Guide to XSLs for CDSM-TDF

CDSM-TDF XSL Guide

Version 2021-JAN

January 15, 2021

Table of Contents

Chapter 1 - Introduction

1.1 - Purpose

This is an informative supplement for CDSM-TDF. This document provides XSL files developed for CDSM-TDF.

Chapter 2 - XSL Files

2.1 - AddBackNamespacesCDSM-TDF.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"
                              exclude-result-prefixes="#all"
                              version="2.0">
                  <xd:doc scope="stylesheet">
                     <xd:desc>
                        <xd:p>
                           <xd:b>Created on:</xd:b> Mar 13, 2019</xd:p>
                        <xd:p>
                           <xd:b>Author:</xd:b> bob</xd:p>
                        <xd:p>Add back CDSM-TDF namespaces after processing CDSM-TDF schema for guards.</xd:p>
                     </xd:desc>
                  </xd:doc>

                  <xsl:output method="xml" indent="yes" encoding="utf-8"/>
                  <xsl:strip-space elements="*"/>

                  <xd:doc>
                     <xd:desc>
                        <xd:p>Template match on xs:schema element</xd:p>
                     </xd:desc>
                  </xd:doc>
                  <xsl:template match="xs:schema">
                     <xsl:copy>
                        <xsl:namespace name="" select="'urn:us:gov:ic:tdf'"/>
                        <xsl:namespace name="cdsm" select="'urn:us:gov:ic:cdsmanifest'"/>
                        <xsl:namespace name="icsfhashv" select="'urn:us:gov:ic:sf:hashverification'"/>
                        <xsl:namespace name="icsf" select="'urn:us:gov:ic:sf'"/>
                        <xsl:apply-templates select="@*|node()"/>
                     </xsl:copy>
                  </xsl:template>

                  <xd:doc>
                     <xd:desc>
                        <xd:p>Identity Template</xd:p>
                     </xd:desc>
                  </xd:doc>
                  <xsl:template match="@*|node()">
                     <xsl:copy>
                        <xsl:apply-templates select="@*|node()"/>
                     </xsl:copy>
                  </xsl:template>

               </xsl:stylesheet>
```

2.2 - CDSM-TDF-Changes.xml

```
<changes xmlns:xs="http://www.w3.org/2001/XMLSchema"
                        xmlns="urn:x-us:gov:ic:cdsmanifest:changes"
                        xmlns:xhtml="http://www.w3.org/1999/xhtml-StopBrowserRendering"
                        xmlns:ch="urn:x-us:gov:ic:cdsmanifest:changes">

  <!-- The configuration file is manually edited. The documentation text below is used in the output which is a generated XSL
  for turning BASE-TDF into CSDM-TDF.-->
<documentation>
  This file is auto-generated, do not edit this file directly.

  Modified by to be made more restrictive for Cross Domain Guards and to be more open for
  non IC Corporate use in support of cross domain system manifests.  All
  changes are marked "CdsManifest:" with comments.

  CDSM-TDF-Changes.xml
</documentation>

                <!-- INFO: new feature causes null pointer exception so any namespaces needed are directly added in GenerateXSDChangeXSL.xsl
  on the <xslo:stylesheet> -->
  <!--injectNamespaces
    ch:match="xs:schema[@targetNamespace='urn:us:gov:ic:tdf']" cdsm="urn:us:gov:ic:cdsmanifest">
    <!-/- CdsManifest: Add the CDSM namespace -/->
  </injectNamespaces-->

  <replaceElement ch:match="xs:import[@namespace='urn:us:gov:ic:cvenum:tdf:signaturealgorithm']">
    <!-- CdsManifest: Remove signaturealgorithm Import since CdsManifest does not allow encryption and add CDSM schema import-->
  <xs:import namespace="urn:us:gov:ic:cdsmanifest" schemaLocation="../CDSM/CDSM.xsd"/>
                </replaceElement>

                <replaceElement ch:match="xs:import[@namespace='urn:us:gov:ic:cvenum:tdf:hashalgorithm']">
  <!-- CdsManifest: Remove hashalgorithm Import since CdsManifest does not allow encryption. -->
</replaceElement>

                <replaceElement ch:match="xs:import[@namespace='urn:us:gov:ic:cvenum:tdf:state']">
  <!-- CdsManifest: Remove tdf state Import since CdsManifest does not allow encryption. -->
</replaceElement>

                <replaceElement ch:match="xs:complexType[@name='AssertionType']/xs:sequence/xs:element[@name='StatementMetadata']">
  <!-- CdsManifest: Remove Statement Metadata on Statements. -->
</replaceElement>

                <replaceElement ch:match="xs:complexType[@name='StatementMetadataType']">
  <!-- CdsManifest: Remove StatementMetadataType on Statements. -->
</replaceElement>

                <replaceElement ch:match="xs:complexType[@name='HandlingAssertionType']">
  <!-- CdsManifest: Remove HandlingAssertionType So we can't do any classified or CUI  -->
</replaceElement>

                <replaceElement ch:match="xs:complexType[@name='HandlingStatementType']">
  <!-- CdsManifest: Remove HandlingStatementType So we can't do any classified or CUI -->
</replaceElement>
```

```
                <replaceElement ch:match="xs:element[@name='HandlingAssertion']">
  <!-- CdsManifest: Remove HandlingAssertion So we can't do any classified or CUI  -->
</replaceElement>

                <replaceElement ch:match="xs:group[@name='EncryptionInformationGroup']">
  <!-- CdsManifest: Remove EncryptionInformationGroup Nothing is allowed to be encrypted  -->
</replaceElement>

                <replaceElement ch:match="xs:group[@ref='EncryptionInformationGroup']">
  <!-- CdsManifest: Remove references to EncryptionInformationGroup Nothing is allowed to be encrypted  -->
</replaceElement>

                <replaceElement ch:match="xs:complexType[@ref='EncryptionMethodType']">
  <!-- CdsManifest: Remove references to EncryptionMethodType Nothing is allowed to be encrypted  -->
</replaceElement>

                <replaceElement ch:match="xs:element[@name='StringPayload']">
  <!-- CdsManifest: Remove references to StringPayload only ReferenceValuePayload is allowed for CdsManifest project.  -->
</replaceElement>

                <replaceElement ch:match="xs:element[@name='Base64BinaryPayload']">
  <!-- CdsManifest: Remove references to Base64BinaryPayload only ReferenceValuePayload is allowed for CdsManifest project.  -->
</replaceElement>

                <replaceElement ch:match="xs:element[@name='StructuredPayload']">
  <!-- CdsManifest: Remove references to StructuredPayload only ReferenceValuePayload is allowed for CdsManifest project.  -->
</replaceElement>

                <replaceElement ch:match="xs:element[@name='StringStatement']">
  <!-- CdsManifest: Remove references to StringStatement only StructuredStatement is allowed for CdsManifest project.  -->
</replaceElement>

                <replaceElement ch:match="xs:element[@name='Base64BinaryStatement']">
  <!-- CdsManifest: Remove references to Base64BinaryStatement only StructuredStatement is allowed for CdsManifest project.  -->
</replaceElement>

                <replaceElement ch:match="xs:element[@name='ReferenceStatement']">
  <!-- CdsManifest: Remove references to ReferenceStatement only StructuredStatement is allowed for CdsManifest project.  -->
</replaceElement>

                <replaceElement ch:match="xs:complexType[@name='Base64BinaryValueType']">
  <!-- CdsManifest: Remove references to Base64BinaryValueType All uses of this type were tailored out.  -->
</replaceElement>

                <replaceElement ch:match="xs:complexType[@name='StringValueType']">
  <!-- CdsManifest: Remove references to StringValueType All uses of this type were tailored out.  -->
</replaceElement>

                <replaceElement ch:match="xs:complexType[@name='KeyAccessType']">
  <!-- CdsManifest: Remove references to KeyAccessType All uses of this type were tailored out.  -->
</replaceElement>

                <replaceElement ch:match="xs:complexType[@name='AttachedKeyType']">
```

```
  <!-- CdsManifest: Remove references to AttachedKeyType All uses of this type were tailored out.  -->
</replaceElement>

                <replaceElement ch:match="xs:complexType[@name='PreSharedKeyType']">
  <!-- CdsManifest: Remove references to PreSharedKeyType All uses of this type were tailored out.  -->
</replaceElement>

                <replaceElement ch:match="xs:complexType[@name='RemoteKeyType']">
  <!-- CdsManifest: Remove references to RemoteKeyType All uses of this type were tailored out.  -->
</replaceElement>

                <replaceElement ch:match="xs:complexType[@name='PasswordKeyType']">
  <!-- CdsManifest: Remove references to PasswordKeyType All uses of this type were tailored out.  -->
</replaceElement>

                <replaceElement ch:match="xs:complexType[@name='WrappedPDPKeyType']">
  <!-- CdsManifest: Remove references to WrappedPDPKeyType All uses of this type were tailored out.  -->
</replaceElement>

                <replaceElement ch:match="xs:complexType[@name='WrappedKeyType']">
  <!-- CdsManifest: Remove references to WrappedKeyType All uses of this type were tailored out.  -->
</replaceElement>

