GIMA

Geographical Information Management and Applications

Integrating data:

Appendix F, G and H (full)

GIMA Thesis

Ing. Huibert-Jan Lekkerkerk B.Sc.

Table of contents

Appendix F: **MAPPING TO REFERENCE SET**

This Appendix shows the generated XSLT2 and OML code for the transformation of the location reference table as computed in Chapter 6 to the INSPIRE Gazetteer and INSPIRE Geographical Names table as described in Chapter 7.

F.1 XSLT2: Gazetteer

XSLT2: http://gima.lekkerkerk.info/ProofOfConcept/ReferenceSet/MappingMapToGazetteer.xslt

HTML: http://gima.lekkerkerk.info/ProofOfConcept/ReferenceSet/mpn_nl_all_geonames.html

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This file was generated by Altova MapForce 2011r3

YOU SHOULD NOT MODIFY THIS FILE, BECAUSE IT WILL BE
OVERWRITTEN WHEN YOU RE-RUN CODE GENERATION.-->
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:core="http://www.altova.com/MapForce/UDF/core" xmlns:vmf="http://www.altova.com/MapForce/UDF/vmf"
xmlns:grp="http://www.altova.com/MapForce/grouping" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:fn="http://www.w3.org/2005/xpath-functions" exclude-result-
prefixes="core vmf grp xs fn">
 <xsl:template name="core:firstCharacter">
  <xsl:param name="value" select="()"/>
  <xsl:param name="default" select="()"/>
  <xsl:choose>
   <xsl:when test="(fn:string-length($value) = xs:integer('0'))">
    <xsl:sequence select="($default)"/>
   </xsl:when>
   <xsl:otherwise>
    <xsl:sequence select="(fn:substring($value, xs:double(xs:integer('1')), xs:double(xs:integer('1'))))"/>
   </xsl:otherwise>
  </xsl:choose>
 </xsl:template>
 <xsl:template name="vmf:vmf1_inputtoresult">
  <xsl:param name="input" select="()"/>
  <xsl:choose>
   <xsl:when test="$input='BWN'">  <xsl:value-of select="'other: BWN;Bekenwerkgroep Nederland'"/>  </xsl:when>
   <xsl:when test="$input='HHD'">  <xsl:value-of select="'15'"/>  </xsl:when>
   <xsl:when test="$input='HHN'">  <xsl:value-of select="'12'"/>  </xsl:when>
   <xsl:when test="$input='HHR'">  <xsl:value-of select="'13'"/>  </xsl:when>
   <xsl:when test="$input='HHS'">  <xsl:value-of select="'39'"/>  </xsl:when>
   <xsl:when test="$input='HSR'">  <xsl:value-of select="'14'"/>  </xsl:when>
   <xsl:when test="$input='KUN'">  <xsl:value-of select="'other:KUN;Katholieke Universiteit Nijmegen'"/>  </xsl:when>
   <xsl:when test="$input='PGR'">  <xsl:value-of select="'61'"/>  </xsl:when>
   <xsl:when test="$input='PNH'">  <xsl:value-of select="'65'"/>  </xsl:when>
   <xsl:when test="$input='PROV'">  <xsl:value-of select="'66'"/>  </xsl:when>
   <xsl:when test="$input='PRF'">  <xsl:value-of select="'62'"/>  </xsl:when>
   <xsl:when test="$input='PRU'">  <xsl:value-of select="'67'"/>  </xsl:when>
   <xsl:when test="$input='PSC'">  <xsl:value-of select="'other:PSC;Piscaria'"/>  </xsl:when>
   <xsl:when test="$input='RWS'">  <xsl:value-of select="'80'"/>  </xsl:when>
   <xsl:when test="$input='STO'">  <xsl:value-of select="'other:STO;STOWA'"/>  </xsl:when>
   <xsl:when test="$input='WA'">  <xsl:value-of select="'other:WA;Waterleidingbedrijf Amsterdam'"/>  </xsl:when>
   <xsl:when test="$input='WAM'">  <xsl:value-of select="'38'"/>  </xsl:when>
   <xsl:when test="$input='WAN'">  <xsl:value-of select="'11'"/>  </xsl:when>
   <xsl:when test="$input='WBD'">  <xsl:value-of select="'25'"/>  </xsl:when>
   <xsl:when test="$input='WD'">  <xsl:value-of select="'27'"/>  </xsl:when>
   <xsl:when test="$input='WF'">  <xsl:value-of select="'02'"/>  </xsl:when>
   <xsl:when test="$input='WGS'">  <xsl:value-of select="'04'"/>  </xsl:when>
   <xsl:when test="$input='WHA'">  <xsl:value-of select="'33'"/>  </xsl:when>
   <xsl:when test="$input='WHD'">  <xsl:value-of select="'40'"/>  </xsl:when>
   <xsl:when test="$input='WN'">  <xsl:value-of select="'34'"/>  </xsl:when>
   <xsl:when test="$input='WPM'">  <xsl:value-of select="'57'"/>  </xsl:when>
   <xsl:when test="$input='WRD'">  <xsl:value-of select="'05'"/>  </xsl:when>
   <xsl:when test="$input='WRIJ'">  <xsl:value-of select="'07'"/>  </xsl:when>
   <xsl:when test="$input='WRL'">  <xsl:value-of select="'09'"/>  </xsl:when>
   <xsl:when test="$input='WRO'">  <xsl:value-of select="'58'"/>  </xsl:when>
   <xsl:when test="$input='WRW'">  <xsl:value-of select="'35'"/>  </xsl:when>
   <xsl:when test="$input='WV'">  <xsl:value-of select="'08'"/>  </xsl:when>
   <xsl:when test="$input='WVE'">  <xsl:value-of select="'10'"/>  </xsl:when>
   <xsl:when test="$input='WVV'">  <xsl:value-of select="'36'"/>  </xsl:when>
   <xsl:when test="$input='WZE'">  <xsl:value-of select="'18'"/>  </xsl:when>
   <xsl:when test="$input='WZV'">  <xsl:value-of select="'23'"/>  </xsl:when>
```

