```
<?xml version='1.0'?>
<!-- /*********************************************************************/ -->
<!-- /* (C) Copyright IBM Corp.   2004, 2004   All Rights Reserved.       */ -->
<!-- /* DISCLAIMER OF WARRANTIES.                                         */ -->
<!-- /*                                                                   */ -->
<!-- /* The following code is presented to you solely for the purpose of  */ -->
<!-- /* enhancing your understanding of the IMS MFS Web Enablement        */ -->
<!-- /* Technology Preview.                                               */ -->
<!-- /* The code is presented "AS IS." IBM MAKES NO WARRANTIES, EXPRESS OR*/ -->
<!-- /* IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF   */ -->
<!-- /* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, REGARDING   */ -->
<!-- /* THE FUNCTION OR PERFORMANCE OF THIS CODE.                         */ -->
<!-- /* IBM shall not be liable for any damages arising out of your use   */ -->
<!-- /* of the provided code, even if it has been advised of the          */ -->
<!-- /* possibility of such damages.                                      */ -->
<!-- /*                                                                   */ -->
<!-- /* The code and materials accessed on this demo site may not be      */ -->
<!-- /* distributed, copied, altered, or incorporated into other software.*/ -->
<!-- /*********************************************************************/ -->
<!--                                                                       -->
<!--  This style sheet is customized for demo purposes to generate enhanced  -->
<!--  interface HTML pages for MFS-based IMS applications.                 -->
<!--  It is targeted for browsers supporting cascading stylesheet and      -->
<!--  style attributes, i.e. IE 6+ and Mozilla 1.7.  Attempting to display -->
<!--  the html output rendered with this style sheet on browsers not       -->
<!--  supporting cascading style sheet and style attributes will result in -->
<!--  misplaced text and input fields.                                     -->

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:axslt="http://xml.apache.org/xslt"
                xmlns:xmi="http://www.omg.org/XMI"
                xmlns:MFS="MFS.xmi"
                version="1.0">
  <!-- Users can modify color, font, position, and input field definitions -->
  <!-- color definitions -->
  <xsl:variable name="blue">blue</xsl:variable>
  <xsl:variable name="red">red</xsl:variable>
  <xsl:variable name="green">rgb(33,70,40)</xsl:variable>
  <xsl:variable name="pink">rgb(160,50,140)</xsl:variable>
  <xsl:variable name="turquoise">rgb(52,126,124)</xsl:variable>
  <xsl:variable name="yellow">rgb(244,122,0)</xsl:variable>
  <xsl:variable name="default">rgb(100,50,0)</xsl:variable>
  <xsl:variable name="neutral">rgb(111,111,111)</xsl:variable>
  <xsl:variable name="input">rgb(60,60,60)</xsl:variable>
  <xsl:variable name="background">white</xsl:variable>
  <xsl:variable name="inputBackground">rgb(192,192,192)</xsl:variable>

  <!-- font definitions -->
  <xsl:variable name="font-family">'Courier New'</xsl:variable>
  <xsl:variable name="font-size">12pt</xsl:variable>
  <xsl:variable name="font-weight">bold</xsl:variable>

  <!-- position definitions -->
  <xsl:variable name="row-multiplier">21</xsl:variable>
  <xsl:variable name="column-multiplier">10</xsl:variable>

  <!-- input field definitions -->
```

```
  <xsl:variable name="border">.5in</xsl:variable>

  <!-- cursor definitions -->
  <xsl:variable name="cursorRow">0</xsl:variable>
  <xsl:variable name="cursorColumn">0</xsl:variable>

  <!-- start of html doc -->
  <xsl:output method="html"/>

  <!-- beginning of xmi document -->
  <xsl:template match="/">
    <html>
      <head>
        <!-- Style declarations -->
        <!-- Users can modify this section for different font colors, background
colors, -->
        <!-- font-family, font-size, font-weight, or button styles
-->
        <style type="text/css" media="screen">
            <!-- Default style for body, table, and input elements -->
          <xsl:text>body, table</xsl:text>
          <xsl:text>  { color: black</xsl:text>
          <xsl:text>; background: </xsl:text><xsl:value-of
select="$background"/>
          <xsl:text>; font-family: </xsl:text><xsl:value-of select="$font-
family"/>
          <xsl:text>; font-size: </xsl:text><xsl:value-of select="$font-size"/>
          <xsl:text>; font-weight: </xsl:text><xsl:value-of select="$font-
weight"/>
          <xsl:text> } </xsl:text>
          <xsl:text>input</xsl:text>
          <xsl:text>  { color: black</xsl:text>
          <xsl:text>; background: </xsl:text><xsl:value-of
select="$inputBackground"/>
          <xsl:text>; font-family: </xsl:text><xsl:value-of select="$font-
