** -->
            <xsl:text>//FGI</xsl:text>
                              </xsl:when>
                           </xsl:choose>
                        </xsl:if>
                     </xsl:variable>

                     <!-- **** Determine the dissemination marking **** -->
    <xsl:variable name="dissemVal">
                        <xsl:if test="$n-dissem != ''">
                           <xsl:text>//</xsl:text>
                           <xsl:call-template name="ism:get.dissem.banner">
                              <xsl:with-param name="all" select="$n-dissem"/>
                              <xsl:with-param name="relto" select="$n-releaseto"/>
                              <xsl:with-param name="displayonly" select="$n-displayonly"/>
                              <xsl:with-param name="ownerproducer" select="$n-ownerproducer"/>
                           </xsl:call-template>
                        </xsl:if>
                     </xsl:variable>

                     <!-- **** Determine the non-IC marking **** -->
    <xsl:variable name="nonicVal">
                        <xsl:if test="$n-nonic != ''">
                           <xsl:text>//</xsl:text>
                           <xsl:value-of select="ism-func:get.nonic($n-nonic, $DissemLookup)"/>
                        </xsl:if>
                     </xsl:variable>

                     <!-- **** Determine the cuiBasic marking **** -->
    <xsl:variable name="cuiBasicVal">
                        <xsl:if test="$n-cuiBasic != ''">
                           <xsl:choose>
```

```
                              <xsl:when test="$n-cuiSpecified != ''">
                                 <xsl:text>/</xsl:text>
                              </xsl:when>
                              <xsl:otherwise>
                                 <xsl:text>//</xsl:text>
                              </xsl:otherwise>
                           </xsl:choose>
                           <xsl:value-of select="ism-func:get.cuiBasic($n-cuiBasic)"/>
                        </xsl:if>
                     </xsl:variable>

                     <!-- **** Determine the cuiSpecified marking **** -->
    <xsl:variable name="cuiSpecifiedVal">
                        <xsl:if test="$n-cuiSpecified != ''">
                           <xsl:text>//</xsl:text>
                           <xsl:value-of select="ism-func:get.cuiSpecified($n-cuiSpecified)"/>
                        </xsl:if>
                     </xsl:variable>

                     <!-- **** Determine the second banner line value, using a pipe '|' to separate from normal banner line **** -->
    <xsl:variable name="secondBannerLineVal">
                        <xsl:if test="$n-secondBannerLine != ''">
                           <xsl:text>|</xsl:text>
                           <xsl:call-template name="ism:get.secondBannerLine">
                              <xsl:with-param name="all" select="$n-secondBannerLine"/>
                              <xsl:with-param name="HVCO" select="$handleViaChannels"/>
                           </xsl:call-template>
                        </xsl:if>
                     </xsl:variable>

                     <!-- **** Determine the declassification Manual Review marking **** -->
    <xsl:variable name="declassmanualreviewVal">
                        <xsl:if test="not($portion) and ($n-class != '') and ($n-class != 'U')">
                           <xsl:if test="             (($n-declassmanualreview = 'true') or             ($n-typeofexemptedsource != '') or             (contains($n-declassexception, '25X1-
human')) or             (contains($n-sci, 'HCS')) or             ($n-declassevent != '') or             ($fgiVal != '' and $n-declassdate = '' and $n-declassexception = '') or             (($n-
ownerproducer != '') and ($n-ownerproducer != 'USA')) or             (contains($n-dissem, 'RD')))">
                              <xsl:text>//MR</xsl:text>
                           </xsl:if>
                        </xsl:if>
                     </xsl:variable>