```
    <xsl:when test="$input='WZZ'">      <xsl:value-of select="'37'"/>   </xsl:when>
    <xsl:otherwise>       <xsl:value-of select="'other:unknown'"/>     </xsl:otherwise>
  </xsl:choose>
 </xsl:template>
 <xsl:template name="vmf:vmf2_inputtoresult">
  <xsl:param name="input" select="()"/>
  <xsl:choose>
    <xsl:when test="$input='15'">      <xsl:value-of select="'15'"/>   </xsl:when>
    <xsl:when test="$input='12'">      <xsl:value-of select="'12'"/>   </xsl:when>
    <xsl:when test="$input='13'">      <xsl:value-of select="'13'"/>   </xsl:when>
    <xsl:when test="$input='16'">      <xsl:value-of select="'39'"/>   </xsl:when>
    <xsl:when test="$input='14'">      <xsl:value-of select="'14'"/>   </xsl:when>
    <xsl:when test="$input='11'">      <xsl:value-of select="'11'"/>   </xsl:when>
    <xsl:when test="$input='25'">      <xsl:value-of select="'25'"/>   </xsl:when>
    <xsl:when test="$input='27'">      <xsl:value-of select="'27'"/>   </xsl:when>
    <xsl:when test="$input='02'">      <xsl:value-of select="'02'"/>   </xsl:when>
    <xsl:when test="$input='04'">      <xsl:value-of select="'04'"/>   </xsl:when>
    <xsl:when test="$input='03b'">      <xsl:value-of select="'33'"/>   </xsl:when>
    <xsl:when test="$input='17'">      <xsl:value-of select="'40'"/>   </xsl:when>
    <xsl:when test="$input='01'">      <xsl:value-of select="'34'"/>   </xsl:when>
    <xsl:when test="$input='29'">      <xsl:value-of select="'57'"/>   </xsl:when>
    <xsl:when test="$input='05'">      <xsl:value-of select="'05'"/>   </xsl:when>
    <xsl:when test="$input='07'">      <xsl:value-of select="'07'"/>   </xsl:when>
    <xsl:when test="$input='09'">      <xsl:value-of select="'09'"/>   </xsl:when>
    <xsl:when test="$input='30'">      <xsl:value-of select="'58'"/>   </xsl:when>
    <xsl:when test="$input='08'">      <xsl:value-of select="'08'"/>   </xsl:when>
    <xsl:when test="$input='10'">      <xsl:value-of select="'10'"/>   </xsl:when>
    <xsl:when test="$input='05a'">      <xsl:value-of select="'36'"/>   </xsl:when>
    <xsl:when test="$input='20'">      <xsl:value-of select="'18'"/>   </xsl:when>
    <xsl:when test="$input='23'">      <xsl:value-of select="'23'"/>   </xsl:when>
    <xsl:when test="$input='06'">      <xsl:value-of select="'37'"/>   </xsl:when>
    <xsl:when test="$input='28'">      <xsl:value-of select="'38'"/>   </xsl:when>
    <xsl:when test="$input='03a'">      <xsl:value-of select="'35'"/>   </xsl:when>
    <xsl:when test="$input='40'">      <xsl:value-of select="'81'"/>   </xsl:when>
    <xsl:when test="$input='43'">      <xsl:value-of select="'92'"/>   </xsl:when>
    <xsl:when test="$input='45'">      <xsl:value-of select="'95'"/>   </xsl:when>
    <xsl:when test="$input='47'">      <xsl:value-of select="'93'"/>   </xsl:when>
    <xsl:when test="$input='48'">      <xsl:value-of select="'94'"/>   </xsl:when>
    <xsl:when test="$input='49'">      <xsl:value-of select="'89'"/>   </xsl:when>
    <xsl:when test="$input='50'">      <xsl:value-of select="'91'"/>   </xsl:when>
    <xsl:otherwise>       <xsl:value-of select="'other:unknown'"/>     </xsl:otherwise>
  </xsl:choose>
 </xsl:template>
 <xsl:output method="xml" encoding="UTF-8" indent="yes"/>
 <xsl:function name="grp:var2_function">  <xsl:param name="var1_param" as="node()"/>
  <xsl:for-each select="$var1_param/ID1">     <xsl:sequence select="xs:string(xs:integer(fn:string(.)))"/>  </xsl:for-each>
 </xsl:function>
 <xsl:template match="/">
  <gml:FeatureCollection xmlns:gml="http://www.opengis.net/gml/3.2" xmlns:gco="http://www.isotc211.org/2005/gco"
xmlns:gmd="http://www.isotc211.org/2005/gmd" xmlns:gsr="http://www.isotc211.org/2005/gsr"
xmlns:gss="http://www.isotc211.org/2005/gss" xmlns:gts="http://www.isotc211.org/2005/gts"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:gaz="urn:x-inspire:specification:gmlas:Gazetteer:3.2">
    <xsl:attribute name="xsi:schemaLocation" select="'urn:x-inspire:specification:gmlas:Gazetteer:3.2 Y:/projects/P0902-
GIMA/MODULE~9/03-SOLL/inspire/XSD/Gazetteer.xsd'"/>
    <xsl:attribute name="gml:id" select="'_f8b5d793-2ade-488e-a89c-9142e0b1528f'"/>
    <gml:featureMembers>
     <xsl:for-each-group select="Import/Row" group-by="grp:var2_function(.)">
      <xsl:variable name="var3_" as="xs:double" select="xs:double(xs:decimal('2'))"/>
      <xsl:variable name="var4_resultof_group_items" as="node()+" select="current-group()"/>
      <xsl:variable name="var5_resultof_concat" as="xs:string" select="fn:concat('ID_', current-grouping-key())"/>
      <xsl:variable name="var6_val" as="item()*" select="()"/>
      <xsl:variable name="var7_let" as="xs:dateTime*">
       <xsl:for-each select="$var4_resultof_group_items/DatIn2[fn:not((fn:translate(fn:string(@xsi:nil), 'true ', '1') = '1'))]">
        <xsl:sequence select="xs:dateTime(fn:string(.))"/>
       </xsl:for-each>
      </xsl:variable>
      <xsl:variable name="var8_result" as="xs:dateTime+">
       <xsl:choose>
        <xsl:when test="fn:exists($var7_let)">          <xsl:sequence select="$var7_let"/>          </xsl:when>
        <xsl:otherwise>          <xsl:sequence select="xs:dateTime('9999-01-01T00:00:00')"/>          </xsl:otherwise>
       </xsl:choose>
      </xsl:variable>
      <xsl:variable name="var9_result" as="xs:decimal+">
       <xsl:for-each select="$var8_result"> <xsl:sequence select="xs:decimal(format-dateTime(., '[Y][M,2][D,2]'))"/></xsl:for-each>
      </xsl:variable>
      <xsl:variable name="var10_resultof_string" as="xs:string" select="fn:string(fn:min($var9_result))"/>
```