family"/>
          <xsl:text>; font-size: </xsl:text><xsl:value-of select="$font-size"/>
          <xsl:text>; font-weight: </xsl:text><xsl:value-of select="$font-
weight"/>
          <xsl:text> } </xsl:text>
          <xsl:text>textarea</xsl:text>
          <xsl:text>  { color: black</xsl:text>
          <xsl:text>; background: </xsl:text><xsl:value-of
select="$inputBackground"/>
          <xsl:text>; font-family: </xsl:text><xsl:value-of select="$font-
family"/>
          <xsl:text>; font-size: </xsl:text><xsl:value-of select="$font-size"/>
          <xsl:text>; font-weight: </xsl:text><xsl:value-of select="$font-
weight"/>
          <xsl:text> } </xsl:text>
          <xsl:text>.buttonStyle</xsl:text>
          <xsl:text>  { color: </xsl:text><xsl:value-of
select="$inputBackground"/>
          <xsl:text>; font-family:Helvetica</xsl:text><!-- xsl:value-of
select="$font-family"/ -->
          <xsl:text>; font-size: 11</xsl:text><!-- xsl:value-of select="$font-
size"/ -->
```

```
          <xsl:text>; font-weight:</xsl:text><xsl:value-of select="$font-
weight"/>
          <xsl:text>; width: 45px </xsl:text>
          <xsl:text>; height:20px </xsl:text>
          <xsl:text>; background:#006028</xsl:text>
          <xsl:text>; border-color:#B0D868</xsl:text>
          <xsl:text> } </xsl:text>
          <xsl:text>.imageStyle</xsl:text>
          <xsl:text>  { color:white </xsl:text>
          <xsl:text>; background:white</xsl:text>
          <xsl:text>; width: 86px</xsl:text>
          <xsl:text>; height: 26px</xsl:text>
          <xsl:text>; border:none</xsl:text>
          <xsl:text> } </xsl:text>

        </style>
        <!-- Do NOT modify existing JavaScript functions -->
        <SCRIPT Language="JavaScript">
          <![CDATA[
            function setFocus(field) {
                var layersSupport = (document.layers!=null);
                var fField = -1;
                if (!layersSupport) {
                    // for IE 5 and NS 6
                    for (var j = 0; j < document.forms.length; j++) {
                        var form = document.forms[j];
                        for (var i=0; i < form.length; i++) {
                            if (fField == -1) {
                                //if field is empty, focus on the first input
element, else focus on the specified input field
                                if (((field.length < 1) &&
(form.elements[i].type == "text")) || (form.elements[i].name == field)) {
                                    fField = i;
                                    form.elements[i].focus();
                                    break;
                                }
                            }
                        }
                    }
                }
            }

            function goToURL()
            {
               myWindow = open("","newwin","height=600,width=920,resizable=1");
               myWindow.location = "http://www-306.ibm.com/software/data/ims/";
            }

            function submitEnter(myfield,e)
            {
               var keycode;
               if (window.event) keycode = window.event.keyCode;
               else if (e) keycode = e.which;
               else return true;

              // submits the form when presses 'Enter'
              if (keycode == 13){
```

```
              for (var i=0; i < document.forms.length; i++){
                if (document.forms[i].name=="Process") {
                      //processSubmit(document.forms[i]);
                          for (var j=0; j < document.forms[i].length; j++){
                            if (document.forms[i].elements[j].name=="submitButton")
{
                               document.forms[i].elements[j].click();
                                        break;
                                    }
                            }
                }
                }
                return false;
              }
              else
                return true;
            }

            function clearForm()
            {
                var layersSupport = (document.layers!=null);
                var fLayer = -1;
                var fField = -1;
                var fCnt = -1;
                if (!layersSupport) {
                    // for IE and NS 6
                    for (var j = 0; j < document.forms.length; j++) {
                        var form = document.forms[j];
                        for (var i=0; i < form.length; i++) {
                            //clear out text fields
                            if (form.elements[i].type == "text" ||
form.elements[i].type == "textarea") {
                                if (fField == -1) {
                                        fField = i;
                                        fCnt = j;