                     <!-- **** Determine the declassification date marking **** -->
    <xsl:variable name="declassdateVal">
                        <xsl:if test="(not($portion) and ($n-declassexception = '') and ($declassmanualreviewVal = '') and ($n-declassdate != ''))">
                           <xsl:text>//</xsl:text>
                           <xsl:value-of select="substring($n-declassdate, 1, 4)"/>
                           <xsl:variable name="month" select="substring($n-declassdate, 6, 2)"/>
                           <xsl:choose>
                              <xsl:when test="$month != ''">
                                 <xsl:value-of select="$month"/>
                              </xsl:when>
                              <xsl:otherwise>
                                 <xsl:text>01</xsl:text>
                              </xsl:otherwise>
                           </xsl:choose>
                           <xsl:variable name="day" select="substring($n-declassdate, 9, 2)"/>
                           <xsl:choose>
                              <xsl:when test="$day != ''">
                                 <xsl:value-of select="$day"/>
                              </xsl:when>
                              <xsl:otherwise>
                                 <xsl:text>01</xsl:text>
                              </xsl:otherwise>
                           </xsl:choose>
                        </xsl:if>
                     </xsl:variable>

                     <!-- **** Determine the declassification exception marking **** -->
    <xsl:variable name="declassexceptionVal">
                        <xsl:if test="not($portion) and ($declassmanualreviewVal = '') and ($n-declassexception != '')">
                           <xsl:text>//</xsl:text>
                           <xsl:choose>
                              <xsl:when test="contains($n-declassexception, ' ')">
                                 <xsl:value-of select="substring-before($n-declassexception, ' ')"/>
                              </xsl:when>
                              <xsl:otherwise>
                                 <xsl:value-of select="$n-declassexception"/>
                              </xsl:otherwise>
                           </xsl:choose>
                        </xsl:if>
                     </xsl:variable>

                     <!-- **** Output the values as a single string **** -->
    <xsl:choose>
                        <xsl:when test="           ($cuiBasicVal != '' or $cuiSpecified != '') and ($n-class = '' or $n-class = 'U') and           ($sciVal = '' and $sarVal = '' and 
$atomicEnergyVal = '' and $fgiVal = '' and $nonicVal = '' and $dissemsNotCui = '')">
                           <xsl:text>CUI</xsl:text>
                        </xsl:when>
                        <xsl:otherwise>
                           <xsl:value-of select="$classVal"/>
                        </xsl:otherwise>
                     </xsl:choose>
                     <xsl:value-of select="$sciVal"/>
                     <xsl:value-of select="$sarVal"/>
                     <xsl:value-of select="$atomicEnergyVal"/>
                     <xsl:value-of select="$fgiVal"/>
                     <xsl:if test="         ($cuiBasicVal != '' or $cuiSpecified != '') and (($sciVal != '' or $sarVal != '' or $atomicEnergyVal != '' or $fgiVal != '' or $dissemsNotCui != 
'')         or ($n-class != '' and $n-class != 'U'))">
                        <xsl:text>//CUI</xsl:text>
                     </xsl:if>
                     <xsl:value-of select="$cuiSpecifiedVal"/>
                     <xsl:value-of select="$cuiBasicVal"/>
                     <xsl:value-of select="$dissemVal"/>
                     <xsl:value-of select="$nonicVal"/>
                     <xsl:value-of select="$secondBannerLineVal"/>

                     <!-- ********************************************************************************* -->
    <!-- Note: Instead of just not including the banner declassification field, it will be -->
    <!--       included if the "document date" is earlier than 20090331.                   -->
    <!-- ********************************************************************************* -->
    <xsl:if test="number($documentdate) &lt; 20090331">
                        <xsl:value-of select="$declassdateVal"/>
                        <xsl:value-of select="$declassexceptionVal"/>
                        <xsl:value-of select="$declassmanualreviewVal"/>

                        <xsl:if test="           (not($portion) and           ($n-class != '') and ($n-class != 'U') and           ($declassdateVal = '') and ($declassexceptionVal = '') and
($declassmanualreviewVal = ''))">
                           <xsl:text>//</xsl:text>
                           <xsl:value-of select="$warn-missing-classif"/>
                        </xsl:if>
                     </xsl:if>