```
<gaz:LocationInstance>
 <xsl:attribute name="gml:id" select="$var5_resultof_concat"/>
 <xsl:variable name="var11_result" as="xs:string*">
  <xsl:for-each select="$var4_resultof_group_items/Descr2">  <xsl:sequence select="fn:string(.)"/>  </xsl:for-each>
 </xsl:variable>
 <gml:description>
  <xsl:sequence select="fn:string-join(fn:distinct-values($var11_result), ' | ')"/>
 </gml:description>
 <gml:identifier>
  <xsl:attribute name="codeSpace" select="xs:string(xs:anyURI('urn:waterdatabase'))"/>
  <xsl:sequence select="$var5_resultof_concat"/>
 </gml:identifier>
 <xsl:for-each select="$var4_resultof_group_items">
  <xsl:variable name="var12_cur" as="node()" select="."/>
  <xsl:for-each select="LocalID2">
   <gml:name>
    <xsl:for-each select="$var12_cur/Source2">
     <xsl:attribute name="codeSpace" select="xs:string(xs:anyURI(fn:string(.)))"/>
    </xsl:for-each>
    <xsl:sequence select="fn:string(.)"/>
   </gml:name>
  </xsl:for-each>
 </xsl:for-each>
 <xsl:variable name="var13_result" as="xs:boolean+">
  <xsl:for-each select="$var4_resultof_group_items">  <xsl:sequence select="fn:exists(Naam1)"/>  </xsl:for-each>
 </xsl:variable>
 <xsl:variable name="var14_result" as="node()+">
  <xsl:choose>
   <xsl:when test="fn:exists($var13_result[.])">
    <xsl:for-each select="$var4_resultof_group_items/Naam1">
     <dummy>  <xsl:sequence select="fn:string(.)"/>  </dummy>
    </xsl:for-each>
   </xsl:when>
   <xsl:otherwise>  <dummy> <xsl:attribute name="xsi:nil" select="'true'"/> </dummy>  </xsl:otherwise>
  </xsl:choose>
 </xsl:variable>
 <xsl:variable name="var15_result" as="xs:string+">
  <xsl:for-each select="$var14_result">  <xsl:sequence select="fn:string(.)"/>  </xsl:for-each>
 </xsl:variable>
 <xsl:for-each select="fn:distinct-values($var15_result)">
  <gaz:geographicIdentifier>
   <xsl:sequence select="."/>
  </gaz:geographicIdentifier>
 </xsl:for-each>
 <xsl:variable name="var16_result" as="xs:string*">
  <xsl:for-each select="$var4_resultof_group_items/Naam2">  <xsl:sequence select="fn:string(.)"/>  </xsl:for-each>
 </xsl:variable>
 <xsl:for-each select="fn:distinct-values($var16_result)">
  <gaz:alternativeGeographicIdentifier>
   <xsl:sequence select="."/>
  </gaz:alternativeGeographicIdentifier>
 </xsl:for-each>
 <gaz:dateOfCreation>
  <xsl:sequence select="xs:string(xs:date(fn:concat(fn:concat(fn:concat(fn:concat(fn:substring($var10_resultof_string,
xs:double(xs:decimal('1')), xs:double(xs:decimal('4'))), '-'), fn:substring($var10_resultof_string, xs:double(xs:decimal('5')), $var3_)), '-
'), fn:substring($var10_resultof_string, xs:double(xs:decimal('7')), $var3_))))"/>
 </gaz:dateOfCreation>
 <gaz:geographicExtent>
  <gml:Point>
   <xsl:attribute name="gml:id" select="fn:concat(fn:concat($var5_resultof_concat, '_'),
xs:string(fn:count($var4_resultof_group_items/X2)))"/>
   <xsl:variable name="var17_result" as="xs:boolean+">
    <xsl:for-each select="$var4_resultof_group_items"> <xsl:sequence select="fn:exists(X2)"/>  </xsl:for-each>
   </xsl:variable>
   <xsl:if test="fn:exists($var17_result[.])">
    <xsl:variable name="var18_result" as="xs:boolean+">
     <xsl:for-each select="$var4_resultof_group_items"><xsl:sequence select="fn:exists(Y2)"/> </xsl:for-each>
    </xsl:variable>
    <xsl:if test="fn:exists($var18_result[.])">
     <xsl:variable name="var19_resultof_firstCharacter" as="xs:string">
      <xsl:call-template name="core:firstCharacter">
       <xsl:with-param name="value" select="'.'" as="xs:string"/>
       <xsl:with-param name="default" select="'.'" as="xs:string"/>
      </xsl:call-template>
     </xsl:variable>
     <xsl:variable name="var20_resultof_firstCharacter" as="xs:string">
```