                                }
                                    form.elements[i].value = "";
                            }
                        }
                    }
                    if (fField >= 0 && fCnt >= 0) {
                        //put focus on the first text field
                        document.forms[fCnt].elements[fField].focus();
                    }
                }
            }

            // Reject numeric inputs
            function numeric(e)
            {
                if (navigator.appName == "Microsoft Internet Explorer")
                    Key = window.event.keyCode;
                else if (navigator.appName == "Netscape")
                    Key = e.which;
                else if (navigator.appName == "Mozilla")
                    Key = e.keyCode;
                if (Key < 48 || Key > 57) {
```

```
                    alert("Please enter only numeric value");
                    return false;
                }
            }


            // Process the submit form
            function processSubmit(frm)
            {
               var elementsLength = frm.length
               for (var i=0; i < elementsLength; i++)
               // clear hidden fields and copy over user data value
               // skip button values
                  if ((frm[i].type=="hidden") &&
                     !(frm[i].name=="resetButton" || frm[i].name==
"logoutButton" || frm[i].name== "submitButton" ||
                        frm[i].name=="PA1Button")){

                    frm[i].value = ""
                    findForms(frm, window.document)
                  }
            }

            // Find fields
            function findForms(fSubmit, doc)
            {
               var formsCnt = doc.forms.length, fSource
               var elementsLength, eSource
               // Enumerate forms and find forms in same group
               for (var i=0; i < formsCnt; i++) {
                  fSource = doc.forms[i]
                     //if ((fSubmit.name==fSource.name) && (fSubmit!=fSource))
                     if ((fSource.name=="Info") && (fSubmit!=fSource))
                     {
                    elementsLength = fSource.length
                    // Copy fields
                    for (var j=0; j < elementsLength; j++) {
                       eSource = fSource[j]
                       // Make sure field exists in submit form
                       if (fSubmit[eSource.name]!=null && eSource.type!="hidden")
                          fSubmit[eSource.name].value = eSource.value
                       }
                    }
                }
                // In NS 4 recurse throughs sub-documents
                var layersSupport = (document.layers!=null) //NS4
                if (layersSupport) {
                  var layersLength = d.layers.length
                  for (var l=0; l < layersLength; l++)
                    findForms(fSubmit, d.layers[l].document)
                  }
            }

            function submitImageButton(buttonname)
            {

               var layersSupport = (document.layers!=null);
```

```
                if (!layersSupport) {
                // for IE and NS 6
                   for (var j = 0; j < document.forms.length; j++) {
                      var form = document.forms[j];
                      for (var i=0; i < form.length; i++) {
                      //clear out text fields
                         if (form.elements[i].name == "resetButton" &&
                             form.elements[i].type == "hidden" &&
                             buttonname=="resetButton") {
                             form.elements[i].value="Reset";
                         }
                         else if (form.elements[i].name == "PA1Button" &&
                             form.elements[i].type == "hidden" &&
                                 buttonname=="PA1Button") {
                             form.elements[i].value="Next Page";
                         }
                         else if (form.elements[i].name == "logoutButton" &&
                             form.elements[i].type == "hidden" &&
                                 buttonname=="logoutButton") {
                             form.elements[i].value="Logout";
                         }

                      }
                   }
                }
                else {
                  this.onload();
                }
             }

             function disableBackButton()
             {
                history.forward();
             }
      ]]>
    </SCRIPT>
    </head>
    <!-- Invoke device template on output MFSMessage -->
    <xsl:element name="body">
    <xsl:attribute name="bgcolor"><xsl:value-of
select="$background"/></xsl:attribute>
    <!-- Check the formatType of MFSFormat before processing -->