                  </xsl:template>

                  <!-- ************************************************** -->
  <!-- A routine for processing SAR name tokens -->
  <!-- ************************************************** -->
  <xsl:template name="ism:get.sar.banner">
                     <xsl:param name="all"/>

                     <!-- Create tokenized SAR variable.                             -->
    <xsl:variable name="tokenizedSARinitial" select="tokenize($all, ' ')"/>
                     <!-- We need to throw away the metadata for SAR owners and any required classification levels before rendering SAPs.
         First throw away any classification levels.  Second, get the unique values.  Example if
         a portion has DOD:TS:SAP1 and another portion has SAR-DOD:C:SAP1 then both will appear in the banner metadata
         (the ISM resource element).  We need to collapse down first to get SAR-DOD:SAP1 DOD:SAP1, then get the
         unique tokens which will be a single token SAR-DOD:SAP1.  Then throw away the owner DOD: because all
         we want to render is the SAP marking SAP1; this last step is done in the function ism-func:get.sar.name
         in IC-ISM-Functions.xsl.  -->
    <!-- Create a STRING variable without any classification substrings -->
    <xsl:variable name="SARnoClassification">
                        <xsl:for-each select="$tokenizedSARinitial">
                           <xsl:if test="not(position() = 1)">
                              <xsl:text> </xsl:text>
                           </xsl:if>
                           <xsl:choose>
          <!-- does token have two : characters.  If so, throw away the classification
            string, which is between the two : characters, and also throw away the SAR- prefix.
            Otherwise, just take the entire token minus the SAR- prefix. Note will add back SAR- prefix
            as needed when doing the final rendering.  -->
          <xsl:when test="contains(substring-after(., ':'), ':')">
                                 <xsl:value-of select="concat(substring-before(substring-after(.,'SAR-'), ':'), ':', substring-after(substring-after(., ':'), ':'))"/>
                              </xsl:when>
                              <xsl:otherwise>
                                 <xsl:value-of select="substring-after(.,'SAR-')"/>
                              </xsl:otherwise>
                           </xsl:choose>
                        </xsl:for-each>
                     </xsl:variable>
                     <!-- Get the unique values of the form SAROwner:SAPMarking -->
    <xsl:variable name="tokenizedSARwithOwner"
                                   select="distinct-values(tokenize($SARnoClassification))"/>
```

```
                 <!-- Convert sequence to string for sorting -->
<xsl:variable name="stringSARwithOwner">
                    <xsl:for-each select="$tokenizedSARwithOwner">
                       <xsl:if test="not(position() = 1)">
                          <xsl:text> </xsl:text>
                       </xsl:if>
                       <xsl:value-of select="."/>
                    </xsl:for-each>
                 </xsl:variable>

                 <!-- Sort SAR without classifications alphabetically.          -->
<!-- Note we cannot sort the SARs until we have eliminated any -->
<!-- classification requirements in the marking                -->
<xsl:variable name="n-sar">
                    <xsl:value-of select="ism-func:sortSar($stringSARwithOwner)"/>
                 </xsl:variable>

                 <!-- Tokenize again -->
<xsl:variable name="tokenizedSAR" select="tokenize($n-sar)"/>

