Guide to XSLs for DHZMC-TDF

DHZMC-TDF XSL Guide

Version 2021-JAN

January 15, 2021

Table of Contents

Chapter 1 - Introduction

1.1 - Purpose

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

Chapter 2 - XSL Files

2.1 - AddBackNamespacesDHZMC-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 DHZMC-TDF namespaces after processing DHZMC-TDF schema for guards.</xd:p>
                     </xd:desc>
                  </xd:doc>

                  <xsl:output method="xml" indent="yes" encoding="US-ASCII"/>
                  <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="dhzm" select="'urn:us:gov:ic:digitalhazmat'"/>
                        <xsl:namespace name="icsfhashv" select="'urn:us:gov:ic:sf:hashverification'"/>
                        <xsl:namespace name="icsf" select="'urn:us:gov:ic:sf'"/>
                        <xsl:namespace name="tdfsigal" select="'urn:us:gov:ic:cvenum:tdf:signaturealgorithm'"/>
                        <xsl:namespace name="tdfstate" select="'urn:us:gov:ic:cvenum:tdf:state'"/>
                        <xsl:namespace name="tdfhashal" select="'urn:us:gov:ic:cvenum:tdf:hashalgorithm'"/>
                        <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 - DHZMC-TDF-Changes.xml

```
<changes xmlns:xs="http://www.w3.org/2001/XMLSchema"
                        xmlns="urn:x-us:gov:ic:dhzm:changes"
                        xmlns:xhtml="http://www.w3.org/1999/xhtml-StopBrowserRendering"
                        xmlns:icsfhashv="urn:us:gov:ic:sf:hashverification"
                        xmlns:ch="urn:x-us:gov:ic:dhzm: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 DHZMC-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 transport of digital hazmat.  All
  changes are marked "DigitalHazMat:" with comments.

  DHZMC-TDF-Changes.xml
</documentation>

                <!--injectNamespaces
    ch:match="xs:schema[@targetNamespace='urn:us:gov:ic:tdf']"
    dhzm="urn:us:gov:ic:digitalhazmat">
    <!-/- DigitalHazMat: Add the DHZM namespace -/->
  </injectNamespaces-->

  <injectAttributes ch:match="xs:import[@namespace='urn:us:gov:ic:cvenum:tdf:signaturealgorithm']"
                                schemaLocation="../BASE-TDF/CVEGenerated/CVEnumTDFSignatureAlgorithm.xsd">
<injectAttributes ch:match="xs:import[@namespace='urn:us:gov:ic:cvenum:tdf:signaturealgorithm']"
                                schemaLocation="../BASE-TDF/CVEGenerated/CVEnumTDFSignatureAlgorithm.xsd">
  <!-- DigitalHazMat: Update import path to BASE-TDF CVEGenerated for CVEnumTDFHashAlgorithm -->
</injectAttributes>

                <injectAttributes ch:match="xs:import[@namespace='urn:us:gov:ic:cvenum:tdf:hashalgorithm']"
                                schemaLocation="../BASE-TDF/CVEGenerated/CVEnumTDFHashAlgorithm.xsd">
  <!-- DigitalHazMat: Update import path to BASE-TDF CVEGenerated for CVEnumTDFSignatureAlgorithm -->
</injectAttributes>

                <injectAttributes ch:match="xs:import[@namespace='urn:us:gov:ic:cvenum:tdf:state']"
                                schemaLocation="../BASE-TDF/CVEGenerated/CVEnumTDFAppliesToState.xsd">
  <!-- DigitalHazMat: Update import path to BASE-TDF CVEGenerated for CVEnumTDFAppliesToState -->
</injectAttributes>

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

                <injectBefore ch:match="xs:element[@name='TrustedDataCollection']">
  <!-- DigitalHazMat: Add DHZM schema imports -->
  <xs:import namespace="urn:us:gov:ic:digitalhazmat"
                            schemaLocation="../DHZM/DHZM.xsd"/>
                </injectBefore>

                <replaceElement ch:match="xs:element[@name='StringPayload']">
  <!-- DigitalHazMat: Remove references to StringPayload -->
```

```
</replaceElement>

            <replaceElement ch:match="xs:element[@name='StructuredPayload']">
  <!-- DigitalHazMat: Remove references to StructuredPayload -->
</replaceElement>

            <replaceElement ch:match="xs:complexType[@name='TdoType']">
  <!-- DigitalHazMat: Replace EncryptionInformationGroup with EncryptionInformationGroupRequired -->
  <xs:complexType name="TdoType">
                  <xs:sequence>
                     <xs:group maxOccurs="1" minOccurs="1" ref="AssertionGroup"/>
                     <xs:group maxOccurs="1"
                               minOccurs="1"
                               ref="EncryptionInformationGroupRequired"/>
                     <xs:group ref="PayloadGroup"/>
                  </xs:sequence>
                  <xs:attribute ref="version"/>
                  <xs:attribute ref="icsf:DESVersion" use="optional"/>