                <replaceElement ch:match="xs:complexType[@name='EncryptionMethodType']">
  <!-- CdsManifest: Remove references to EncryptionMethodType All uses of this type were tailored out.  -->
</replaceElement>

                <replaceElement ch:match="xs:element[@name='TrustedDataCollection']">
  <!-- CdsManifest: Remove references to TrustedDataCollection Only TrustedDataObjects are valid for CdsManifest project.  -->
</replaceElement>

                <replaceElement ch:match="xs:complexType[@name='TdcType']">
  <!-- CdsManifest: Remove references to TrustedDataCollection All uses of this type were tailored out.  -->
</replaceElement>

                <replaceElement ch:match="xs:attribute[@name='isEncrypted']">
  <!-- CdsManifest: Remove attribute isEncrypted since nothing in CdsManifest allows encryption.  -->
</replaceElement>

                <replaceElement ch:match="xs:attribute[@ref='isEncrypted']">
  <!-- CdsManifest: Remove attribute isEncrypted since nothing in CdsManifest allows encryption.  -->
</replaceElement>

                <replaceElement ch:match="xs:attribute[@name='filename']">
  <!-- CdsManifest: Remove attribute filename since nothing it was only used for Base64 and String which were tailored out.  -->
</replaceElement>

                <replaceElement ch:match="xs:attribute[@ref='filename']">
  <!-- CdsManifest: Remove attribute filename since nothing it was only used for Base64 and String which were tailored out.  -->
</replaceElement>

                <replaceElement ch:match="xs:enumeration[@value='TDC']">
  <!-- CdsManifest: Remove Scope enumerations TDC since there are no TDC's allowed.  -->
</replaceElement>
```

```
                <replaceElement ch:match="xs:enumeration[@value='DESC_TDO']">
  <!-- CdsManifest: Remove Scope enumerations DESC_TDO since there are no TDC's allowed.  -->
</replaceElement>

                <replaceElement ch:match="xs:enumeration[@value='DESC_PAYL']">
  <!-- CdsManifest: Remove Scope enumerations DESC_PAYL since there are no TDC's allowed.  -->
</replaceElement>

                <replaceElement ch:match="xs:enumeration[@value='TDC_MEMBER']">
  <!-- CdsManifest: Remove Scope enumerations TDC_MEMBER since there are no TDC's allowed.  -->
</replaceElement>

                <replaceElement ch:match="xs:enumeration[@value='EXPLICIT']">
  <!-- CdsManifest: Remove Scope enumerations EXPLICIT since Explicit is not supported for CdsManifest.  -->
</replaceElement>

                <replaceElement ch:match="xs:complexType[@name='BoundValueListType']">
  <!-- CdsManifest: Remove references to BoundValueListType All uses of this type were tailored out.  -->
</replaceElement>

                <replaceElement ch:match="xs:complexType[@name='BoundValueType']">
  <!-- CdsManifest: Remove references to BoundValueType All uses of this type were tailored out.  -->
</replaceElement>

                <replaceElement ch:match="xs:element[@name='BoundValueList']">
  <!-- CdsManifest: Remove references to BoundValueList this type of binding is not supported for CdsManifest.  -->
</replaceElement>

                <replaceElement ch:match="xs:complexType[@name='ReferenceType']">
  <!-- CdsManifest: Remove references to ReferenceType this type of binding is not supported for CdsManifest.  -->
</replaceElement>

                <replaceElement ch:match="xs:complexType[@name='ReferenceListType']">
  <!-- CdsManifest: Remove references to ReferenceListType this type of binding is not supported for CdsManifest.  -->
</replaceElement>

                <replaceElement ch:match="xs:element[@name='ReferenceList']">
  <!-- CdsManifest: Remove references to ReferenceList this type of binding is not supported for CdsManifest.  -->
</replaceElement>

                <replaceElement ch:match="xs:attribute[@name='type']">
  <!-- CdsManifest: Remove the type attribute as it is not required for CdsManifest.  -->
</replaceElement>

                <!-- The following changes are to Make Xsat Happy and limit scope for CdsManifest  -->

  <!-- CdsManifest: Replace pattern for media type with specific media types.
 <xs:restriction base="xs:string">
    <xs:enumeration value="application/octet-stream"/>
    <xs:enumeration value="application/x-rpm"/>
    <xs:enumeration value="application/x-msi"/>
    <xs:enumeration value="application/x-msdownload"/>
    <xs:enumeration value="application/x-msdos-program"/>
```

```
      <xs:enumeration value="application/x-apple-diskimage"/>
    </xs:restriction>
-->
  <replaceElement ch:match="xs:simpleType[@name='MediaTypeType']/xs:restriction">
                    <xs:restriction base="xs:string">
                       <xs:annotation>
                          <xs:documentation>
         A restriction on string for the format of mediaType (i.e.
         audio/GSM) as defined in
         <xhtml:a href="http://tools.ietf.org/html/rfc4288">RFC 4288</xhtml:a>.
       </xs:documentation>
                       </xs:annotation>
                       <xs:maxLength value="256"/>
                       <xs:pattern value="[a-zA-Z]*/[a-zA-Z+-.]*"/>
                    </xs:restriction>
                 </replaceElement>

                 <replaceElement ch:match="xs:attribute[@name='normalizationMethod']">
   <!-- CdsManifest: Restrict normalizationMethod to specific allowed URIs. -->
   <xs:attribute name="normalizationMethod">
                       <xs:annotation>
                          <xs:documentation>
                             <xhtml:p ism:classification="U" ism:ownerProducer="USA">
          A URI that provides guidance on how to format the included values such as whitespace, attributes, and child
          nodes in a universally consistent manner. The normalization method is essential to prevent formatting such as
          whitespace and order from interfering with the validation of the cryptographic integrity of data.
          Assertions should explicitly declare all their namespaces at the assertion level rather than relying on those
          provided by the root node.
       </xhtml:p>
                          </xs:documentation>
                       </xs:annotation>
                       <xs:simpleType>
                          <xs:restriction base="xs:anyURI">
                             <xs:enumeration value="http://www.w3.org/TR/xml-c14n11"/>
                          </xs:restriction>
                       </xs:simpleType>
                    </xs:attribute>
                 </replaceElement>

                 <replaceElement ch:match="xs:complexType[@name='StructuredValueType']/xs:sequence/xs:any">
<!-- CdsManifest: Remove xs:any and force assertion to be cdsm:CdsManifestAssertion and keep Xsat Happy.  -->
<xs:element xmlns:cdsm="urn:us:gov:ic:cdsmanifest" ref="cdsm:CdsManifestAssertion"/>
                 </replaceElement>

                 <replaceElement ch:match="xs:attribute[@name='issuer']">
<!-- CdsManifest: Replace entirety of issuer force a max length and pattern to keep Xsat Happy.  -->
<!-- CdsManifest: Force issuer to be required since we don't use serial.  -->
<xs:attribute name="issuer" use="required">
                       <xs:simpleType>
                          <xs:restriction base="xs:string">
                             <xs:minLength value="1"/>
                             <xs:maxLength value="50"/>
                             <xs:pattern value="([a-zA-Z0-9\.\s=_-])*"/>
                          </xs:restriction>
```

```
                  </xs:simpleType>
               </xs:attribute>
            </replaceElement>

            <replaceElement ch:match="xs:attribute[@name='serial']">
  <!-- CdsManifest: Remove serial since we require issuer.  -->
</replaceElement>

            <!-- The following changes are just to Make Xsat Happy -->
<replaceElement ch:match="xs:attribute[@name='signatureAlgorithm']">
  <!-- CdsManifest: Replace entirety of signatureAlgorithm to disallow newer signing algorithms and keep Xsat Happy.  -->
  <xs:attribute name="signatureAlgorithm" use="required">
                  <xs:simpleType>
                     <xs:restriction base="xs:string">
                        <xs:enumeration value="SHA256withRSA"/>
                        <xs:enumeration value="SHA384withRSA"/>
                        <xs:enumeration value="SHA512withRSA"/>
                        <xs:enumeration value="SHA256withECDSA"/>
                        <xs:enumeration value="SHA384withECDSA"/>
                        <xs:enumeration value="SHA512withECDSA"/>
                     </xs:restriction>
                  </xs:simpleType>
               </xs:attribute>
            </replaceElement>

            <replaceElement ch:match="xs:attribute[@name='subject']">
  <!-- CdsManifest: Replace entirety of subject to enable max length and a pattern to keep Xsat Happy.  -->
   <xs:attribute name="subject" use="required">
                  <xs:simpleType>
                     <xs:restriction base="xs:string">
                        <xs:minLength value="1"/>
                        <xs:maxLength value="50"/>
                        <xs:pattern value="([a-zA-Z0-9i\*\.\s=_-])*"/>
                     </xs:restriction>
                  </xs:simpleType>
               </xs:attribute>
            </replaceElement>