                 <!-- If going by DoD rules (called by the                       -->
<!-- IC-ISM-DOD-Rendering.xsl) and if 3 or more SARs            -->
<!-- then use 'SAR-MULTIPLE PROGRAMS' per DOD Manual 5205.07 V4 -->
<xsl:choose>
                    <xsl:when test="$SAPRenderingRuleSet = 'DOD' and count($tokenizedSAR) &gt; 2">
                       <xsl:value-of select="'MULTIPLE PROGRAMS'"/>
                    </xsl:when>
                    <xsl:otherwise>
    <!-- Loop over all the SAR tokens -->
    <xsl:for-each select="$tokenizedSAR">
                          <xsl:variable name="tokenizedSARToken" select="tokenize(current(), '-')"/>
                          <!-- In non-DoD SARs, a dash signifies a compartment and two dashes signify a subcompartment.
           In DOD SARs, there are no compartments or subcompartments so always set $compartmentLevelCount to zero -->
      <xsl:variable name="compartmentLevelCount">
                             <xsl:choose>
                                <xsl:when test="substring-before(.,':')='DOD'">
                                   <xsl:value-of select="0"/>
                                </xsl:when>
                                <xsl:otherwise>
                                   <xsl:value-of select="count($tokenizedSARToken) - 1"/>
                                </xsl:otherwise>
                             </xsl:choose>
                          </xsl:variable>
                          <!--  Now get an appropriate separator (dash, slash or space) to go before the SAP marking value -->
      <xsl:choose>
        <!-- Not the first SAR and has no compartment/subcompartments add a / -->
        <xsl:when test="$compartmentLevelCount = 0 and not(position() = 1)">
                                <xsl:choose>
                                   <xsl:when test="$SAPRenderingRuleSet = 'DOD'">
                                      <xsl:text>/SAR-</xsl:text>
                                   </xsl:when>
                                   <xsl:otherwise>
                                      <xsl:text>/</xsl:text>
```

```
                                     </xsl:otherwise>
                                  </xsl:choose>
                               </xsl:when>
                               <!-- A compartment add a - -->
          <xsl:when test="$compartmentLevelCount = 1">
                                  <xsl:text>-</xsl:text>
                               </xsl:when>
                               <!-- A subcompartment add a space -->
          <xsl:when test="$compartmentLevelCount = 2">
                                  <xsl:text> </xsl:text>
                               </xsl:when>
                            </xsl:choose>
                            <!-- Now generate the rendered value.  If no compartments/subcompartments, send the current $tokenizedSAR
             token to ism-func:get.sar.name.  If there are compartments/subcompartments, then we need to send
             the last part of the marking after the last dash, which is $tokenizedSARToken[last()], but we need to add back
             the SAR owner in front of the last marking -->
        <xsl:choose>
                               <xsl:when test="$compartmentLevelCount = 0">
                                  <xsl:value-of select="ism-func:get.sar.name(.)"/>
                               </xsl:when>
                               <xsl:otherwise>
            <!-- For compartments and subcompartments, we need to add back the SAR owner followed by colon,
              just before the SAR marking value -->
            <xsl:variable name="SARstringWithOwner"
                                                select="concat(substring-before(.,':'),':',$tokenizedSARToken[last()])"/>
                                  <xsl:value-of select="ism-func:get.sar.name($SARstringWithOwner)"/>
                               </xsl:otherwise>
                            </xsl:choose>
                         </xsl:for-each>
                      </xsl:otherwise>
                   </xsl:choose>
                </xsl:template>

                <!-- ******************************************************************** -->
<!-- A  routine for processing dissemination control name tokens -->
<!-- ******************************************************************** -->
<xsl:template name="ism:get.dissem.banner">
                   <xsl:param name="all"/>
                   <xsl:param name="relto"/>
                   <xsl:param name="displayonly"/>
                   <xsl:param name="ownerproducer"/>

                   <xsl:for-each select="tokenize($all, ' ')">
    <!-- The dissemination control EXEMPT_FROM_ICD501_DISCOVERY is not rendered -->
    <xsl:if test="not(current() = 'EXEMPT_FROM_ICD501_DISCOVERY')">
      <!-- Add a preceding / for all but the first dissem control. -->
      <xsl:if test="position() != 1">
                            <xsl:text>/</xsl:text>
                         </xsl:if>
                         <xsl:call-template name="ism:get.dissemControl.names">
                            <xsl:with-param name="name" select="current()"/>
                            <xsl:with-param name="rel" select="$relto"/>
                            <xsl:with-param name="displayonly" select="$displayonly"/>
                            <xsl:with-param name="ownerproducer" select="$ownerproducer"/>
                         </xsl:call-template>
                      </xsl:if>
                   </xsl:for-each>