    <xsl:variable name="formatType"><xsl:value-of
select="//MFS:MFSFormat/devices[1]/divisions/@type"/></xsl:variable>
    <xsl:if test="$formatType='out' or $formatType='inout'">
        <xsl:apply-templates select="//MFS:MFSFormat/devices[1]"/>
    </xsl:if>
    </xsl:element>
    </html>
</xsl:template>
<!-- device template: generates HTML body contents -->
<xsl:template match="devices">
   <!-- Invoke cursor template for cursor position -->
   <xsl:apply-templates select="divisions/devicePages/physicalPages/cursor"/>
   <!-- Load images for the banner -->
   <table style="position: absolute; top: 1px; left: 10"><img src =
"MFSXML_banner.gif" height="40"/></table>
```

```
   <!-- Invoke deviceFields template for generating labels and input fields -->
   <xsl:apply-templates
select="divisions/devicePages/physicalPages/deviceFields"/>
   <!-- Create Function Key buttons in a form-->
   <xsl:element name="table">
       <xsl:attribute name="style">position: absolute; top: 60px; left:
890px</xsl:attribute>
       <xsl:element name="form">
       <!-- Invoke processSubmit() to fill in device field data -->
       <xsl:attribute name="Name">PFKeys</xsl:attribute>
       <xsl:attribute name="ONSUBMIT">processSubmit(this);
submit</xsl:attribute>
       <xsl:attribute name="Action"></xsl:attribute>
       <xsl:attribute name="method">get</xsl:attribute>
       <!-- Create hidden field for each input data device field -->
         <xsl:for-each
select="//MFS:MFSFormat/devices[1]/divisions/devicePages/physicalPages/deviceFie
lds">
         <xsl:if test="not(@password='true') and
not(attributes/@protected='true')">
           <xsl:element name="input">
               <xsl:attribute name="type">hidden</xsl:attribute>
               <xsl:attribute name="name"><xsl:value-of
select="@label"/></xsl:attribute>
           </xsl:element>
         </xsl:if>
       </xsl:for-each>
          <!-- Function Key Buttons -->
          <xsl:if test="functionKeyList">
          <xsl:for-each select="functionKeyList/functionKeys">
        <!-- display button if there is a functionKey in a functionKeyList-->
             <xsl:variable name="number"><xsl:number/></xsl:variable>
             <xsl:if test="$number &lt; 18 or $number = '18'">
               <xsl:element name="tr">
               <xsl:element name="td">
               <xsl:element name="input">
               <xsl:attribute name="type">submit</xsl:attribute>
               <xsl:attribute name="value">PF<xsl:number/></xsl:attribute>
               <xsl:attribute name="class">buttonStyle</xsl:attribute>
               <xsl:attribute name="name">PF<xsl:number/></xsl:attribute>
               </xsl:element>
               </xsl:element>
               </xsl:element>
             </xsl:if>
             </xsl:for-each>
          </xsl:if>
     </xsl:element>
     </xsl:element>

    <xsl:element name="table">
       <xsl:attribute name="style">position: absolute; top: 60px; left:
942px</xsl:attribute>
       <xsl:element name="form">
       <!-- Invoke processSubmit() to fill in device field data -->
          <xsl:attribute name="Name">PFKeys</xsl:attribute>
          <xsl:attribute name="ONSUBMIT">processSubmit(this);
submit</xsl:attribute>
```

```
          <xsl:attribute name="Action"></xsl:attribute>
          <xsl:attribute name="method">get</xsl:attribute>
          <!-- Create hidden field for each input data device field -->
          <xsl:for-each
select="//MFS:MFSFormat/devices[1]/divisions/devicePages/physicalPages/deviceFie
lds">
          <xsl:if test="not(@password='true') and
not(attributes/@protected='true')">
             <xsl:element name="input">
             <xsl:attribute name="type">hidden</xsl:attribute>
             <xsl:attribute name="name"><xsl:value-of
select="@label"/></xsl:attribute>
             </xsl:element>
          </xsl:if>
          </xsl:for-each>
          <!-- Function Key Buttons -->
          <xsl:if test="functionKeyList">
               <!-- display button if there is a functionKey in a
functionKeyList -->
               <xsl:for-each select="functionKeyList/functionKeys">
                 <xsl:variable name="number"><xsl:number/></xsl:variable>
                 <xsl:if test="$number &gt; 18">
                   <xsl:element name="tr">
                   <xsl:element name="td">
                   <xsl:element name="input">
                   <xsl:attribute name="type">submit</xsl:attribute>
                   <xsl:attribute name="value">PF<xsl:number/></xsl:attribute>
                   <xsl:attribute name="class">buttonStyle</xsl:attribute>
                   <xsl:attribute name="name">PF<xsl:number/></xsl:attribute>
                   </xsl:element>
                   </xsl:element>
                   </xsl:element>