```
            <xsl:call-template name="core:firstCharacter">
              <xsl:with-param name="value" select="','" as="xs:string"/>
              <xsl:with-param name="default" select="','" as="xs:string"/>
            </xsl:call-template>
          </xsl:variable>
          <xsl:variable name="var21_resultof_concat" as="xs:string" select="fn:concat($var19_resultof_firstCharacter,
$var20_resultof_firstCharacter)"/>
          <xsl:variable name="var22_result" as="xs:decimal*">
            <xsl:for-each select="$var4_resultof_group_items/X2">
              <xsl:sequence select="xs:decimal(fn:string(.))"/>
            </xsl:for-each>
          </xsl:variable>
          <xsl:variable name="var23_result" as="xs:decimal*">
            <xsl:for-each select="$var4_resultof_group_items/Y2">
              <xsl:sequence select="xs:decimal(fn:string(.))"/>
            </xsl:for-each>
          </xsl:variable>
          <gml:pos>
            <xsl:sequence select="fn:concat(fn:concat(fn:translate(format-number(fn:avg($var22_result), '#0.00#'), '.,',
$var21_resultof_concat), ' '), fn:translate(format-number(fn:avg($var23_result), '#0.00#'), '.,', $var21_resultof_concat))"/>
          </gml:pos>
        </xsl:if>
      </xsl:if>
    </gml:Point>
  </gaz:geographicExtent>
  <gaz:admin>
    <gmd:CI_ResponsibleParty>
      <xsl:variable name="var29_result" as="xs:string*">
        <xsl:for-each select="$var4_resultof_group_items">
          <xsl:variable name="var28_cur" as="node()" select="."/>
          <xsl:for-each select="Source1">
            <xsl:variable name="var24_resultof_cast" as="xs:string" select="fn:string(.)"/>
            <xsl:variable name="var25_WBHCode" as="item()*" select="$var28_cur/WBHCode1"/>
            <xsl:variable name="var26_WBHCode" as="node()?" select="$var25_WBHCode"/>
            <xsl:choose>
              <xsl:when test="('LD' = $var24_resultof_cast)">
                <xsl:for-each select="$var26_WBHCode">
                  <xsl:call-template name="vmf:vmf1_inputtoresult">
                    <xsl:with-param name="input" select="fn:string(.)" as="xs:string"/>
                  </xsl:call-template>
                </xsl:for-each>
              </xsl:when>
              <xsl:when test="('WFD_BW' = $var24_resultof_cast)">
                <xsl:for-each select="$var26_WBHCode">
                  <xsl:call-template name="vmf:vmf2_inputtoresult">
                    <xsl:with-param name="input" select="fn:string(.)" as="xs:string"/>
                  </xsl:call-template>
                </xsl:for-each>
              </xsl:when>
              <xsl:otherwise>
                <xsl:variable name="var27_let" as="xs:string?">
                  <xsl:for-each select="$var26_WBHCode">  <xsl:sequence select="fn:string(.)"/>  </xsl:for-each>
                </xsl:variable>
                <xsl:choose>
                  <xsl:when test="fn:exists($var27_let)"> <xsl:sequence select="$var27_let"/> </xsl:when>
                  <xsl:otherwise>  <xsl:sequence select="'other:unknown'"/> </xsl:otherwise>
                </xsl:choose>
              </xsl:otherwise>
            </xsl:choose>
          </xsl:for-each>
        </xsl:for-each>
      </xsl:variable>
      <xsl:for-each select="fn:distinct-values($var29_result)"> <xsl:attribute name="uuid" select="."/></xsl:for-each>
      <gmd:individualName> <xsl:sequence select="$var6_val"/></gmd:individualName>
      <gmd:organisationName><xsl:sequence select="$var6_val"/>    </gmd:organisationName>
      <gmd:positionName><xsl:sequence select="$var6_val"/>            </gmd:positionName>
      <gmd:contactInfo>              <xsl:sequence select="$var6_val"/>            </gmd:contactInfo>
      <gmd:role>              <xsl:sequence select="$var6_val"/>            </gmd:role>
    </gmd:CI_ResponsibleParty>
  </gaz:admin>
  <gaz:spatialObject>          <xsl:attribute name="xsi:nil" select="'true'"/>         </gaz:spatialObject>
  <gaz:locationType>          <xsl:sequence select="$var6_val"/>         </gaz:locationType>
  <gaz:gazetteer>          <xsl:sequence select="$var6_val"/>         </gaz:gazetteer>
  <gaz:parent>          <xsl:sequence select="$var6_val"/>         </gaz:parent>
</gaz:LocationInstance>
</xsl:for-each-group>
```

```
    </gml:featureMembers>
  </gml:FeatureCollection>

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

F.2 XSLT2: Geographical Names

XSLT2: http://gima.lekkerkerk.info/ProofOfConcept/ReferenceSet/MappingMapToGeographicalNames.xslt

HTML: http://gima.lekkerkerk.info/ProofOfConcept/ReferenceSet/mpn_nl_all_gazetteer.html

XML from XSLT2:
http://gima.lekkerkerk.info/ProofOfConcept/ReferenceSet/MPN_GeographicalNames_Altova_RDIJ.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!--This file was generated by Altova MapForce 2011r3