            <replaceElement ch:match="xs:attribute[@name='id']">
<!-- CdsManifest: Replace entirety of id to enable max length to keep Xsat Happy.  -->
<xs:attribute name="id">
                  <xs:annotation>
                     <xs:documentation>
                        <xhtml:p ism:ownerProducer="USA" ism:classification="U">A unique local identifier
        used for binding and signing purposes. Not guaranteed to be unique across
        multiple TDC/TDOs but must be unique within a single instance of
        either.</xhtml:p>
                     </xs:documentation>
                  </xs:annotation>
                  <xs:simpleType>
                     <xs:restriction base="xs:ID">
                        <xs:maxLength value="50"/>
                     </xs:restriction>
                  </xs:simpleType>
```

```
                  </xs:attribute>
               </replaceElement>

               <replaceElement ch:match="xs:attribute[@name='idRef']">
   <!-- CdsManifest: Replace entirety of idRef to enable max length to keep Xsat Happy.  -->
   <xs:attribute name="idRef">
                     <xs:simpleType>
                        <xs:restriction base="xs:IDREF">
                           <xs:maxLength value="50"/>
                        </xs:restriction>
                     </xs:simpleType>
                  </xs:attribute>
               </replaceElement>

               <replaceElement ch:match="xs:attribute[@name='uri']">
   <!-- CdsManifest: Replace entirety of uri to enable max length to keep Xsat Happy.  -->
   <xs:attribute name="uri">
                     <xs:annotation>
                        <xs:documentation>
                           <xhtml:p ism:classification="U" ism:ownerProducer="USA">A uri expressing the
           location of the referenced material.</xhtml:p>
                        </xs:documentation>
                     </xs:annotation>
                     <xs:simpleType>
                        <xs:restriction base="xs:anyURI">
                           <xs:maxLength value="1024"/>
                        </xs:restriction>
                     </xs:simpleType>
                  </xs:attribute>
               </replaceElement>

               <injectAttributes ch:match="xs:element[@name='Binding']" maxOccurs="5">
     <!-- CdsManifest: Restrict to 5 signatures instead of unbounded . -->
   </injectAttributes>

               <!-- TODO: Bob needs to check if this is needed? -->
   <injectAttributes ch:match="xs:element[@name='Assertion']" maxOccurs="1">
     <!-- CdsManifest: Restrict to 1 assertions instead of unbounded . -->
   </injectAttributes>

               <injectAttributes ch:match="xs:attribute[@name='version']/xs:simpleType/xs:restriction/xs:pattern"
                                 value="[0-9]{6}(\.[0-9]{6})?\-CDSM\-TDF\.[0-9]{6}(\.[0-9]{6})?(\-.{1,23})?">
     <!-- CdsManifest: Update tdf:version regex to be for CDSM-TDF customization -->
   </injectAttributes>

            </changes>
```

2.3 - CDSM-TDF-Changes.xsl

```
<xsl:stylesheet xmlns:cdsm="urn:us:gov:ic:cdsmanifest"
                              xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                              xmlns="urn:x-us:gov:ic:cdsmanifest:changes"
                              xmlns:xs="http://www.w3.org/2001/XMLSchema"
                              xmlns:xhtml="http://www.w3.org/1999/xhtml-StopBrowserRendering"
                              xmlns:ch="urn:x-us:gov:ic:cdsmanifest:changes"
                              version="2.0"
                              exclude-result-prefixes="ch">
                  <xsl:output encoding="US-ASCII"/>
                  <!--present documentation before input comments--><xsl:template match="/">
                     <xsl:text xml:space="preserve">
</xsl:text>
                     <xsl:comment>
  This file is auto-generated, do not edit this file directly.

  Modified by to be made more restrictive for Cross Domain Guards and to be more open for
  non IC Corporate use in support of cross domain system manifests.  All
  changes are marked "CdsManifest:" with comments.

  CDSM-TDF-Changes.xml
</xsl:comment>
                     <xsl:text xml:space="preserve">
</xsl:text>
                     <xsl:apply-templates/>
                  </xsl:template>
                  <!--assume comments in prologue start on new lines--><xsl:template match="/comment()">
                     <xsl:text xml:space="preserve">
</xsl:text>
                     <xsl:copy/>
                  </xsl:template>
                  <!--assume document element not being matched; copy namespaces to reduce the need for declarations--><xsl:template match="/*">
                     <xsl:text xml:space="preserve">
</xsl:text>
                     <xsl:element name="{name(/*)}" namespace="{namespace-uri(/*)}">
                        <xsl:copy-of select="(/ | document(''))/*/                    namespace::*[.!='http://www.w3.org/1999/XSL/Transform'                                and .!='urn:x-
us:gov:ic:cdsmanifest:changes']"/>
                        <xsl:apply-templates select="@*|node()"/>
                     </xsl:element>
                  </xsl:template>
                  <!--Identity transform for all unmached leaf nodes--><xsl:template match="@*|text()|comment()|processing-instruction()">
                     <xsl:copy/>
                  </xsl:template>
                  <!--Identity transform for all unmached elements--><xsl:template match="*">
                     <xsl:element name="{name(.)}" namespace="{namespace-uri(.)}">
                        <xsl:apply-templates select="@*|node()"/>
                     </xsl:element>
                  </xsl:template>
                  <!--Using a template allows xml:space to be used in param--><xsl:template name="copy-content">
                     <xsl:param name="rtf"/>
                     <xsl:copy-of select="$rtf"/>
                  </xsl:template>
                  <!--Expose current element's start tag--><xsl:template name="show-old-start-tag">
```

```
                  <xsl:comment>
                     <xsl:text xml:space="preserve">Replacing:
</xsl:text>
                     <xsl:apply-templates mode="expose" select=".">
                        <xsl:with-param name="suppress" select="true()"/>
                     </xsl:apply-templates>
                     <xsl:text xml:space="preserve">
</xsl:text>
                  </xsl:comment>
               </xsl:template>
               <xsl:template name="show-old-element">
                  <xsl:comment>
                     <xsl:text xml:space="preserve">Replacing:
</xsl:text>
                     <xsl:apply-templates mode="expose" select="."/>
                     <xsl:text xml:space="preserve">
</xsl:text>
                  </xsl:comment>
               </xsl:template>
               <xsl:template mode="expose" match="*">
                  <xsl:param name="suppress" select="false()"/>
                  <xsl:value-of disable-output-escaping="yes" select="concat('&lt;',name(.))"/>
                  <xsl:for-each select="@*">
                     <xsl:value-of select="concat(' ',name(.),'=&#34;',.,'&#34;')"/>
                  </xsl:for-each>
                  <xsl:if test="not(node())">/</xsl:if>
                  <xsl:text>&gt;</xsl:text>
                  <xsl:if test="not( $suppress ) and node()">
                     <xsl:apply-templates mode="expose"/>
                     <xsl:value-of disable-output-escaping="yes" select="concat('&lt;/',name(.),'&gt;')"/>
                  </xsl:if>
               </xsl:template>
               <xsl:template mode="expose" match="comment()">
                  <xsl:value-of disable-output-escaping="yes" select="concat('&lt;!-',.,'-&gt;')"/>
               </xsl:template>
               <xsl:template mode="expose" match="processing-instruction()">
                  <xsl:value-of disable-output-escaping="yes" select="concat('&lt;?',name(.),' ',.,'?&gt;')"/>
               </xsl:template>
               <!--Specific node changes follow--><!--replaceElement--><xsl:template match="xs:import[@namespace='urn:us:gov:ic:cvenum:tdf:signaturealgorithm']">
                  <xsl:call-template name="show-old-element"/>
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove signaturealgorithm Import since CdsManifest does not allow encryption and add CDSM schema import</xsl:comment>
  <xs:import namespace="urn:us:gov:ic:cdsmanifest" schemaLocation="../CDSM/CDSM.xsd"/>
</xsl:with-param>
                  </xsl:call-template>
               </xsl:template>
               <!--replaceElement--><xsl:template match="xs:import[@namespace='urn:us:gov:ic:cvenum:tdf:hashalgorithm']">
                  <xsl:call-template name="show-old-element"/>
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove hashalgorithm Import since CdsManifest does not allow encryption. </xsl:comment>
</xsl:with-param>
                  </xsl:call-template>
```