                  <xs:attribute ref="id" use="optional"/>
               </xs:complexType>
            </replaceElement>

            <replaceElement ch:match="xs:complexType[@name='StructuredValueType']/xs:sequence/xs:any">
  <!-- DigitalHazMat: Replace StructuredValueType xs:any with xs:choice of dhzm:ProvenanceAssertion or dhzm:AnalysisAssertion -->
  <xs:choice>
                  <xs:element xmlns:dhzm="urn:us:gov:ic:digitalhazmat" ref="dhzm:ProvenanceAssertion"/>
                  <xs:element xmlns:dhzm="urn:us:gov:ic:digitalhazmat" ref="dhzm:AnalysisAssertion"/>
               </xs:choice>
            </replaceElement>

            <replaceElement ch:match="xs:complexType[@name='Base64BinaryValueType']/xs:simpleContent/xs:extension[@base='xs:base64Binary']">
<!-- DigitalHazMat: Replace xs:base64Binary with Base64BinaryRestrictedType -->
<xs:extension base="Base64BinaryRestrictedType">
                  <xs:attribute ref="mediaType" use="optional"/>
                  <xs:attribute ref="filename" use="optional"/>
                  <xs:attribute ref="isEncrypted" use="optional"/>
               </xs:extension>
            </replaceElement>

            <injectAfter ch:match="xs:complexType[@name='Base64BinaryValueType']">
<!-- DigitalHazMat: Add Base64BinaryRestrictedType type -->
<xs:simpleType name="Base64BinaryRestrictedType">
                  <xs:restriction base="xs:base64Binary">
                     <xs:maxLength value="10000000"/>
                  </xs:restriction>
               </xs:simpleType>
            </injectAfter>

            <injectBefore ch:match="xs:group[@name='EncryptionInformationGroup']">
<!-- DigitalHazMat: Add EncryptionInformationGroupRequired group -->
<xs:group name="EncryptionInformationGroupRequired">
                  <xs:annotation>
                     <xs:documentation>
                        <xhtml:p ism:classification="U" ism:ownerProducer="USA">The group of elements used
```

```
        to express encryption information in an Assertion or a TDO.</xhtml:p>
                  </xs:documentation>
               </xs:annotation>
               <xs:sequence>
                  <xs:element maxOccurs="unbounded" minOccurs="1" name="EncryptionInformation">
                     <xs:annotation>
                        <xs:documentation>
                           <xhtml:p ism:classification="U" ism:ownerProducer="USA">Top level element
    for holding information related to the encryption of an assertion or
    payload. Multiple child KeyAccess and/or EncryptionMethod elements
    represent onion or layered encryption. In this case, the first child
    represents the outermost layer of encryption.</xhtml:p>
                        </xs:documentation>
                     </xs:annotation>
                     <xs:complexType>
                        <xs:choice maxOccurs="1">
                           <xs:sequence>
                              <xs:element minOccurs="1" name="KeyAccess" type="KeyAccessType">
                                 <xs:annotation>
                                    <xs:documentation>
                                       <xhtml:p ism:classification="U" ism:ownerProducer="USA">
            Contains information pertaining to the key for which the
            application value(s) was/were encrypted and/or that is
            necessary for decryption.</xhtml:p>
                                    </xs:documentation>
                                 </xs:annotation>
                              </xs:element>
                              <xs:element maxOccurs="1"
                                          minOccurs="1"
                                          name="EncryptionMethod"
                                          type="EncryptionMethodType">
                                 <xs:annotation>
                                    <xs:documentation>
                                       <xhtml:p ism:classification="U" ism:ownerProducer="USA">
            Contains information pertaining to the methods for which
            the applicable value(s) was/were encrypted. (i.e.
            SHA256)</xhtml:p>
                                    </xs:documentation>
                                 </xs:annotation>
                              </xs:element>
                           </xs:sequence>
                        </xs:choice>
                        <xs:attribute name="sequenceNum" type="xs:integer" use="optional"/>
                     </xs:complexType>
                  </xs:element>
               </xs:sequence>
            </xs:group>
         </xs:injectBefore>

      </xs:changes>
```