                </xsl:template>

                <!-- ********************************************************** -->
<!-- Full name conversion for dissemination control name tokens -->
<!-- and determine ReleasableTo name tokens for REL and EYES    -->
<!-- ********************************************************** -->
<xsl:template name="ism:get.dissemControl.names">
                   <xsl:param name="name"/>
                   <xsl:param name="rel"/>
                   <xsl:param name="displayonly"/>
                   <xsl:param name="portion"/>
                   <xsl:param name="overalldissem"/>
                   <xsl:param name="overallrelto"/>
                   <xsl:param name="ownerproducer"/>

                   <xsl:choose>
                      <xsl:when test="$DissemLookup//BannerMap[@portion = $name]">
                         <xsl:value-of select="$DissemLookup//BannerMap[@portion = $name]/text()"/>
                      </xsl:when>
                      <xsl:when test="$name = 'REL'">
                         <xsl:choose>
                            <xsl:when test="($rel != '') and (contains($rel, ' '))">
                               <xsl:choose>
                                  <xsl:when test="($portion and contains($overalldissem, 'REL') and ($overallrelto = $rel))">
                                     <xsl:text>REL</xsl:text>
                                  </xsl:when>
                                  <xsl:otherwise>
                                     <xsl:choose>
                                        <xsl:when test="$ownerproducer = 'NATO'">
                                           <xsl:text>RELEASABLE TO </xsl:text>
                                        </xsl:when>
                                        <xsl:otherwise>
                                           <xsl:text>REL TO </xsl:text>
                                        </xsl:otherwise>
                                     </xsl:choose>

                                     <xsl:variable name="relString">
                                        <xsl:call-template name="ism:NMTOKENS-to-CSV">
                                           <xsl:with-param name="text" select="$rel"/>
                                        </xsl:call-template>
                                     </xsl:variable>
                                     <xsl:value-of select="translate($relString, '_:', '  ')"/>
                                  </xsl:otherwise>
                               </xsl:choose>
                            </xsl:when>
                            <xsl:otherwise>
                               <xsl:value-of select="$warn-parse-relto"/>
                            </xsl:otherwise>
                         </xsl:choose>
                      </xsl:when>
                      <xsl:when test="$name = 'EYES'">
                         <xsl:choose>
                            <xsl:when test="($rel != '') and (contains($rel, ' '))">
                               <xsl:choose>
                                  <xsl:when test="($portion and contains($overalldissem, 'EYES') and ($overallrelto = $rel))">
                                     <xsl:text>EYES</xsl:text>
                                  </xsl:when>
                                  <xsl:otherwise>
                                     <xsl:call-template name="ism:replace">
                                        <xsl:with-param name="text" select="$rel"/>
                                        <xsl:with-param name="find" select="' '"/>
                                        <xsl:with-param name="replace" select="'/'"/>
                                     </xsl:call-template>

                                     <xsl:text> EYES ONLY</xsl:text>
                                  </xsl:otherwise>
                               </xsl:choose>
                            </xsl:when>
                            <xsl:otherwise>
                               <xsl:value-of select="$warn-parse-eyes"/>
                            </xsl:otherwise>
                         </xsl:choose>
                      </xsl:when>
                      <xsl:when test="$name = 'DISPLAYONLY'">
                         <xsl:text>DISPLAY ONLY </xsl:text>
                         <xsl:choose>
                            <xsl:when test="($displayonly != '')">
                               <xsl:variable name="displayString">
                                  <xsl:call-template name="ism:NMTOKENS-to-CSV">
                                     <xsl:with-param name="text" select="$displayonly"/>
                                  </xsl:call-template>
                               </xsl:variable>
                               <xsl:value-of select="translate($displayString, '_:', '  ')"/>
                            </xsl:when>
                            <xsl:otherwise>
                               <xsl:value-of select="$warn-parse-displayonly"/>
                            </xsl:otherwise>
                         </xsl:choose>
                      </xsl:when>
                      <xsl:otherwise>
                         <xsl:value-of select="$name"/>
                      </xsl:otherwise>
                   </xsl:choose>