```
                </xsl:template>
                <!--replaceElement--><xsl:template match="xs:import[@namespace='urn:us:gov:ic:cvenum:tdf:state']">
                   <xsl:call-template name="show-old-element"/>
                   <xsl:call-template name="copy-content">
                      <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove tdf state Import since CdsManifest does not allow encryption. </xsl:comment>
</xsl:with-param>
                   </xsl:call-template>
                </xsl:template>
                <!--replaceElement--><xsl:template match="xs:complexType[@name='AssertionType']/xs:sequence/xs:element[@name='StatementMetadata']">
                   <xsl:call-template name="show-old-element"/>
                   <xsl:call-template name="copy-content">
                      <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove Statement Metadata on Statements. </xsl:comment>
</xsl:with-param>
                   </xsl:call-template>
                </xsl:template>
                <!--replaceElement--><xsl:template match="xs:complexType[@name='StatementMetadataType']">
                   <xsl:call-template name="show-old-element"/>
                   <xsl:call-template name="copy-content">
                      <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove StatementMetadataType on Statements. </xsl:comment>
</xsl:with-param>
                   </xsl:call-template>
                </xsl:template>
                <!--replaceElement--><xsl:template match="xs:complexType[@name='HandlingAssertionType']">
                   <xsl:call-template name="show-old-element"/>
                   <xsl:call-template name="copy-content">
                      <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove HandlingAssertionType So we can't do any classified or CUI  </xsl:comment>
</xsl:with-param>
                   </xsl:call-template>
                </xsl:template>
                <!--replaceElement--><xsl:template match="xs:complexType[@name='HandlingStatementType']">
                   <xsl:call-template name="show-old-element"/>
                   <xsl:call-template name="copy-content">
                      <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove HandlingStatementType So we can't do any classified or CUI </xsl:comment>
</xsl:with-param>
                   </xsl:call-template>
                </xsl:template>
                <!--replaceElement--><xsl:template match="xs:element[@name='HandlingAssertion']">
                   <xsl:call-template name="show-old-element"/>
                   <xsl:call-template name="copy-content">
                      <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove HandlingAssertion So we can't do any classified or CUI  </xsl:comment>
</xsl:with-param>
                   </xsl:call-template>
                </xsl:template>
                <!--replaceElement--><xsl:template match="xs:group[@name='EncryptionInformationGroup']">
                   <xsl:call-template name="show-old-element"/>
                   <xsl:call-template name="copy-content">
                      <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove EncryptionInformationGroup Nothing is allowed to be encrypted  </xsl:comment>
```

```
</xsl:with-param>
                  </xsl:call-template>
               </xsl:template>
               <!--replaceElement--><xsl:template match="xs:group[@ref='EncryptionInformationGroup']">
                  <xsl:call-template name="show-old-element"/>
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove references to EncryptionInformationGroup Nothing is allowed to be encrypted  </xsl:comment>
</xsl:with-param>
                  </xsl:call-template>
               </xsl:template>
               <!--replaceElement--><xsl:template match="xs:complexType[@ref='EncryptionMethodType']">
                  <xsl:call-template name="show-old-element"/>
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove references to EncryptionMethodType Nothing is allowed to be encrypted  </xsl:comment>
</xsl:with-param>
                  </xsl:call-template>
               </xsl:template>
               <!--replaceElement--><xsl:template match="xs:element[@name='StringPayload']">
                  <xsl:call-template name="show-old-element"/>
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove references to StringPayload only ReferenceValuePayload is allowed for CdsManifest project.  </xsl:comment>
</xsl:with-param>
                  </xsl:call-template>
               </xsl:template>
               <!--replaceElement--><xsl:template match="xs:element[@name='Base64BinaryPayload']">
                  <xsl:call-template name="show-old-element"/>
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove references to Base64BinaryPayload only ReferenceValuePayload is allowed for CdsManifest project.  </xsl:comment>
</xsl:with-param>
                  </xsl:call-template>
               </xsl:template>
               <!--replaceElement--><xsl:template match="xs:element[@name='StructuredPayload']">
                  <xsl:call-template name="show-old-element"/>
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove references to StructuredPayload only ReferenceValuePayload is allowed for CdsManifest project.  </xsl:comment>
</xsl:with-param>
                  </xsl:call-template>
               </xsl:template>
               <!--replaceElement--><xsl:template match="xs:element[@name='StringStatement']">
                  <xsl:call-template name="show-old-element"/>
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove references to StringStatement only StructuredStatement is allowed for CdsManifest project.  </xsl:comment>
</xsl:with-param>
                  </xsl:call-template>
               </xsl:template>
               <!--replaceElement--><xsl:template match="xs:element[@name='Base64BinaryStatement']">
                  <xsl:call-template name="show-old-element"/>
                  <xsl:call-template name="copy-content">
```

```
                  <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove references to Base64BinaryStatement only StructuredStatement is allowed for CdsManifest project.  </xsl:comment>
</xsl:with-param>
               </xsl:call-template>
            </xsl:template>
            <!--replaceElement--><xsl:template match="xs:element[@name='ReferenceStatement']">
               <xsl:call-template name="show-old-element"/>
               <xsl:call-template name="copy-content">
                  <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove references to ReferenceStatement only StructuredStatement is allowed for CdsManifest project.  </xsl:comment>
</xsl:with-param>
               </xsl:call-template>
            </xsl:template>
            <!--replaceElement--><xsl:template match="xs:complexType[@name='Base64BinaryValueType']">
               <xsl:call-template name="show-old-element"/>
               <xsl:call-template name="copy-content">
                  <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove references to Base64BinaryValueType All uses of this type were tailored out.  </xsl:comment>
</xsl:with-param>
               </xsl:call-template>
            </xsl:template>
            <!--replaceElement--><xsl:template match="xs:complexType[@name='StringValueType']">
               <xsl:call-template name="show-old-element"/>
               <xsl:call-template name="copy-content">
                  <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove references to StringValueType All uses of this type were tailored out.  </xsl:comment>
</xsl:with-param>
               </xsl:call-template>
            </xsl:template>
            <!--replaceElement--><xsl:template match="xs:complexType[@name='KeyAccessType']">
               <xsl:call-template name="show-old-element"/>
               <xsl:call-template name="copy-content">
                  <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove references to KeyAccessType All uses of this type were tailored out.  </xsl:comment>
</xsl:with-param>
               </xsl:call-template>
            </xsl:template>
            <!--replaceElement--><xsl:template match="xs:complexType[@name='AttachedKeyType']">
               <xsl:call-template name="show-old-element"/>
               <xsl:call-template name="copy-content">
                  <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove references to AttachedKeyType All uses of this type were tailored out.  </xsl:comment>
</xsl:with-param>
               </xsl:call-template>
            </xsl:template>
            <!--replaceElement--><xsl:template match="xs:complexType[@name='PreSharedKeyType']">
               <xsl:call-template name="show-old-element"/>
               <xsl:call-template name="copy-content">
                  <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove references to PreSharedKeyType All uses of this type were tailored out.  </xsl:comment>
</xsl:with-param>
               </xsl:call-template>
            </xsl:template>
            <!--replaceElement--><xsl:template match="xs:complexType[@name='RemoteKeyType']">
```

```
                  <xsl:call-template name="show-old-element"/>
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove references to RemoteKeyType All uses of this type were tailored out.  </xsl:comment>
</xsl:with-param>
                  </xsl:call-template>
               </xsl:template>
               <!--replaceElement--><xsl:template match="xs:complexType[@name='PasswordKeyType']">
                  <xsl:call-template name="show-old-element"/>
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove references to PasswordKeyType All uses of this type were tailored out.  </xsl:comment>
</xsl:with-param>
                  </xsl:call-template>
               </xsl:template>
               <!--replaceElement--><xsl:template match="xs:complexType[@name='WrappedPDPKeyType']">
                  <xsl:call-template name="show-old-element"/>
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove references to WrappedPDPKeyType All uses of this type were tailored out.  </xsl:comment>
</xsl:with-param>
                  </xsl:call-template>
               </xsl:template>
               <!--replaceElement--><xsl:template match="xs:complexType[@name='WrappedKeyType']">
                  <xsl:call-template name="show-old-element"/>
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove references to WrappedKeyType All uses of this type were tailored out.  </xsl:comment>
</xsl:with-param>
                  </xsl:call-template>
               </xsl:template>
               <!--replaceElement--><xsl:template match="xs:complexType[@name='EncryptionMethodType']">
                  <xsl:call-template name="show-old-element"/>
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove references to EncryptionMethodType All uses of this type were tailored out.  </xsl:comment>
</xsl:with-param>
                  </xsl:call-template>
               </xsl:template>
               <!--replaceElement--><xsl:template match="xs:element[@name='TrustedDataCollection']">
                  <xsl:call-template name="show-old-element"/>
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove references to TrustedDataCollection Only TrustedDataObjects are valid for CdsManifest project.  </xsl:comment>
</xsl:with-param>
                  </xsl:call-template>
               </xsl:template>
               <!--replaceElement--><xsl:template match="xs:complexType[@name='TdcType']">
                  <xsl:call-template name="show-old-element"/>
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove references to TrustedDataCollection All uses of this type were tailored out.  </xsl:comment>
</xsl:with-param>
                  </xsl:call-template>
```