2.3 - DHZMC-TDF-Changes.xsl

```
<xsl:stylesheet xmlns:edh="urn:us:gov:ic:edh"
                                xmlns:arh="urn:us:gov:ic:arh"
                                xmlns:rr="urn:us:gov:ic:revrecall"
                                xmlns:dhzm="urn:us:gov:ic:digitalhazmat"
                                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                                xmlns="urn:x-us:gov:ic:dhzm:changes"
                                xmlns:xs="http://www.w3.org/2001/XMLSchema"
                                xmlns:xhtml="http://www.w3.org/1999/xhtml-StopBrowserRendering"
                                xmlns:icsfhashv="urn:us:gov:ic:sf:hashverification"
                                xmlns:ch="urn:x-us:gov:ic:dhzm: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 transport of digital hazmat.  All
  changes are marked "DigitalHazMat:" with comments.

  DHZMC-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:dhzm: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--><!--injectAttribute--><xsl:template match="xs:import[@namespace='urn:us:gov:ic:cvenum:tdf:signaturealgorithm']">
                  <xsl:call-template name="show-old-start-tag"/>
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> DigitalHazMat: Update import path to BASE-TDF CVEGenerated for CVEnumTDFHashAlgorithm </xsl:comment>
</xsl:with-param>
                  </xsl:call-template>
                  <xsl:copy>
                     <xsl:apply-templates select="@*"/>
                     <xsl:attribute name="schemaLocation" namespace="">../BASE-TDF/CVEGenerated/CVEnumTDFSignatureAlgorithm.xsd</xsl:attribute>
                     <xsl:apply-templates select="node()"/>
                  </xsl:copy>
```

```
            </xsl:template>
            <!--injectAttribute--><xsl:template match="xs:import[@namespace='urn:us:gov:ic:cvenum:tdf:hashalgorithm']">
               <xsl:call-template name="show-old-start-tag"/>
               <xsl:call-template name="copy-content">
                  <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> DigitalHazMat: Update import path to BASE-TDF CVEGenerated for CVEnumTDFSignatureAlgorithm </xsl:comment>
</xsl:with-param>
               </xsl:call-template>
               <xsl:copy>
                  <xsl:apply-templates select="@*"/>
                  <xsl:attribute name="schemaLocation" namespace="">../BASE-TDF/CVEGenerated/CVEnumTDFHashAlgorithm.xsd</xsl:attribute>
                  <xsl:apply-templates select="node()"/>
               </xsl:copy>
            </xsl:template>
            <!--injectAttribute--><xsl:template match="xs:import[@namespace='urn:us:gov:ic:cvenum:tdf:state']">
               <xsl:call-template name="show-old-start-tag"/>
               <xsl:call-template name="copy-content">
                  <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> DigitalHazMat: Update import path to BASE-TDF CVEGenerated for CVEnumTDFAppliesToState </xsl:comment>
</xsl:with-param>
               </xsl:call-template>
               <xsl:copy>
                  <xsl:apply-templates select="@*"/>
                  <xsl:attribute name="schemaLocation" namespace="">../BASE-TDF/CVEGenerated/CVEnumTDFAppliesToState.xsd</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> DigitalHazMat: Update tdf:version regex to be for DHZMC-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})?\-DHZMC\-TDF\.[0-9]{6}(\.[0-9]{6})?(\-.{1,23})?</xsl:attribute>
                  <xsl:apply-templates select="node()"/>
               </xsl:copy>
            </xsl:template>
            <!--injectBefore--><xsl:template match="xs:element[@name='TrustedDataCollection']">
               <xsl:call-template name="copy-content">
                  <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> DigitalHazMat: Add DHZM schema imports </xsl:comment>
  <xs:import namespace="urn:us:gov:ic:digitalhazmat" schemaLocation="../DHZM/DHZM.xsd"/>
</xsl:with-param>
               </xsl:call-template>
               <xsl:copy>
                  <xsl:apply-templates select="@*|node()"/>
               </xsl:copy>
            </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> DigitalHazMat: Remove references to StringPayload </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> DigitalHazMat: Remove references to StructuredPayload </xsl:comment>
</xsl:with-param>
                  </xsl:call-template>
               </xsl:template>
               <!--replaceElement--><xsl:template match="xs:complexType[@name='TdoType']">
                  <xsl:call-template name="show-old-element"/>
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> DigitalHazMat: Replace EncryptionInformationGroup with EncryptionInformationGroupRequired </xsl:comment>
  <xs:complexType name="TdoType">
    <xs:sequence>
      <xs:group maxOccurs="1" minOccurs="1" ref="AssertionGroup"/>
      <xs:group maxOccurs="1" minOccurs="1" ref="EncryptionInformationGroupRequired"/>
      <xs:group ref="PayloadGroup"/>
    </xs:sequence>
    <xs:attribute ref="version"/>
    <xs:attribute ref="icsf:DESVersion" use="optional"/>
    <xs:attribute ref="id" use="optional"/>
  </xs:complexType>
</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> DigitalHazMat: Replace StructuredValueType xs:any with xs:choice of dhzm:ProvenanceAssertion or dhzm:AnalysisAssertion </xsl:comment>
  <xs:choice>
    <xs:element ref="dhzm:ProvenanceAssertion"/>
    <xs:element ref="dhzm:AnalysisAssertion"/>
  </xs:choice>
</xsl:with-param>
                  </xsl:call-template>
               </xsl:template>