                 </xsl:if>
               </xsl:for-each>
          </xsl:if>
   </xsl:element>
   </xsl:element>

   <!-- Create Submit, Clear, Reset, Next, and Previous buttons in a form -->
   <xsl:element name="table">
       <xsl:attribute name="style">position: absolute; top: 43px; left:
10px</xsl:attribute>
       <xsl:element name="form">
   <!-- Invoke processSubmit() to fill in device field data -->
       <xsl:attribute name="Name">Process</xsl:attribute>
       <xsl:attribute name="ONSUBMIT">processSubmit(this);
submit</xsl:attribute>
       <xsl:attribute name="Action"></xsl:attribute>
       <xsl:attribute name="method">get</xsl:attribute>
       <xsl:attribute name="onKeyPress">return
submitEnter(this,event)</xsl:attribute>
       <!-- Create hidden field for each input data device field -->
         <xsl:for-each
select="//MFS:MFSFormat/devices[1]/divisions/devicePages/physicalPages/deviceFie
lds">
         <xsl:if test="not(@password='true') and
not(attributes/@protected='true')">
```

```
           <xsl:element name="input">
               <xsl:attribute name="type">hidden</xsl:attribute>
               <xsl:attribute name="name"><xsl:value-of
select="@label"/></xsl:attribute>
           </xsl:element>
       </xsl:if>
       </xsl:for-each>
       <xsl:element name="tr">
       <xsl:element name = "td">
       <!-- Submit button -->
       <button class="imageStyle" type="submit" name="submitButton">
       <img src="submit.gif" alt="submit inputs to host" title="submit inputs to
host"/>
       </button>
       </xsl:element>
       <xsl:element name="td">
       <!-- Clear button - Invoke clearForm() to clear data on a form -->
       <button class="imageStyle" type="button" onclick="clearForm()"
name="clearButton">
       <img src="clear.gif" alt="clear contents of all input fields"
title="clear contents of all input fields"/>
       </button>
       </xsl:element>
       <xsl:element name="td">
       <!-- Next button -->
       <input type="image" name="PA1Button" src="next.gif" alt="go to next
physical page" title="go to next physical page"
onclick="submitImageButton('PA1Button')"/>
       <input name="PA1Button" type="hidden" value=""/>
       </xsl:element>
       <xsl:element name="td">
       <!-- Unformat button -->
       <input type="image" name="resetButton" src="reset.gif" alt="back to the
first session where a command or a transaction can be entered" title="back to
the first session where a command or a transaction can be entered"
onclick="submitImageButton('resetButton')"/>
       <input name="resetButton" type="hidden"/>
         </xsl:element>
       <xsl:element name="td">
       <!-- Logout button -->
       <input type="image" name="logoutButton" src="logout.gif" alt="close all
connections and exit" title="close all connections and exit"
onclick="submitImageButton('logoutButton')"/>
         <input name="logoutButton" type="hidden" value=""/>
       </xsl:element>
       <xsl:element name="td">
       <!-- Help button - Invoke goToURL() to go to the MFS Web Enablement
User's Guide -->
       <button class="imageStyle" type="button" onclick="goToURL()"
name="helpButton">
       <img src="help.gif" alt="display MFS Web Enablement User's Guide"
title="display MFS Web Enablement User's Guide"/>
       </button>
       </xsl:element>
       </xsl:element> <!-- end element "tr" -->
       </xsl:element> <!-- end element "form" -->
    </xsl:element>  <!-- end element "table" -->
```

```
</xsl:template>  <!-- end template "devices" -->

<!-- cursor template: find the specified field -->
<xsl:template match="cursor">
    <xsl:attribute name="onload">
    <xsl:text>setFocus('</xsl:text>

    <!-- Based on the row and column specified, find the name of the matching
field -->
    <!-- If not found, focus will be set on the first input field -->
    <xsl:variable name="cursorRow"><xsl:value-of
select="position/@row"/></xsl:variable>
    <xsl:variable name="cursorColumn"><xsl:value-of
select="position/@column"/></xsl:variable>
    <xsl:for-each select="../deviceFields">
    <xsl:if test="position/@row = $cursorRow and position/@column =
$cursorColumn ">
        <xsl:value-of select="@label" />
    </xsl:if>
    </xsl:for-each>
    <xsl:text>')</xsl:text>
    <xsl:text>;</xsl:text>
    <xsl:text>disableBackButton()</xsl:text>
    </xsl:attribute>  <!-- end of attribute "onload" -->
</xsl:template>  <!-- end of template "cursor" -->

<!-- deviceFields template: generates labels and input fields -->
<!-- Logic is as follows: enclose each label and input field in a table to
ensure absolute positioning -->
<!--                      and color assignments.  Reverse highlighting will
result in an hidden label  -->
<!--                      with the same foreground and background color.