YOU SHOULD NOT MODIFY THIS FILE, BECAUSE IT WILL BE
OVERWRITTEN WHEN YOU RE-RUN CODE GENERATION. -->
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:core="http://www.altova.com/MapForce/UDF/core" xmlns:vmf="http://www.altova.com/MapForce/UDF/vmf"
xmlns:grp="http://www.altova.com/MapForce/grouping" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:fn="http://www.w3.org/2005/xpath-functions" exclude-result-
prefixes="core vmf grp xs fn">
 <xsl:template name="core:firstCharacter">
  <xsl:param name="value" select="()"/>
  <xsl:param name="default" select="()"/>
  <xsl:choose>
   <xsl:when test="(fn:string-length($value) = xs:integer('0'))">  <xsl:sequence select="($default)"/>  </xsl:when>
   <xsl:otherwise>
    <xsl:sequence select="(fn:substring($value, xs:double(xs:integer('1')), xs:double(xs:integer('1'))))"/>
   </xsl:otherwise>
  </xsl:choose>
 </xsl:template>
 <xsl:template name="vmf:vmf1_inputtoresult">
  <xsl:param name="input" select="()"/>
  <xsl:choose>
   <xsl:when test="$input='BWN'">  <xsl:value-of select="'other: BWN;Bekenwerkgroep Nederland'"/>  </xsl:when>
   <xsl:when test="$input='HHD'">  <xsl:value-of select="'15'"/>  </xsl:when>
   <xsl:when test="$input='HHN'">  <xsl:value-of select="'12'"/>  </xsl:when>
   <xsl:when test="$input='HHR'">  <xsl:value-of select="'13'"/>  </xsl:when>
   <xsl:when test="$input='HHS'">  <xsl:value-of select="'39'"/>  </xsl:when>
   <xsl:when test="$input='HSR'">  <xsl:value-of select="'14'"/>  </xsl:when>
   <xsl:when test="$input='KUN'">  <xsl:value-of select="'other:KUN;Katholieke Universiteit Nijmegen'"/>  </xsl:when>
   <xsl:when test="$input='PGR'">  <xsl:value-of select="'61'"/>  </xsl:when>
   <xsl:when test="$input='PNH'">  <xsl:value-of select="'65'"/>  </xsl:when>
   <xsl:when test="$input='PROV'">  <xsl:value-of select="'66'"/>  </xsl:when>
   <xsl:when test="$input='PRF'">  <xsl:value-of select="'62'"/>  </xsl:when>
   <xsl:when test="$input='PRU'">  <xsl:value-of select="'67'"/>  </xsl:when>
   <xsl:when test="$input='PSC'">  <xsl:value-of select="'other:PSC;Piscaria'"/>  </xsl:when>
   <xsl:when test="$input='RWS'">  <xsl:value-of select="'80'"/>  </xsl:when>
   <xsl:when test="$input='STO'">  <xsl:value-of select="'other:STO;STOWA'"/>  </xsl:when>
   <xsl:when test="$input='WA'">  <xsl:value-of select="'other:WA;Waterleidingbedrijf Amsterdam'"/>  </xsl:when>
   <xsl:when test="$input='WAM'">  <xsl:value-of select="'38'"/>  </xsl:when>
   <xsl:when test="$input='WAN'">  <xsl:value-of select="'11'"/>  </xsl:when>
   <xsl:when test="$input='WBD'">  <xsl:value-of select="'25'"/>  </xsl:when>
   <xsl:when test="$input='WD'">  <xsl:value-of select="'27'"/>  </xsl:when>
   <xsl:when test="$input='WF'">  <xsl:value-of select="'02'"/>  </xsl:when>
   <xsl:when test="$input='WGS'">  <xsl:value-of select="'04'"/>  </xsl:when>
   <xsl:when test="$input='WHA'">  <xsl:value-of select="'33'"/>  </xsl:when>
   <xsl:when test="$input='WHD'">  <xsl:value-of select="'40'"/>  </xsl:when>
   <xsl:when test="$input='WN'">  <xsl:value-of select="'34'"/>  </xsl:when>
   <xsl:when test="$input='WPM'">  <xsl:value-of select="'57'"/>  </xsl:when>
   <xsl:when test="$input='WRD'">  <xsl:value-of select="'05'"/>  </xsl:when>
   <xsl:when test="$input='WRIJ'">  <xsl:value-of select="'07'"/>  </xsl:when>
   <xsl:when test="$input='WRL'">  <xsl:value-of select="'09'"/>  </xsl:when>
   <xsl:when test="$input='WRO'">  <xsl:value-of select="'58'"/>  </xsl:when>
   <xsl:when test="$input='WRW'">  <xsl:value-of select="'35'"/>  </xsl:when>
   <xsl:when test="$input='WV'">  <xsl:value-of select="'08'"/>  </xsl:when>
   <xsl:when test="$input='WVE'">  <xsl:value-of select="'10'"/>  </xsl:when>
   <xsl:when test="$input='WVV'">  <xsl:value-of select="'36'"/>  </xsl:when>
   <xsl:when test="$input='WZE'">  <xsl:value-of select="'18'"/>  </xsl:when>
   <xsl:when test="$input='WZV'">  <xsl:value-of select="'23'"/>  </xsl:when>
   <xsl:when test="$input='WZZ'">  <xsl:value-of select="'37'"/>  </xsl:when>
   <xsl:otherwise>  <xsl:value-of select="'other:unknown'"/>  </xsl:otherwise>
  </xsl:choose>
 </xsl:template>
 <xsl:template name="vmf:vmf2_inputtoresult">
  <xsl:param name="input" select="()"/>
```