               <!--replaceElement--><xsl:template match="xs:complexType[@name='Base64BinaryValueType']/xs:simpleContent/xs:extension[@base='xs:base64Binary']">
                  <xsl:call-template name="show-old-element"/>
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> DigitalHazMat: Replace xs:base64Binary with Base64BinaryRestrictedType </xsl:comment>
  <xs:extension base="Base64BinaryRestrictedType">
    <xs:attribute ref="mediaType" use="optional"/>
    <xs:attribute ref="filename" use="optional"/>
    <xs:attribute ref="isEncrypted" use="optional"/>
  </xs:extension>
</xsl:with-param>
```

```
                  </xsl:call-template>
               </xsl:template>
               <!--injectAfter--><xsl:template match="xs:complexType[@name='Base64BinaryValueType']">
                  <xsl:copy>
                     <xsl:apply-templates select="@*|node()"/>
                  </xsl:copy>
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> DigitalHazMat: Add Base64BinaryRestrictedType type </xsl:comment>
  <xs:simpleType name="Base64BinaryRestrictedType">
    <xs:restriction base="xs:base64Binary">
      <xs:maxLength value="10000000"/>
    </xs:restriction>
  </xs:simpleType>
</xsl:with-param>
                  </xsl:call-template>
               </xsl:template>
               <!--injectBefore--><xsl:template match="xs:group[@name='EncryptionInformationGroup']">
                  <xsl:call-template name="copy-content">
                     <xsl:with-param name="rtf" xml:space="preserve">
  <xsl:comment> DigitalHazMat: Add EncryptionInformationGroupRequired group </xsl:comment>
  <xs:group name="EncryptionInformationGroupRequired">
    <xs:annotation>
      <xs:documentation>
        <xhtml:p ism:classification="U" ism:ownerProducer="USA">The group of elements used
          to express encryption information in an Assertion or a TDO.</xhtml:p>
      </xs:documentation>
    </xs:annotation>
  </xs:documentation>
</xs:annotation>
<xs:sequence>
  <xs:element maxOccurs="unbounded" minOccurs="1" name="EncryptionInformation">
    <xs:annotation>
      <xs:documentation>
        <xhtml:p ism:classification="U" ism:ownerProducer="USA">Top level element
          for holding information related to the encryption of an assertion or
          payload. Multiple child KeyAccess and/or EncryptionMethod elements
          represent onion or layered encryption. In this case, the first child
          represents the outermost layer of encryption.</xhtml:p>
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:choice maxOccurs="1">
        <xs:sequence>
          <xs:element minOccurs="1" name="KeyAccess" type="KeyAccessType">
            <xs:annotation>
              <xs:documentation>
                <xhtml:p ism:classification="U" ism:ownerProducer="USA">
                  Contains information pertaining to the key for which the
                  application value(s) was/were encrypted and/or that is
                  necessary for decryption.</xhtml:p>
              </xs:documentation>
            </xs:annotation>
          </xs:element>
          <xs:element maxOccurs="1" minOccurs="1" name="EncryptionMethod" type="EncryptionMethodType">
            <xs:annotation>
```

```
                <xs:documentation>
                  <xhtml:p ism:classification="U" ism:ownerProducer="USA">
                    Contains information pertaining to the methods for which
                    the applicable value(s) was/were encrypted. (i.e.
                    SHA256)</xhtml:p>
                </xs:documentation>
              </xs:annotation>
            </xs:element>
          </xs:sequence>
        </xs:choice>
        <xs:attribute name="sequenceNum" type="xs:integer" use="optional"/>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
 </xs:group>
</xsl:with-param>
                </xsl:call-template>
                <xsl:copy>
                   <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:dhzm: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:edh="urn:us:gov:ic:edh"
                                    xmlns:arh="urn:us:gov:ic:arh"
                                    xmlns:rr="urn:us:gov:ic:revrecall"
                                    xmlns:dhzm="urn:us:gov:ic:digitalhazmat">
    <!--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:dhzm: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:dhzm: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>
</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:dhzm: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 - StripComments.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="xs xd"
                              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>Strip comments and annotations out of xsd file 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="*"/>

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

               <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>
```

2.8 - 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 DHZMC-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 DigitalHazMat Commercial TDF
               (DHZMC-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 DigitalHazMat Commercial TDF
               (DHZMC-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/DHZMC-TDF/DesDhzmcTdfXml.pdf">XML Data Encoding Specification for DigitalHazMat Commercial 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 DHZMC-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 DHZMC-TDF is:
               <xhtml:a href="DHZMC-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>
```