```
                </xsl:template>
                <!--replaceElement--><xsl:template match="xs:attribute[@name='isEncrypted']">
                   <xsl:call-template name="show-old-element"/>
                   <xsl:call-template name="copy-content">
                      <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove attribute isEncrypted since nothing in CdsManifest allows encryption.  </xsl:comment>
</xsl:with-param>
                   </xsl:call-template>
                </xsl:template>
                <!--replaceElement--><xsl:template match="xs:attribute[@ref='isEncrypted']">
                   <xsl:call-template name="show-old-element"/>
                   <xsl:call-template name="copy-content">
                      <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove attribute isEncrypted since nothing in CdsManifest allows encryption.  </xsl:comment>
</xsl:with-param>
                   </xsl:call-template>
                </xsl:template>
                <!--replaceElement--><xsl:template match="xs:attribute[@name='filename']">
                   <xsl:call-template name="show-old-element"/>
                   <xsl:call-template name="copy-content">
                      <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove attribute filename since nothing it was only used for Base64 and String which were tailored out.  </xsl:comment>
</xsl:with-param>
                   </xsl:call-template>
                </xsl:template>
                <!--replaceElement--><xsl:template match="xs:attribute[@ref='filename']">
                   <xsl:call-template name="show-old-element"/>
                   <xsl:call-template name="copy-content">
                      <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove attribute filename since nothing it was only used for Base64 and String which were tailored out.  </xsl:comment>
</xsl:with-param>
                   </xsl:call-template>
                </xsl:template>
                <!--replaceElement--><xsl:template match="xs:enumeration[@value='TDC']">
                   <xsl:call-template name="show-old-element"/>
                   <xsl:call-template name="copy-content">
                      <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove Scope enumerations TDC since there are no TDC's allowed.  </xsl:comment>
</xsl:with-param>
                   </xsl:call-template>
                </xsl:template>
                <!--replaceElement--><xsl:template match="xs:enumeration[@value='DESC_TDO']">
                   <xsl:call-template name="show-old-element"/>
                   <xsl:call-template name="copy-content">
                      <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove Scope enumerations DESC_TDO since there are no TDC's allowed.  </xsl:comment>
</xsl:with-param>
                   </xsl:call-template>
                </xsl:template>
                <!--replaceElement--><xsl:template match="xs:enumeration[@value='DESC_PAYL']">
                   <xsl:call-template name="show-old-element"/>
                   <xsl:call-template name="copy-content">
                      <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove Scope enumerations DESC_PAYL since there are no TDC's allowed.  </xsl:comment>
```

```
</xsl:with-param>
                  </xsl:call-template>
               </xsl:template>
               <!--replaceElement--><xsl:template match="xs:enumeration[@value='TDC_MEMBER']">
                  <xsl:call-template name="show-old-element"/>
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove Scope enumerations TDC_MEMBER since there are no TDC's allowed.  </xsl:comment>
</xsl:with-param>
                  </xsl:call-template>
               </xsl:template>
               <!--replaceElement--><xsl:template match="xs:enumeration[@value='EXPLICIT']">
                  <xsl:call-template name="show-old-element"/>
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove Scope enumerations EXPLICIT since Explicit is not supported for CdsManifest.  </xsl:comment>
</xsl:with-param>
                  </xsl:call-template>
               </xsl:template>
               <!--replaceElement--><xsl:template match="xs:complexType[@name='BoundValueListType']">
                  <xsl:call-template name="show-old-element"/>
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove references to BoundValueListType All uses of this type were tailored out.  </xsl:comment>
</xsl:with-param>
                  </xsl:call-template>
               </xsl:template>
               <!--replaceElement--><xsl:template match="xs:complexType[@name='BoundValueType']">
                  <xsl:call-template name="show-old-element"/>
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove references to BoundValueType All uses of this type were tailored out.  </xsl:comment>
</xsl:with-param>
                  </xsl:call-template>
               </xsl:template>
               <!--replaceElement--><xsl:template match="xs:element[@name='BoundValueList']">
                  <xsl:call-template name="show-old-element"/>
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove references to BoundValueList this type of binding is not supported for CdsManifest.  </xsl:comment>
</xsl:with-param>
                  </xsl:call-template>
               </xsl:template>
               <!--replaceElement--><xsl:template match="xs:complexType[@name='ReferenceType']">
                  <xsl:call-template name="show-old-element"/>
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove references to ReferenceType this type of binding is not supported for CdsManifest.  </xsl:comment>
</xsl:with-param>
                  </xsl:call-template>
               </xsl:template>
               <!--replaceElement--><xsl:template match="xs:complexType[@name='ReferenceListType']">
                  <xsl:call-template name="show-old-element"/>
                  <xsl:call-template name="copy-content">
```

```
                  <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove references to ReferenceListType this type of binding is not supported for CdsManifest.  </xsl:comment>
</xsl:with-param>
                  </xsl:call-template>
               </xsl:template>
               <!--replaceElement--><xsl:template match="xs:element[@name='ReferenceList']">
                  <xsl:call-template name="show-old-element"/>
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove references to ReferenceList this type of binding is not supported for CdsManifest.  </xsl:comment>
</xsl:with-param>
                  </xsl:call-template>
               </xsl:template>
               <!--replaceElement--><xsl:template match="xs:attribute[@name='type']">
                  <xsl:call-template name="show-old-element"/>
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove the type attribute as it is not required for CdsManifest.  </xsl:comment>
</xsl:with-param>
                  </xsl:call-template>
               </xsl:template>
               <!--replaceElement--><xsl:template match="xs:simpleType[@name='MediaTypeType']/xs:restriction">
                  <xsl:call-template name="show-old-element"/>
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xs:restriction base="xs:string">
    <xs:annotation>
      <xs:documentation>
        A restriction on string for the format of mediaType (i.e.
        audio/GSM) as defined in
        <xhtml:a href="http://tools.ietf.org/html/rfc4288">RFC 4288</xhtml:a>.
      </xs:documentation>
    </xs:annotation>
    <xs:maxLength value="256"/>
    <xs:pattern value="[a-zA-Z]*/[a-zA-Z+-.]*"/>
  </xs:restriction>
</xsl:with-param>
                  </xsl:call-template>
               </xsl:template>
               <!--replaceElement--><xsl:template match="xs:attribute[@name='normalizationMethod']">
                  <xsl:call-template name="show-old-element"/>
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Restrict normalizationMethod to specific allowed URIs. </xsl:comment>
  <xs:attribute name="normalizationMethod">
    <xs:annotation>
      <xs:documentation>
        <xhtml:p ism:classification="U" ism:ownerProducer="USA">
          A URI that provides guidance on how to format the included values such as whitespace, attributes, and child
          nodes in a universally consistent manner. The normalization method is essential to prevent formatting such as
          whitespace and order from interfering with the validation of the cryptographic integrity of data.
          Assertions should explicitly declare all their namespaces at the assertion level rather than relying on those
          provided by the root node.
        </xhtml:p>
```

```
      </xs:documentation>
    </xs:annotation>
    <xs:simpleType>
      <xs:restriction base="xs:anyURI">
        <xs:enumeration value="http://www.w3.org/TR/xml-c14n11"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xsl:with-param>
                  </xsl:call-template>
               </xsl:template>
               <!--replaceElement--><xsl:template match="xs:complexType[@name='StructuredValueType']/xs:sequence/xs:any">
                  <xsl:call-template name="show-old-element"/>
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove xs:any and force assertion to be cdsm:CdsManifestAssertion and keep Xsat Happy.  </xsl:comment>
  <xs:element ref="cdsm:CdsManifestAssertion"/>
</xsl:with-param>
                  </xsl:call-template>
               </xsl:template>
               <!--replaceElement--><xsl:template match="xs:attribute[@name='issuer']">
                  <xsl:call-template name="show-old-element"/>
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Replace entirety of issuer force a max length and pattern to keep Xsat Happy.  </xsl:comment>
  <xsl:comment> CdsManifest: Force issuer to be required since we don't use serial.  </xsl:comment>
  <xs:attribute name="issuer" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:minLength value="1"/>
        <xs:maxLength value="50"/>
        <xs:pattern value="([a-zA-Z0-9\.\s=_-])*"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xsl:with-param>
                  </xsl:call-template>
               </xsl:template>
               <!--replaceElement--><xsl:template match="xs:attribute[@name='serial']">
                  <xsl:call-template name="show-old-element"/>
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Remove serial since we require issuer.  </xsl:comment>
</xsl:with-param>
                  </xsl:call-template>
               </xsl:template>
               <!--replaceElement--><xsl:template match="xs:attribute[@name='signatureAlgorithm']">
                  <xsl:call-template name="show-old-element"/>
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Replace entirety of signatureAlgorithm to disallow newer signing algorithms and keep Xsat Happy.  </xsl:comment>
  <xs:attribute name="signatureAlgorithm" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string">
```

```
        <xs:enumeration value="SHA256withRSA"/>
        <xs:enumeration value="SHA384withRSA"/>
        <xs:enumeration value="SHA512withRSA"/>
        <xs:enumeration value="SHA256withECDSA"/>
        <xs:enumeration value="SHA384withECDSA"/>
        <xs:enumeration value="SHA512withECDSA"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xsl:with-param>
                  </xsl:call-template>
               </xsl:template>
               <!--replaceElement--><xsl:template match="xs:attribute[@name='subject']">
                  <xsl:call-template name="show-old-element"/>
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Replace entirety of subject to enable max length and a pattern to keep Xsat Happy.  </xsl:comment>
   <xs:attribute name="subject" use="required">
   <xs:simpleType>
     <xs:restriction base="xs:string">
       <xs:minLength value="1"/>
       <xs:maxLength value="50"/>
       <xs:pattern value="([a-zA-Z0-9i\*\.\s=_-])*"/>
     </xs:restriction>
   </xs:simpleType>
 </xs:attribute>
</xsl:with-param>
                  </xsl:call-template>
</xsl:with-param>
                  </xsl:call-template>
               </xsl:template>
               <!--replaceElement--><xsl:template match="xs:attribute[@name='id']">
                  <xsl:call-template name="show-old-element"/>
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Replace entirety of id to enable max length to keep Xsat Happy.  </xsl:comment>
  <xs:attribute name="id">
    <xs:annotation>
      <xs:documentation>
        <xhtml:p ism:ownerProducer="USA" ism:classification="U">A unique local identifier
          used for binding and signing purposes. Not guaranteed to be unique across
          multiple TDC/TDOs but must be unique within a single instance of
          either.</xhtml:p>
      </xs:documentation>
    </xs:annotation>
    <xs:simpleType>
      <xs:restriction base="xs:ID">
        <xs:maxLength value="50"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xsl:with-param>
                  </xsl:call-template>
               </xsl:template>
               <!--replaceElement--><xsl:template match="xs:attribute[@name='idRef']">
                  <xsl:call-template name="show-old-element"/>
```