```
    <xsl:choose>
      <xsl:when test="$input='15'">     <xsl:value-of select="'15'"/>   </xsl:when>
      <xsl:when test="$input='12'">     <xsl:value-of select="'12'"/>   </xsl:when>
      <xsl:when test="$input='13'">     <xsl:value-of select="'13'"/>   </xsl:when>
      <xsl:when test="$input='16'">     <xsl:value-of select="'39'"/>   </xsl:when>
      <xsl:when test="$input='14'">     <xsl:value-of select="'14'"/>   </xsl:when>
      <xsl:when test="$input='11'">     <xsl:value-of select="'11'"/>   </xsl:when>
      <xsl:when test="$input='25'">     <xsl:value-of select="'25'"/>   </xsl:when>
      <xsl:when test="$input='27'">     <xsl:value-of select="'27'"/>   </xsl:when>
      <xsl:when test="$input='02'">     <xsl:value-of select="'02'"/>   </xsl:when>
      <xsl:when test="$input='04'">     <xsl:value-of select="'04'"/>   </xsl:when>
      <xsl:when test="$input='03b'">     <xsl:value-of select="'33'"/>   </xsl:when>
      <xsl:when test="$input='17'">     <xsl:value-of select="'40'"/>   </xsl:when>
      <xsl:when test="$input='01'">     <xsl:value-of select="'34'"/>   </xsl:when>
      <xsl:when test="$input='29'">     <xsl:value-of select="'57'"/>   </xsl:when>
      <xsl:when test="$input='05'">     <xsl:value-of select="'05'"/>   </xsl:when>
      <xsl:when test="$input='07'">     <xsl:value-of select="'07'"/>   </xsl:when>
      <xsl:when test="$input='09'">     <xsl:value-of select="'09'"/>   </xsl:when>
      <xsl:when test="$input='30'">     <xsl:value-of select="'58'"/>   </xsl:when>
      <xsl:when test="$input='08'">     <xsl:value-of select="'08'"/>   </xsl:when>
      <xsl:when test="$input='10'">     <xsl:value-of select="'10'"/>   </xsl:when>
      <xsl:when test="$input='05a'">     <xsl:value-of select="'36'"/>   </xsl:when>
      <xsl:when test="$input='20'">     <xsl:value-of select="'18'"/>   </xsl:when>
      <xsl:when test="$input='23'">     <xsl:value-of select="'23'"/>   </xsl:when>
      <xsl:when test="$input='06'">     <xsl:value-of select="'37'"/>   </xsl:when>
      <xsl:when test="$input='28'">     <xsl:value-of select="'38'"/>   </xsl:when>
      <xsl:when test="$input='03a'">     <xsl:value-of select="'35'"/>   </xsl:when>
      <xsl:when test="$input='40'">     <xsl:value-of select="'81'"/>   </xsl:when>
      <xsl:when test="$input='43'">     <xsl:value-of select="'92'"/>   </xsl:when>
      <xsl:when test="$input='45'">     <xsl:value-of select="'95'"/>   </xsl:when>
      <xsl:when test="$input='47'">     <xsl:value-of select="'93'"/>   </xsl:when>
      <xsl:when test="$input='48'">     <xsl:value-of select="'94'"/>   </xsl:when>
      <xsl:when test="$input='49'">     <xsl:value-of select="'89'"/>   </xsl:when>
      <xsl:when test="$input='50'">     <xsl:value-of select="'91'"/>   </xsl:when>
      <xsl:otherwise>       <xsl:value-of select="'other:unknown'"/>     </xsl:otherwise>
    </xsl:choose>
  </xsl:template>
  <xsl:output method="xml" encoding="UTF-8" indent="yes"/>
  <xsl:function name="grp:var2_function">  <xsl:param name="var1_param" as="node()"/>
    <xsl:for-each select="$var1_param/ID1">     <xsl:sequence select="xs:string(xs:integer(fn:string(.)))"/>  </xsl:for-each>
  </xsl:function>
  <xsl:function name="grp:var59_function">  <xsl:param name="var58_param" as="node()"/>
    <xsl:for-each select="$var58_param/ID2">     <xsl:sequence select="xs:string(xs:integer(fn:string(.)))"/>  </xsl:for-each>
  </xsl:function>
  <xsl:template match="/">
    <gml:FeatureCollection xmlns:gml="http://www.opengis.net/gml/3.2" xmlns:gco="http://www.isotc211.org/2005/gco"
xmlns:gmd="http://www.isotc211.org/2005/gmd" xmlns:gsr="http://www.isotc211.org/2005/gsr"
xmlns:gss="http://www.isotc211.org/2005/gss" xmlns:gts="http://www.isotc211.org/2005/gts"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:base="urn:x-inspire:specification:gmlas:BaseTypes:3.2" xmlns:gn="urn:x-
inspire:specification:gmlas:GeographicalNames:3.0">
      <xsl:attribute name="xsi:schemaLocation" select="'http://www.opengis.net/gml/3.2 Y:/projects/P0902-GIMA/MODULE~9/03-
SOLL/inspire/XSD/GeographicalNames.xsd'"/>
      <xsl:attribute name="gml:id" select="'_f8b5d793-2ade-488e-a89c-9142e0b1528f'"/>
      <gml:featureMembers>
        <xsl:for-each-group select="Import/Row" group-by="grp:var2_function(.)">
          <xsl:variable name="var3_" as="xs:double" select="xs:double(xs:decimal('7'))"/>
          <xsl:variable name="var4_" as="xs:double" select="xs:double(xs:decimal('4'))"/>
          <xsl:variable name="var5_" as="xs:double" select="xs:double(xs:decimal('2'))"/>
          <xsl:variable name="var6_" as="xs:double" select="xs:double(xs:decimal('5'))"/>
          <xsl:variable name="var7_" as="xs:double" select="xs:double(xs:decimal('1'))"/>
          <xsl:variable name="var8_resultof_group_items" as="node()+" select="current-group()"/>
          <xsl:variable name="var9_resultof_concat" as="xs:string" select="fn:concat('ID_', current-grouping-key())"/>
          <xsl:variable name="var10_resultof_create_attribute" as="item()*">
            <xsl:attribute name="xsi:nil" select="'true'"/>
          </xsl:variable>
          <xsl:variable name="var11_resultof_greater" as="xs:boolean" select="(fn:count($var8_resultof_group_items/X2) &gt;
xs:decimal('1'))"/>
          <xsl:variable name="var12_let" as="xs:dateTime*">
            <xsl:for-each select="$var8_resultof_group_items/DatIn2[fn:not((fn:translate(fn:string(@xsi:nil), 'true ', '1') = '1'))]">
              <xsl:sequence select="xs:dateTime(fn:string(.))"/>
            </xsl:for-each>
          </xsl:variable>
          <xsl:variable name="var13_result" as="xs:dateTime+">
            <xsl:choose>
              <xsl:when test="fn:exists($var12_let)">             <xsl:sequence select="$var12_let"/>          </xsl:when>
              <xsl:otherwise>             <xsl:sequence select="xs:dateTime('9999-01-01T00:00:00')"/>          </xsl:otherwise>
```