                </xsl:template>



                <!-- *************************************************** -->
<!-- Tail recursion template used to replace the occurrance of one value with another -->
<!-- *************************************************** -->
<xsl:template name="ism:replace">
                   <xsl:param name="text"/>
                   <xsl:param name="find"/>
                   <xsl:param name="replace"/>

                   <xsl:choose>
                      <xsl:when test="not(contains($text, $find))">
                         <xsl:value-of select="$text"/>
                      </xsl:when>
                      <xsl:otherwise>
                         <xsl:value-of select="substring-before($text, $find)"/>
                         <xsl:value-of select="$replace"/>
                         <xsl:call-template name="ism:replace">
                            <xsl:with-param name="text" select="substring-after($text, $find)"/>
                            <xsl:with-param name="find" select="$find"/>
                            <xsl:with-param name="replace" select="$replace"/>
                         </xsl:call-template>
                      </xsl:otherwise>
                   </xsl:choose>
                </xsl:template>


                <!-- ************************************************************ -->
<!-- Convenience template to convert space separated values (NMTOKENS) into comma separated values (CSV) -->
<!-- ************************************************************ -->
<xsl:template name="ism:NMTOKENS-to-CSV">
                   <xsl:param name="text"/>
                   <xsl:call-template name="ism:replace">
                      <xsl:with-param name="text" select="$text"/>
                      <xsl:with-param name="find" select="' '"/>
                      <xsl:with-param name="replace" select="', '"/>
                   </xsl:call-template>
                </xsl:template>


                <!-- ************************************************************ -->
<!-- Get the NATO NAC string                                          -->
<!-- ************************************************************ -->
<xsl:template name="ism:get.nato.nac">
                   <xsl:param name="source"/>
                   <xsl:value-of select="replace(substring-after($source, ':'), '_', ' ')"/>
                </xsl:template>

                <!-- ************************************************************ -->
<!-- Get the Classification string                                    -->
<!-- ************************************************************ -->
<xsl:template name="ism:get.classString">
                   <xsl:param name="source"/>
                   <xsl:choose>
                      <xsl:when test="$DissemLookup//BannerMap[@portion = $source]">
                         <xsl:value-of select="$DissemLookup//BannerMap[@portion = $source]/text()"/>
                      </xsl:when>
                      <xsl:otherwise>
                         <xsl:value-of select="$warn-parse-classif"/>
                      </xsl:otherwise>
                   </xsl:choose>
                </xsl:template>
```

```
                  <!-- ************************************************************ -->
  <!-- Get the secondBannerLine string                              -->
  <!-- ************************************************************ -->
  <xsl:template name="ism:get.secondBannerLine">
                     <xsl:param name="all"/>
                     <xsl:param name="HVCO"/>
                     <xsl:variable name="tokenizedsecondBannerLine" select="tokenize($all, ' ')"/>
                     <xsl:for-each select="$tokenizedsecondBannerLine">
                        <xsl:value-of select="ism-func:get.secondBannerLine.name(current(), $HVCO)"/>
                        <!-- Add a trailing / for all but the last non-ic dissem control. -->
      <xsl:if test="position() != last()">
                           <xsl:text>/</xsl:text>
                        </xsl:if>
                     </xsl:for-each>
                  </xsl:template>



               </xsl:stylesheet>
               <!-- **************************************************************** --><!-- **************************************************************** --><!--
UNCLASSIFIED                                                        --><!-- **************************************************************** -->
```

2.8 - nacs.xml

```
<nacs>
                  <nac name="ISAF" portion="I"/>
                  <nac name="Partnership_For_Peace" portion="PfP"/>
                  <nac name="Stabilization_Forces_In_Kosovo" portion="KFOR"/>
                  <nac name="Operation_Unified_Protector" portion="OUP"/>
               </nacs>
```