```
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Replace entirety of idRef to enable max length to keep Xsat Happy.  </xsl:comment>
  <xs:attribute name="idRef">
    <xs:simpleType>
      <xs:restriction base="xs:IDREF">
        <xs:maxLength value="50"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xsl:with-param>
                  </xsl:call-template>
               </xsl:template>
               <!--replaceElement--><xsl:template match="xs:attribute[@name='uri']">
                  <xsl:call-template name="show-old-element"/>
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Replace entirety of uri to enable max length to keep Xsat Happy.  </xsl:comment>
  <xs:attribute name="uri">
    <xs:annotation>
      <xs:documentation>
        <xhtml:p ism:classification="U" ism:ownerProducer="USA">A uri expressing the
          location of the referenced material.</xhtml:p>
      </xs:documentation>
    </xs:annotation>
    <xs:simpleType>
      <xs:restriction base="xs:anyURI">
        <xs:maxLength value="1024"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xsl:with-param>
                  </xsl:call-template>
               </xsl:template>
               <!--injectAttribute--><xsl:template match="xs:element[@name='Binding']">
                  <xsl:call-template name="show-old-start-tag"/>
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Restrict to 5 signatures instead of unbounded . </xsl:comment>
</xsl:with-param>
                  </xsl:call-template>
                  <xsl:copy>
                     <xsl:apply-templates select="@*"/>
                     <xsl:attribute name="maxOccurs" namespace="">5</xsl:attribute>
                     <xsl:apply-templates select="node()"/>
                  </xsl:copy>
               </xsl:template>
               <!--injectAttribute--><xsl:template match="xs:element[@name='Assertion']">
                  <xsl:call-template name="show-old-start-tag"/>
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Restrict to 1 assertions instead of unbounded . </xsl:comment>
</xsl:with-param>
                  </xsl:call-template>
```