-->

<xsl:template match="deviceFields">
<!-- If password, don't create the input field -->
<xsl:choose>
    <xsl:when test="@password='true'">
    <!-- No op -->
    </xsl:when>
<xsl:otherwise>
    <xsl:element name="table">
    <xsl:attribute name="style">
    <!-- Specify absolute positioning -->
    <xsl:if test="position">
        <xsl:text>position: absolute</xsl:text>
        <xsl:text>; top: </xsl:text><xsl:value-of select="position/@row * $row-
multiplier + 50"/><xsl:text>px</xsl:text>
        <xsl:text>; left: </xsl:text><xsl:value-of select="position/@column *
$column-multiplier"/><xsl:text>px</xsl:text>
    </xsl:if>
    <!-- Specify color -->
    <xsl:if test="attributes/@protected='true'">
        <xsl:text>; color: </xsl:text>
          <!-- Assign default color -->
          <xsl:if test="not(attributes) and not(extendedAttributes)">
            <xsl:value-of select="$default"/>
```

```
        </xsl:if>
        <!-- attributes template: assigns color -->
        <xsl:apply-templates select="attributes"/>
        <!-- extendedAttributes template: assigns color and highlighting
features -->
        <xsl:apply-templates select="extendedAttributes"/>
    </xsl:if>
    </xsl:attribute> <!-- end of attribute "style" -->
    <xsl:element name="tr">
    <xsl:element name="td">
    <xsl:if test="@label='OUTL'">
        <xsl:text>Please enter a command or a transaction:</xsl:text>
    </xsl:if>
    </xsl:element>
    </xsl:element>

    <xsl:element name="tr">
    <xsl:element name="td">
    <xsl:choose>
        <!-- Create input text fields -->
        <!--   when no literal value specified and attribute not protected -->
        <xsl:when test="not(attributes/@protected='true')">
            <!--  Embedd each input field in a non-active form -->
            <xsl:element name="form">
            <xsl:attribute name="Name">Info</xsl:attribute>
          <xsl:attribute name="ONSUBMIT">return false</xsl:attribute>
            <xsl:attribute name="onKeyPress">return
submitEnter(this,event)</xsl:attribute>

            <!--  Specify type, value, class, size, maxlength, and readonly
attributes -->
            <!--  if length is greater than 80 generate textarea field otherwise
generate input field -->
            <xsl:variable name="cols">80</xsl:variable>
            <xsl:variable name="rows">
            <xsl:choose>
                <xsl:when test="@length &gt; 80"><xsl:value-of
select="round(@length div 80)"/></xsl:when>
            </xsl:choose>
            </xsl:variable>
            <xsl:choose>
                <xsl:when test="@length &gt; 80">
                <xsl:element name="textarea">
                <xsl:attribute name="rows"><xsl:value-of
select="$rows"/></xsl:attribute>
                <xsl:attribute name="cols"><xsl:value-of
select="$cols"/></xsl:attribute>
                <xsl:attribute name="name"><xsl:value-of
select="@label"/></xsl:attribute>
                  <xsl:choose>
                      <xsl:when test="attributes/@intensity='nondisplayable'">
                      <xsl:attribute name="type">hidden</xsl:attribute>
                      </xsl:when>
                  <xsl:otherwise>
                      <xsl:attribute name="style">
                      <xsl:text>color: </xsl:text>
                      <xsl:if test="attributes">
```

```
                        <!-- attributes template: assigns color -->
                        <xsl:apply-templates select="attributes"/>
                    </xsl:if>
                    <xsl:if test="extendedAttributes">
                    <!-- extendedAttributes template: assigns color and
highlighting features -->
                        <xsl:apply-templates select="extendedAttributes"/>
                    </xsl:if>
                    <xsl:text>; border: </xsl:text><xsl:value-of
select="$border"/>
                    </xsl:attribute> <!-- end of attribute "style" -->
                    <xsl:attribute name="size"><xsl:value-of
select="@length"/></xsl:attribute>
                    <xsl:attribute name="maxlength"><xsl:value-of
select="@length"/></xsl:attribute>

                    <!-- Check for numeric only fields -->
                    <xsl:if test = "attributes/@numeric='true'">
                    <xsl:attribute name="onKeyPress"><xsl:text>return
numeric(event)</xsl:text></xsl:attribute>
                    </xsl:if>
                  </xsl:otherwise>
                  </xsl:choose>
                </xsl:element>   <!-- end of element "textarea" -->
            </xsl:when>

            <xsl:otherwise>
                <xsl:element name="input">
                <xsl:attribute name="name"><xsl:value-of
select="@label"/></xsl:attribute>
                <xsl:choose>