```
        </xsl:choose>
      </xsl:variable>
      <xsl:variable name="var14_result" as="xs:decimal+">
        <xsl:for-each select="$var13_result">
          <xsl:sequence select="xs:decimal(format-dateTime(., '[Y][M,2][D,2]'))"/>
        </xsl:for-each>
      </xsl:variable>
      <xsl:variable name="var15_resultof_string" as="xs:string" select="fn:string(fn:min($var14_result))"/>
      <xsl:variable name="var16_let" as="xs:dateTime*">
        <xsl:for-each select="$var8_resultof_group_items/DatOut2[fn:not((fn:translate(fn:string(@xsi:nil), 'true ', '1') = '1'))]">
          <xsl:sequence select="xs:dateTime(fn:string(.))"/>
        </xsl:for-each>
      </xsl:variable>
      <xsl:variable name="var17_result" as="xs:dateTime+">
        <xsl:choose>
          <xsl:when test="fn:exists($var16_let)">          <xsl:sequence select="$var16_let"/>          </xsl:when>
          <xsl:otherwise>          <xsl:sequence select="xs:dateTime('1000-01-01T00:00:00')"/>          </xsl:otherwise>
        </xsl:choose>
      </xsl:variable>
      <xsl:variable name="var18_result" as="xs:decimal+">
        <xsl:for-each select="$var17_result">
          <xsl:sequence select="xs:decimal(format-dateTime(., '[Y][M,2][D,2]'))"/>
        </xsl:for-each>
      </xsl:variable>
      <xsl:variable name="var19_resultof_string" as="xs:string" select="fn:string(fn:max($var18_result))"/>
      <xsl:variable name="var20_" as="xs:dateTime"
select="xs:dateTime(xs:date(fn:concat(fn:concat(fn:concat(fn:concat(fn:substring($var15_resultof_string, $var7_, $var4_), '-'),
fn:substring($var15_resultof_string, $var6_, $var5_)), '-'), fn:substring($var15_resultof_string, $var3_, $var5_))))"/>
      <xsl:variable name="var21_" as="xs:dateTime"
select="xs:dateTime(xs:date(fn:concat(fn:concat(fn:concat(fn:concat(fn:substring($var19_resultof_string, $var7_, $var4_), '-'),
fn:substring($var19_resultof_string, $var6_, $var5_)), '-'), fn:substring($var19_resultof_string, $var3_, $var5_))))"/>
      <gn:NamedPlace>
        <xsl:attribute name="gml:id" select="$var9_resultof_concat"/>
        <xsl:variable name="var22_result" as="xs:string*">
          <xsl:for-each select="$var8_resultof_group_items/Descr2">          <xsl:sequence select="fn:string(.)"/></xsl:for-each>
        </xsl:variable>
        <gml:description>
          <xsl:sequence select="fn:string-join(fn:distinct-values($var22_result), ' | ')"/>
        </gml:description>
        <gml:identifier>
          <xsl:attribute name="codeSpace" select="xs:string(xs:anyURI('urn:waterdatabase'))"/>
          <xsl:sequence select="$var9_resultof_concat"/>
        </gml:identifier>
        <gn:beginLifespanVersion>
          <xsl:if test="(format-dateTime($var20_, '[Y][M,2][D,2]') = format-dateTime(xs:dateTime('9999-01-01T00:00:00'),
'[Y][M,2][D,2]'))">
            <xsl:variable name="var23_resultof_create_attribute" as="node()">
              <xsl:attribute name="nilReason" select="'other:undetermined'"/>
            </xsl:variable>
            <xsl:sequence select="$var23_resultof_create_attribute"/>
          </xsl:if>
          <xsl:sequence select="xs:string($var20_)"/>
        </gn:beginLifespanVersion>
        <gn:endLifespanVersion>
          <xsl:if test="(format-dateTime($var21_, '[Y][M,2][D,2]') = format-dateTime(xs:dateTime('1000-01-01T00:00:00'),
'[Y][M,2][D,2]'))">
            <xsl:variable name="var24_resultof_create_attribute" as="node()">
              <xsl:attribute name="nilReason" select="'other:undetermined'"/>
            </xsl:variable>
            <xsl:sequence select="$var24_resultof_create_attribute"/>
          </xsl:if>
          <xsl:sequence select="xs:string($var21_)"/>
        </gn:endLifespanVersion>
        <gn:geometry>
          <gml:Point>
            <xsl:attribute name="gml:id" select="fn:concat($var9_resultof_concat, '_xy')"/>
            <xsl:variable name="var25_result" as="xs:boolean+">
              <xsl:for-each select="$var8_resultof_group_items">          <xsl:sequence select="fn:exists(X2)"/> </xsl:for-each>
            </xsl:variable>
            <xsl:if test="fn:exists($var25_result[.])">
              <xsl:variable name="var26_result" as="xs:boolean+">
                <xsl:for-each select="$var8_resultof_group_items">          <xsl:sequence select="fn:exists(Y2)"/> </xsl:for-each>
              </xsl:variable>
              <xsl:if test="fn:exists($var26_result[.])">
                <xsl:variable name="var27_resultof_firstCharacter" as="xs:string">
                  <xsl:call-template name="core:firstCharacter">
```