```
                  <xsl:copy>
                     <xsl:apply-templates select="@*"/>
                     <xsl:attribute name="maxOccurs" namespace="">1</xsl:attribute>
                     <xsl:apply-templates select="node()"/>
                  </xsl:copy>
               </xsl:template>
               <!--injectAttribute--><xsl:template match="xs:attribute[@name='version']/xs:simpleType/xs:restriction/xs:pattern">
                  <xsl:call-template name="show-old-start-tag"/>
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> CdsManifest: Update tdf:version regex to be for CDSM-TDF customization </xsl:comment>
</xsl:with-param>
                  </xsl:call-template>
                  <xsl:copy>
                     <xsl:apply-templates select="@*"/>
                     <xsl:attribute name="value" namespace="">[0-9]{6}(\.[0-9]{6})?\-CDSM\-TDF\.[0-9]{6}(\.[0-9]{6})?(\-.{1,23})?</xsl:attribute>
                     <xsl:apply-templates select="node()"/>
                  </xsl:copy>
               </xsl:template>
            </xsl:stylesheet>
```

2.4 - GenerateXSDChangeXSL.xsl

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                              xmlns:ch="urn:x-us:gov:ic:cdsmanifest:changes"
                              xmlns:xslo="dummy"
                              exclude-result-prefixes="ch"
                              version="2.0">
<!--
     A stylesheet to convert XPath expressions of changes into a stylesheet
     to actually perform the changes on an input file.

     $Id: changes2xsl.xsl,v 1.4 2006/04/05 19:31:17 holmank Exp $
-->

<xsl:namespace-alias stylesheet-prefix="xslo" result-prefix="xsl"/>

                  <xsl:output indent="yes" encoding="US-ASCII"/>

                  <!--============================================================================-->
<!--start the result stylesheet-->

<xsl:template match="ch:changes">
  <!-- NOTE: add namespaces that we need to propagate to xslo:stylesheet tag -->
  <xslo:stylesheet xmlns:cdsm="urn:us:gov:ic:cdsmanifest">
    <!--reduce namespace declarations by doing this up front-->
    <xsl:copy-of select="namespace::*"/>
                        <xsl:attribute name="version">2.0</xsl:attribute>
                        <xsl:attribute name="exclude-result-prefixes">
                           <xsl:for-each select="namespace::*[.='urn:x-us:gov:ic:cdsmanifest:changes']">
                              <xsl:value-of select="name(.)"/>
                           </xsl:for-each>
                        </xsl:attribute>
                        <xslo:output encoding="US-ASCII"/>
                        <xsl:text>
</xsl:text>
                        <xsl:comment>present documentation before input comments</xsl:comment>
                        <xslo:template match="/">
                           <xsl:if test="ch:documentation">
                              <xslo:text xml:space="preserve">
</xslo:text>
                              <xslo:comment>
                                 <xsl:value-of select="ch:documentation"/>
                              </xslo:comment>
                              <xslo:text xml:space="preserve">
</xslo:text>
                           </xsl:if>
                           <xslo:apply-templates/>
                        </xslo:template>

                        <xsl:text>
</xsl:text>
                        <xsl:comment>assume comments in prologue start on new lines</xsl:comment>
                        <xslo:template match="/comment()">
                           <xslo:text xml:space="preserve">
```

```
</xslo:text>
                        <xslo:copy/>
                     </xslo:template>

                     <xsl:text>
</xsl:text>
                     <xsl:comment>assume document element not being matched; copy namespaces to reduce the need for declarations</xsl:comment>
                     <xsl:text>
</xsl:text>
                     <xslo:template match="/*">
                        <xslo:text xml:space="preserve">
</xslo:text>
                        <xslo:element name="{{name(/*)}}" namespace="{{namespace-uri(/*)}}">
                           <xslo:copy-of select="(/ | document(''))/*/                    namespace::*[.!='http://www.w3.org/1999/XSL/Transform'                    and .!
='urn:x-us:gov:ic:cdsmanifest:changes']"/>
                           <xslo:apply-templates select="@*|node()"/>
                        </xslo:element>
                     </xslo:template>

                     <xsl:text>
</xsl:text>
                     <xsl:comment>Identity transform for all unmached leaf nodes</xsl:comment>
                     <xsl:text>
</xsl:text>
                     <xslo:template match="@*|text()|comment()|processing-instruction()">
                        <xslo:copy/>
                     </xslo:template>
                     <xsl:text>
                     </xslo:template>
                     <xsl:text>
</xsl:text>
                     <xsl:comment>Identity transform for all unmached elements</xsl:comment>
                     <xsl:text>
</xsl:text>
                     <xslo:template match="*">
                        <xslo:element name="{{name(.)}}" namespace="{{namespace-uri(.)}}">
                           <xslo:apply-templates select="@*|node()"/>
                        </xslo:element>
                     </xslo:template>

                     <xsl:text>
</xsl:text>
                     <xsl:comment>Using a template allows xml:space to be used in param</xsl:comment>
                     <xsl:text>
</xsl:text>
                     <xslo:template name="copy-content">
                        <xslo:param name="rtf"/>
                        <xslo:copy-of select="$rtf"/>
                     </xslo:template>

                     <xsl:text>
</xsl:text>
                     <xsl:comment>Expose current element's start tag</xsl:comment>
                     <xsl:text>
</xsl:text>
                     <xslo:template name="show-old-start-tag">
```

```
                        <xslo:comment>
                           <xslo:text xml:space="preserve">Replacing:
</xslo:text>
                           <xslo:apply-templates mode="expose" select=".">
                              <xslo:with-param name="suppress" select="true()"/>
                           </xslo:apply-templates>
                           <xslo:text xml:space="preserve">
</xslo:text>
                        </xslo:comment>
                     </xslo:template>
                     <xslo:template name="show-old-element">
                        <xslo:comment>
                           <xslo:text xml:space="preserve">Replacing:
</xslo:text>
                           <xslo:apply-templates mode="expose" select="."/>
                           <xslo:text xml:space="preserve">
</xslo:text>
                        </xslo:comment>
                     </xslo:template>
                     <xslo:template mode="expose" match="*">
                        <xslo:param name="suppress" select="false()"/>
                        <xslo:value-of disable-output-escaping="yes" select="concat('&lt;',name(.))"/>
                        <xslo:for-each select="@*">
                           <xslo:value-of select="concat(' ',name(.),'=&#34;',.,'&#34;')"/>
                        </xslo:for-each>
                        <xslo:if test="not(node())">/</xslo:if>
                        <xslo:text>&gt;</xslo:text>
                        <xslo:if test="not( $suppress ) and node()">
                           <xslo:apply-templates mode="expose"/>
                           <xslo:value-of disable-output-escaping="yes" select="concat('&lt;/',name(.),'&gt;')"/>
                        </xslo:if>
                     </xslo:template>
                     <xslo:template mode="expose" match="comment()">
                        <xslo:value-of disable-output-escaping="yes" select="concat('&lt;!-',.,'-&gt;')"/>
                     </xslo:template>
                     <xslo:template mode="expose" match="processing-instruction()">
                        <xslo:value-of disable-output-escaping="yes" select="concat('&lt;?',name(.),' ',.,'?&gt;')"/>
                     </xslo:template>

                     <xsl:text>
</xsl:text>
                     <xsl:comment>Specific node changes follow</xsl:comment>
                     <xsl:text>
</xsl:text>
                     <xsl:apply-templates/>
                  </xslo:stylesheet>
               </xsl:template>

               <!--============================================================================-->
<!--copying the source tree content into the output stylesheet as stylesheet
    directives; requires preserving comments using a comment instruction-->

<xsl:template mode="copy-content" match="comment()" priority="2">
                  <xslo:comment>
```

```
                     <xsl:value-of select="."/>
                  </xslo:comment>
               </xsl:template>

               <xsl:template mode="copy-content" match="@*|node()">
                  <xsl:copy>
                     <xsl:apply-templates mode="copy-content" select="@*|node()"/>
                  </xsl:copy>
               </xsl:template>

               <!--================================================================================-->
<!--different directives-->

<!--documentation already handled-->
<xsl:template match="ch:documentation"/>

               <!--add content after the element being matched-->
<xsl:template match="ch:injectAfter">
                  <xsl:call-template name="checkAttributes"/>
                  <xsl:comment>injectAfter</xsl:comment>
                  <xslo:template match="{@ch:match}">
                     <xslo:copy>
                        <xslo:apply-templates select="@*|node()"/>
                     </xslo:copy>
                     <xslo:call-template name="copy-content">
                        <xslo:with-param name="rtf">
                           <xsl:attribute name="xml:space">preserve</xsl:attribute>
                           <xsl:apply-templates mode="copy-content" select="node()"/>
                        </xslo:with-param>
                     </xslo:call-template>
                  </xslo:template>
               </xsl:template>

               <!--add content before the element being matched-->
<xsl:template match="ch:injectBefore">
                  <xsl:call-template name="checkAttributes"/>
                  <xsl:comment>injectBefore</xsl:comment>
                  <xslo:template match="{@ch:match}">
                     <xslo:call-template name="copy-content">
                        <xslo:with-param name="rtf">
                           <xsl:attribute name="xml:space">preserve</xsl:attribute>
                           <xsl:apply-templates mode="copy-content" select="node()"/>
                        </xslo:with-param>
                     </xslo:call-template>
                     <xslo:copy>
                        <xslo:apply-templates select="@*|node()"/>
                     </xslo:copy>
                  </xslo:template>
               </xsl:template>

               <!--add content at the beginning of the element being matched-->
<xsl:template match="ch:injectStart">
                  <xsl:call-template name="checkAttributes"/>
                  <xsl:comment>injectStart</xsl:comment>
```

```
                  <xslo:template match="{@ch:match}">
                     <xslo:copy>
                        <xslo:apply-templates select="@*"/>
                        <xslo:call-template name="copy-content">
                           <xslo:with-param name="rtf">
                              <xsl:attribute name="xml:space">preserve</xsl:attribute>
                              <xsl:apply-templates mode="copy-content" select="node()"/>
                           </xslo:with-param>
                        </xslo:call-template>
                        <xslo:apply-templates select="node()"/>
                     </xslo:copy>
                  </xslo:template>
               </xsl:template>

               <!--add content at the end of the element being matched-->
<xsl:template match="ch:injectEnd">
                  <xsl:call-template name="checkAttributes"/>
                  <xsl:comment>injectEnd</xsl:comment>
                  <xslo:template match="{@ch:match}">
                     <xslo:copy>
                        <xslo:apply-templates select="@*|node()"/>
                        <xslo:call-template name="copy-content">
                           <xslo:with-param name="rtf">
                              <xsl:attribute name="xml:space">preserve</xsl:attribute>
                              <xsl:apply-templates mode="copy-content" select="node()"/>
                           </xslo:with-param>
                        </xslo:call-template>
                     </xslo:copy>
                  </xslo:template>
               </xsl:template>

               <!--add attributes to the element being matched-->
<xsl:template match="ch:injectAttributes">
                  <xsl:call-template name="checkAttributes"/>
                  <xsl:comment>injectAttribute</xsl:comment>
                  <xslo:template match="{@ch:match}">
                     <xslo:call-template name="show-old-start-tag"/>
                     <xslo:call-template name="copy-content">
                        <xslo:with-param name="rtf">
                           <xsl:attribute name="xml:space">preserve</xsl:attribute>
                           <xsl:apply-templates mode="copy-content" select="text()|comment()"/>
                        </xslo:with-param>
                     </xslo:call-template>
                     <xslo:copy>
                        <xslo:apply-templates select="@*"/>
                        <xsl:for-each select="@*[count(../@ch:* | .) != count( ../@ch:* )]">
                           <xslo:attribute name="{name(.)}" namespace="{namespace-uri(.)}">
                              <xsl:value-of select="."/>
                           </xslo:attribute>
                        </xsl:for-each>
                        <xslo:apply-templates select="node()"/>
                     </xslo:copy>
                  </xslo:template>
               </xsl:template>
```

```
                <!--add namespace to the element being matched-->
 <!-- <xsl:template match="ch:injectNamespaces">
    <xsl:call-template name="checkAttributes"/>
    <xsl:comment>injectNamespace</xsl:comment>
    <xslo:template match="{@ch:match}">
      <xslo:call-template name="show-old-start-tag"/>
      <xslo:call-template name="copy-content">
        <xslo:with-param name="rtf">
          <xsl:attribute name="xml:space">preserve</xsl:attribute>
          <xsl:apply-templates mode="copy-content" select="text()|comment()"/>
        </xslo:with-param>
      </xslo:call-template>
      <xslo:copy>
        <xslo:apply-templates select="@*"/>
        <xsl:for-each select="@*[count(../@ch:* | .) != count( ../@ch:* )]">
          <xslo:namespace name="{name(.)}" select="'{.}'"/>
        </xsl:for-each>
        <xslo:apply-templates select="node()"/>
      </xslo:copy>
    </xslo:template>
  </xsl:template>-->

<!--replace the element being matched-->
<xsl:template match="ch:replaceElement">
                    <xsl:call-template name="checkAttributes"/>
                    <xsl:comment>replaceElement</xsl:comment>
                    <xslo:template match="{@ch:match}">
                       <xslo:call-template name="show-old-element"/>
                       <xslo:call-template name="copy-content">
                          <xslo:with-param name="rtf">
                             <xsl:attribute name="xml:space">preserve</xsl:attribute>
                             <xsl:apply-templates mode="copy-content" select="node()"/>
                          </xslo:with-param>
                       </xslo:call-template>
                    </xslo:template>
                 </xsl:template>

                 <!--report missing required attribute-->
<xsl:template name="checkAttributes">
                    <xsl:if test="not(@ch:match)">
                       <xsl:message terminate="yes">
                          <xsl:text>Missing {urn:x-us:gov:ic:cdsmanifest:changes}match= for </xsl:text>
                          <xsl:text/>element: <xsl:value-of select="name(.)"/>
                       </xsl:message>
                    </xsl:if>
                 </xsl:template>

                 <!--unexpected element-->
<xsl:template match="*">
                    <xsl:message terminate="yes">
                       <xsl:text/>Unexpected element: <xsl:value-of select="name(.)"/>
                    </xsl:message>
                 </xsl:template>
```

```
            </xsl:stylesheet>
```

2.5 - Identity.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"
                           exclude-result-prefixes="#all"
                           version="2.0">
               <xd:doc scope="stylesheet">
                  <xd:desc>
                     <xd:p>
                        <xd:b>Created on:</xd:b> Nov 27, 2019</xd:p>
                     <xd:p>
                        <xd:b>Author:</xd:b> dagon</xd:p>
                     <xd:p>Identity XSL needed to fix xmlns="".</xd:p>
                  </xd:desc>
               </xd:doc>

               <xsl:output method="xml" indent="yes" encoding="utf-8"/>
               <xsl:strip-space elements="*"/>