                    <xsl:when test="attributes/@intensity='nondisplayable'">
                        <xsl:attribute name="type">hidden</xsl:attribute>
                    </xsl:when>
                <xsl:otherwise>
                    <xsl:attribute name="type">text</xsl:attribute>
                    <xsl:attribute name="value"><xsl:value-of
select="@value"/></xsl:attribute>
                    <xsl:attribute name="style">
                    <xsl:text>color: </xsl:text>
                    <xsl:if test="attributes">
                        <!-- attributes template: assigns color -->
                        <xsl:apply-templates select="attributes"/>
                    </xsl:if>
                    <xsl:if test="extendedAttributes">
                        <!-- extendedAttributes template: assigns color and
highlighting features -->
                        <xsl:apply-templates select="extendedAttributes"/>
                    </xsl:if>
                    <xsl:text>; border: </xsl:text><xsl:value-of
select="$border"/>
                    </xsl:attribute> <!-- end of attribute "style"-->
                    <xsl:attribute name="size"><xsl:value-of
select="@length"/></xsl:attribute>
                    <xsl:attribute name="maxlength"><xsl:value-of
select="@length"/></xsl:attribute>
                    <!-- Check for numeric only fields -->
```

```
                    <xsl:if test = "attributes/@numeric='true'">
                        <xsl:attribute name="onKeyPress"><xsl:text>return
numeric(event)</xsl:text></xsl:attribute>
                    </xsl:if>

                </xsl:otherwise>
                </xsl:choose>
                </xsl:element>  <!-- end of element "input" -->
                 </xsl:otherwise>
                </xsl:choose>
                </xsl:element>
            </xsl:when>

            <!-- Create text label -->
    <xsl:otherwise>
        <!--  Font tag will overrule color assignment -->
        <xsl:element name="font">
        <xsl:attribute name="color">
        <!-- Assign default color -->
        <xsl:if test="not(attributes) and not(extendedAttributes)">
            <xsl:value-of select="$default"/>
        </xsl:if>
        <!-- attributes template: assigns color -->
        <xsl:apply-templates select="attributes"/>
        <!-- extendedAttributes template: assigns color and highlighting
features -->
        <xsl:apply-templates select="extendedAttributes">
            <xsl:with-param name="tag">font</xsl:with-param>
        </xsl:apply-templates>
        </xsl:attribute> <!-- end of attribute "color"-->
        <!-- Specify value of the label, explicitly convert space to displayable
space character -->
        <xsl:choose>
            <xsl:when test="@length &lt;= 80">
                <xsl:value-of select="translate(@value,' ',' ')"/>
            </xsl:when>
        <xsl:otherwise>
            <xsl:value-of select="translate(@value,'','')"/>
        </xsl:otherwise>
        </xsl:choose>
        </xsl:element> <!-- end of element "font" -->
    </xsl:otherwise>
    </xsl:choose>
    </xsl:element>
    </xsl:element>
    </xsl:element>
  </xsl:otherwise>
  </xsl:choose>
  </xsl:template>  <!-- end of template "device fields" -->

  <!-- attribute template: specify color -->
  <xsl:template match="attributes">
  <!-- Specify color for label text fields and input text fields based on
intensity and protected attributes -->
  <xsl:choose>
      <xsl:when test="@intensity='nondisplayable'">
      <xsl:value-of select="$inputBackground"/>
```

```
      </xsl:when>
  <xsl:otherwise>
      <xsl:if test="not(../extendedAttributes/@color)">
      <xsl:choose>
          <xsl:when test="not(@intensity='high') and not(@protected='true')">
              <xsl:value-of select="$green"/>
          </xsl:when>
          <xsl:when test="@intensity='high' and not(@protected='true')">
              <xsl:value-of select="$red"/>
          </xsl:when>
          <xsl:when test="not(@intensity='high') and @protected='true'">
              <xsl:value-of select="$turquoise"/>
          </xsl:when>
          <xsl:when test="@intensity='high' and @protected='true'">
              <xsl:value-of select="$neutral"/>
          </xsl:when>
      </xsl:choose>
      </xsl:if>
  </xsl:otherwise>
  </xsl:choose>
  </xsl:template>

  <!-- extendedAttributes template: specify color and highlighting assignment,
overrides attribute colors -->
  <xsl:template match="extendedAttributes">
  <!--  Based on the parameter passed in, determine whether to specify
highlighting assignment or not -->
  <!--  When param='style', both color and highlighting assignment will be
produced;                  -->
  <!--  When param='font', only color assignment will be produced.