               <xsl:template match="@*|node()">
                  <xsl:copy copy-namespaces="yes">
                     <xsl:apply-templates select="@*|node()"/>
                  </xsl:copy>
               </xsl:template>
            </xsl:stylesheet>
```

2.6 - ProcessForGuards.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"
                              exclude-result-prefixes="#all"
                              version="2.0">
                  <xd:doc scope="stylesheet">
                     <xd:desc>
                        <xd:p>
                           <xd:b>Created on:</xd:b> Mar 13, 2019</xd:p>
                        <xd:p>
                           <xd:b>Author:</xd:b> bob</xd:p>
                        <xd:p>Remove comments, PIs annotations, defaults (minOccurs="1" or maxOccurs="1"), ISM attributes to make the smallest simple file for a guard.</xd:p>
                     </xd:desc>
                  </xd:doc>

                  <xsl:output method="xml" indent="yes" encoding="utf-8"/>
                  <xsl:strip-space elements="*"/>

                  <xd:doc>
                     <xd:desc>Add processing instruction to skip the unit test that checks for proper Processing Instructions on schema files.</xd:desc>
                  </xd:doc>
                  <xsl:template match="/*">
                     <xsl:processing-instruction name="skip-unit-test-PI-validation"/>
                     <xsl:copy copy-namespaces="no">
                        <xsl:apply-templates select="@*|node()"/>
                     </xsl:copy>
                  </xsl:template>

                  <xd:doc>
                     <xd:desc>Identity Transform</xd:desc>
                  </xd:doc>
                  <xsl:template match="@*|node()">
                     <xsl:copy copy-namespaces="no">
                        <xsl:apply-templates select="@*|node()"/>
                     </xsl:copy>
                  </xsl:template>

                  <xd:doc>
                     <xd:desc>
                        <xd:p>Strip out all @minOccurs and @maxOccurs whose value is 1 as those are defined by XSD specification to be the default.</xd:p>
                     </xd:desc>
                  </xd:doc>
                  <xsl:template match="@minOccurs[.=1] | @maxOccurs[.=1]"/>

                  <xd:doc>
                     <xd:desc>
                        <xd:p>Strip out all @ism:* as the ISM attributes are only there for describing the DES</xd:p>
                     </xd:desc>
                  </xd:doc>
                  <xsl:template match="@ism:*"/>

                  <xd:doc>
```

```
            <xd:desc>
               <xd:p>Strip out all comments, PIs and xs:annotations.</xd:p>
            </xd:desc>
         </xd:doc>
         <xsl:template match="comment()|processing-instruction()|xs:annotation" priority="5"/>
      </xsl:stylesheet>
```

2.7 - UpdateXSDDocumentation.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:xhtml="http://www.w3.org/1999/xhtml-StopBrowserRendering"
                              exclude-result-prefixes="xs xd"
                              version="2.0">
                  <xd:doc scope="stylesheet">
                     <xd:desc>
                        <xd:p>
                           <xd:b>Created on:</xd:b> Sep 25, 2019</xd:p>
                        <xd:p>
                           <xd:b>Author:</xd:b> dagon</xd:p>
                        <xd:p>Update the header and footer annotation documentation in the generated CDSM-TDF XSD.</xd:p>
                     </xd:desc>
                  </xd:doc>

                  <xsl:output method="xml" indent="yes" encoding="utf-8"/>
                  <xsl:strip-space elements="*"/>

                  <xsl:template match="@*|node()">
                     <xsl:copy>
                        <xsl:apply-templates select="@*|node()"/>
                     </xsl:copy>
                  </xsl:template>

                  <xsl:template match="/">
                     <xsl:choose>
                        <xsl:when test="not(.//xs:annotation[.//xhtml:a/@href = '../../Documents/BASE-TDF/DesBaseTDFXml.pdf'])">
                           <xsl:call-template name="AbortDueToMissingHeaderAnnotations"/>
                        </xsl:when>
                        <xsl:when test="not(.//xs:annotation[.//xhtml:h2/text() = 'Formal Change List'])">
                           <xsl:call-template name="AbortDueToMissingFooterAnnotations"/>
                        </xsl:when>
                        <xsl:otherwise/>
                     </xsl:choose>
                     <xsl:apply-templates/>
                  </xsl:template>

                  <xsl:template name="AbortDueToMissingHeaderAnnotations">
                     <xsl:message terminate="yes">
                        <xsl:value-of select="'Missing Expected Header Annotations.'"/>
                     </xsl:message>
                  </xsl:template>

                  <xsl:template name="AbortDueToMissingFooterAnnotations">
                     <xsl:message terminate="yes">
                        <xsl:value-of select="'Missing Expected Footer Annotations.'"/>
                     </xsl:message>
                  </xsl:template>

                  <!-- Update annotation documentation header (description, introduction, implementation notes section)-->
   <xsl:template match="xs:annotation[.//xhtml:a/@href = '../../Documents/BASE-TDF/DesBaseTDFXml.pdf']">
```

```
            <xs:annotation>
               <xs:documentation>
                  <xhtml:h1 ism:ownerProducer="USA" ism:classification="U">Intelligence Community
            Technical Specification XML Data Encoding Specification for Cross Domain System Manifest TDF
            (CDSM-TDF.XML)</xhtml:h1>
               </xs:documentation>
               <xs:documentation>
                  <xhtml:h2 ism:ownerProducer="USA" ism:classification="U">Notices</xhtml:h2>
                  <xhtml:p ism:ownerProducer="USA" ism:classification="U">Distribution Notice:
    This document has been approved for Public Release and is available for use without restriction.
</xhtml:p>
               </xs:documentation>
               <xs:documentation>
                  <xhtml:h2 ism:ownerProducer="USA" ism:classification="U">Description</xhtml:h2>
                  <xhtml:p ism:ownerProducer="USA" ism:classification="U">W3C XML Schema for the Intelligence Community
            Technical Specification XML Data Encoding Specification for Cross Domain System Manifest TDF
            (CDSM-TDF.XML).</xhtml:p>
               </xs:documentation>
               <xs:documentation>
                  <xhtml:h2 ism:ownerProducer="USA" ism:classification="U">Introduction</xhtml:h2>
                  <xhtml:p ism:ownerProducer="USA" ism:classification="U">This XML Schema file is one
            component of the XML Data Encoding Specification (DES). Please see the document titled<xhtml:i>
                        <xhtml:a href="../../Documents/CDSM-TDF/DesCdsmTdfXml.pdf">XML Data Encoding Specification for Cross Domain System Manifest TDF</xhtml:a>
                     </xhtml:i>for a complete description of the encoding as well as list of all
            components.</xhtml:p>
                  <xhtml:p ism:ownerProducer="USA" ism:classification="U">It is envisioned that this
            schema 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 XML Schema and the other parts of the DES for any purpose is hereby granted in
            perpetuity.</xhtml:p>
                  <xhtml:p ism:ownerProducer="USA" ism:classification="U">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.</xhtml:p>
                  <xhtml:p ism:ownerProducer="USA" ism:classification="U">If you modify this XML Schema in
            any way label your schema as a variant of CDSM-TDF.XML.</xhtml:p>
                  <xhtml:p ism:ownerProducer="USA" ism:classification="U">Please direct all questions, bug
            reports,or suggestions for changes to the points of contact identified in the
            document referenced above.</xhtml:p>
               </xs:documentation>
               <xs:documentation>
                  <xhtml:h2 ism:ownerProducer="USA" ism:classification="U">Implementation Notes</xhtml:h2>
                  <xhtml:p ism:ownerProducer="USA" ism:classification="U">The root element for CDSM-TDF is:
            <xhtml:a href="CDSM-TDF_xsd_Element_TrustedDataObject.html#TrustedDataObject">tdf:TrustedDataObject</xhtml:a>
                  </xhtml:p>
               </xs:documentation>
               <xs:documentation>
                  <xhtml:h2 ism:ownerProducer="USA" ism:classification="U">Creators</xhtml:h2>
                  <xhtml:p ism:ownerProducer="USA" ism:classification="U">Office of the Director of
            National Intelligence Intelligence Community Chief Information Officer</xhtml:p>
               </xs:documentation>
            </xs:annotation>
         </xsl:template>
```

```
            <!-- Update annotation documentation footer (change history section) -->
<xsl:template match="xs:annotation[.//xhtml:h2/text() = 'Formal Change List']">
               <xs:annotation>
                  <xs:documentation>
                     <xhtml:h2 ism:ownerProducer="USA" ism:classification="U">Formal Change List</xhtml:h2>
                     <xhtml:table ism:ownerProducer="USA" ism:classification="U" id="ChangeHistory">
                        <xhtml:caption>Change History</xhtml:caption>
                        <xhtml:thead>
                           <xhtml:tr>
                              <xhtml:th>Version</xhtml:th>
                              <xhtml:th>Date</xhtml:th>
                              <xhtml:th>By</xhtml:th>
                              <xhtml:th>Description</xhtml:th>
                           </xhtml:tr>
                        </xhtml:thead>
                        <xhtml:tbody>
                           <xhtml:tr>
                              <xhtml:td>2021-JAN</xhtml:td>
                              <xhtml:td>2020-11-17</xhtml:td>
                              <xhtml:td>ODNI/OCIO/ICEA</xhtml:td>
                              <xhtml:td>
                                 <xhtml:ul>
                                    <xhtml:li ism:ownerProducer="USA" ism:classification="U">
                                     Reference the change history in the DES.</xhtml:li>
                                 </xhtml:ul>
                              </xhtml:td>
                           </xhtml:tr>
                        </xhtml:tbody>
                     </xhtml:table>
                  </xs:documentation>
               </xs:annotation>
            </xsl:template>

         </xsl:stylesheet>
```