-->
  <xsl:param name="tag">style</xsl:param>
  <xsl:if test="not(../attributes/@intensity='nondisplayable')">
      <!-- Specify highlighting reverse background color -->
      <xsl:if test="@highlighting='reverse'">
          <xsl:if test="$tag='style'">
              <xsl:choose>
                  <!-- Reverse background color for label and input are
different -->
                  <xsl:when test="../attributes/@protected='true'">
                      <!-- For label text, assign background color -->
                      <xsl:value-of select="$background"/>
                  </xsl:when>
              <xsl:otherwise>
                  <!-- For input fields, assign input color -->
                  <xsl:value-of select="$input"/>
              </xsl:otherwise>
              </xsl:choose>
              <xsl:text>; background-color: </xsl:text>
          </xsl:if>
          <xsl:if test="$tag='font'">
              <xsl:value-of select="$background"/>
          </xsl:if>
      </xsl:if>
      <!-- Specify color assignment for label text -->
      <xsl:if test="not(@highlighting='reverse' and $tag='font') or
$tag='border'">
```

```
            <xsl:choose>
                <xsl:when test="@color='blue'">
                    <xsl:value-of select="$blue"/>
                </xsl:when>
                <xsl:when test="@color='red'">
                    <xsl:value-of select="$red"/>
                </xsl:when>
                <xsl:when test="@color='green'">
                    <xsl:value-of select="$green"/>
                </xsl:when>
                <xsl:when test="@color='pink'">
                    <xsl:value-of select="$pink"/>
                </xsl:when>
                <xsl:when test="@color='turquoise'">
                    <xsl:value-of select="$turquoise"/>
                </xsl:when>
                <xsl:when test="@color='yellow'">
                    <xsl:value-of select="$yellow"/>
                </xsl:when>
                <xsl:when test="@color='default'">
                    <xsl:value-of select="$default"/>
                </xsl:when>
                <xsl:when test="@color='neutral'">
                    <xsl:value-of select="$neutral"/>
                </xsl:when>
            </xsl:choose>
        </xsl:if>
        <!-- Specify highlighting underline and border assignment -->
        <xsl:if test="$tag='style'">
            <xsl:if test="@highlighting">
                <xsl:choose>
                    <xsl:when test="@highlighting='underline'">
                        <xsl:text>; text-decoration:underline</xsl:text>
                    </xsl:when>
                    <xsl:when test="@highlighting='blink'">
                        <xsl:text>; text-decoration:blink</xsl:text>
                    </xsl:when>
                </xsl:choose>
                <!-- Specify border assignment -->
                <xsl:if test="outlining">
                    <!-- Get the border color -->
                    <!-- Recursively call extendedAttributes with parameter
tag=border to get border-color assignment -->
                    <xsl:text>; border-color:</xsl:text>
                    <xsl:apply-templates select=".">
                        <xsl:with-param name="tag">border</xsl:with-param>
                    </xsl:apply-templates>
                    <!-- Assign border: box, right, left, top, or bottom -->
                    <xsl:choose>
                        <xsl:when test="outlining/@box='true'">
                            <xsl:text>; border-style: solid</xsl:text>
                        </xsl:when>
                        <xsl:when test="outlining/@right='true' or
outlining/@left='true' or outlining/@over='true' or outlining/@under='true'">
                            <xsl:text>; border-style: solid; border-right-width:
</xsl:text>
                            <xsl:if test="outlining/@right='true'">
```

```
                        <xsl:text>medium</xsl:text>
                    </xsl:if>
                    <xsl:if test="not(outlining/@right='true')">
                        <xsl:text>0</xsl:text>
                    </xsl:if>
                    <xsl:text>; border-left-width: </xsl:text>
                    <xsl:if test="outlining/@left='true'">
                        <xsl:text>medium</xsl:text>
                    </xsl:if>
                    <xsl:if test="not(outlining/@left='true')">
                        <xsl:text>0</xsl:text>
                    </xsl:if>
                    <xsl:text>; border-top-width: </xsl:text>
                    <xsl:if test="outlining/@over='true'">
                        <xsl:text>medium</xsl:text>
                    </xsl:if>
                    <xsl:if test="not(outlining/@over='true')">
                        <xsl:text>0</xsl:text>
                    </xsl:if>
                    <xsl:text>; border-bottom-width: </xsl:text>
                    <xsl:if test="outlining/@under='true'">
                        <xsl:text>medium</xsl:text>
                    </xsl:if>
                    <xsl:if test="not(outlining/@under='true')">
                        <xsl:text>0</xsl:text>
                    </xsl:if>
                </xsl:when>
            </xsl:choose>
        </xsl:if>
    </xsl:if>
</xsl:if>
</xsl:template>

</xsl:stylesheet>
```