```
                <xsl:with-param name="value" select="'.'" as="xs:string"/>
                <xsl:with-param name="default" select="'.'" as="xs:string"/>
              </xsl:call-template>
            </xsl:variable>
            <xsl:variable name="var28_resultof_firstCharacter" as="xs:string">
              <xsl:call-template name="core:firstCharacter">
                <xsl:with-param name="value" select="','" as="xs:string"/>
                <xsl:with-param name="default" select="','" as="xs:string"/>
              </xsl:call-template>
            </xsl:variable>
            <xsl:variable name="var29_resultof_concat" as="xs:string" select="fn:concat($var27_resultof_firstCharacter,
$var28_resultof_firstCharacter)"/>
            <xsl:variable name="var30_result" as="xs:decimal*">
              <xsl:for-each select="$var8_resultof_group_items/X2">
                <xsl:sequence select="xs:decimal(fn:string(.))"/>
              </xsl:for-each>
            </xsl:variable>
            <xsl:variable name="var31_result" as="xs:decimal*">
              <xsl:for-each select="$var8_resultof_group_items/Y2">
                <xsl:sequence select="xs:decimal(fn:string(.))"/>
              </xsl:for-each>
            </xsl:variable>
            <gml:pos>
              <xsl:sequence select="fn:concat(fn:concat(fn:translate(format-number(fn:avg($var30_result), '#0.00#'), '.',
$var29_resultof_concat), ' '), fn:translate(format-number(fn:avg($var31_result), '#0.00#'), '.', $var29_resultof_concat))"/>
            </gml:pos>
          </xsl:if>
        </xsl:if>
      </gml:Point>
    </gn:geometry>
    <gn:inspireId>
      <base:Identifier>
        <xsl:variable name="var32_result" as="xs:string*">
          <xsl:for-each select="$var8_resultof_group_items/Ident1"><xsl:sequence select="fn:string(.)"/> </xsl:for-each>
        </xsl:variable>
        <xsl:for-each select="fn:distinct-values($var32_result)">
          <base:localId>                <xsl:sequence select="."/>                </base:localId>
        </xsl:for-each>
        <xsl:variable name="var38_result" as="xs:string*">
          <xsl:for-each select="$var8_resultof_group_items">
            <xsl:variable name="var37_cur" as="node()" select="."/>
            <xsl:for-each select="Source1">
              <xsl:variable name="var33_resultof_cast" as="xs:string" select="fn:string(.)"/>
              <xsl:variable name="var34_WBHCode" as="item()*" select="$var37_cur/WBHCode1"/>
              <xsl:variable name="var35_WBHCode" as="node()?" select="$var34_WBHCode"/>
              <xsl:choose>
                <xsl:when test="('LD' = $var33_resultof_cast)">
                  <xsl:for-each select="$var35_WBHCode">
                    <xsl:call-template name="vmf:vmf1_inputtoresult">
                      <xsl:with-param name="input" select="fn:string(.)" as="xs:string"/>
                    </xsl:call-template>
                  </xsl:for-each>
                </xsl:when>
                <xsl:when test="('WFD_BW' = $var33_resultof_cast)">
                  <xsl:for-each select="$var35_WBHCode">
                    <xsl:call-template name="vmf:vmf2_inputtoresult">
                      <xsl:with-param name="input" select="fn:string(.)" as="xs:string"/>
                    </xsl:call-template>
                  </xsl:for-each>
                </xsl:when>
                <xsl:otherwise>
                  <xsl:variable name="var36_let" as="xs:string?">
                    <xsl:for-each select="$var35_WBHCode">
                      <xsl:sequence select="fn:string(.)"/>
                    </xsl:for-each>
                  </xsl:variable>
                  <xsl:choose>
                    <xsl:when test="fn:exists($var36_let)">
                      <xsl:sequence select="$var36_let"/>
                    </xsl:when>
                    <xsl:otherwise>
                      <xsl:sequence select="'other:unknown'"/>
                    </xsl:otherwise>
                  </xsl:choose>
                </xsl:otherwise>
              </xsl:choose>
```

```
      </xsl:for-each>
    </xsl:for-each>
  </xsl:variable>
  <xsl:for-each select="fn:distinct-values($var38_result)">
    <base:namespace>                <xsl:sequence select="."/>                </base:namespace>
  </xsl:for-each>
  <base:versionId> <xsl:sequence select="xs:string(fn:count($var8_resultof_group_items/X2))"/></base:versionId>
 </base:Identifier>
</gn:inspireId>
<gn:leastDetailedViewingResolution>
 <xsl:sequence select="$var10_resultof_create_attribute"/>
</gn:leastDetailedViewingResolution>
<gn:localType>  <xsl:sequence select="$var10_resultof_create_attribute"/>    </gn:localType>
<gn:mostDetailedViewingResolution>
 <gmd:MD_Resolution>
  <gmd:distance>
   <xsl:if test="fn:not((fn:count($var8_resultof_group_items/X2) &gt; xs:decimal('1')))">
    <xsl:attribute name="gco:nilReason" select="'other:undetermined'"/>
   </xsl:if>
   <xsl:variable name="var57_result" as="xs:boolean">
    <xsl:choose>
     <xsl:when test="$var11_resultof_greater">
      <xsl:variable name="var39_result" as="xs:boolean+">
       <xsl:for-each select="$var8_resultof_group_items"> <xsl:sequence select="fn:exists(X2)"/> </xsl:for-each>
      </xsl:variable>
      <xsl:choose>
       <xsl:when test="fn:exists($var39_result[.])">
        <xsl:variable name="var40_result" as="xs:boolean+">
         <xsl:for-each select="$var8_resultof_group_items"> <xsl:sequence select="fn:exists(X2)"/> </xsl:for-each>
        </xsl:variable>
        <xsl:choose>
         <xsl:when test="fn:exists($var40_result[.])">
          <xsl:variable name="var41_result" as="xs:boolean+">
           <xsl:for-each select="$var8_resultof_group_items"> <xsl:sequence select="fn:exists(Y2)"/></xsl:for-each>
          </xsl:variable>
          <xsl:choose>
           <xsl:when test="fn:exists($var41_result[.])">
            <xsl:variable name="var42_result" as="xs:boolean+">
             <xsl:for-each select="$var8_resultof_group_items"> <xsl:sequence select="fn:exists(Y2)"/> </xsl:for-each>
            </xsl:variable>
            <xsl:sequence select="fn:exists($var42_result[.])"/>
           </xsl:when>
           <xsl:otherwise> <xsl:sequence select="fn:false()"/> </xsl:otherwise>
          </xsl:choose>
         </xsl:when>
         <xsl:otherwise> <xsl:sequence select="fn:false()"/> </xsl:otherwise>
        </xsl:choose>
       </xsl:when>
       <xsl:otherwise> <xsl:sequence select="fn:false()"/></xsl:otherwise>
      </xsl:choose>
     </xsl:when>
     <xsl:otherwise> <xsl:sequence select="fn:false()"/> </xsl:otherwise>
    </xsl:choose>
   </xsl:variable>
   <xsl:if test="$var57_result">
    <gco:Distance>
     <xsl:attribute name="uom" select="'urn:aquo:eenheid:code:m'"/>
     <xsl:variable name="var56_result" as="xs:string">
      <xsl:if test="$var11_resultof_greater">
       <xsl:variable name="var43_result" as="xs:boolean+">
        <xsl:for-each select="$var8_resultof_group_items"> <xsl:sequence select="fn:exists(X2)"/> </xsl:for-each>
       </xsl:variable>
       <xsl:if test="fn:exists($var43_result[.])">
        <xsl:variable name="var44_result" as="xs:boolean+">
         <xsl:for-each select="$var8_resultof_group_items"> <xsl:sequence select="fn:exists(X2)"/> </xsl:for-each>
        </xsl:variable>
        <xsl:if test="fn:exists($var44_result[.])">
         <xsl:variable name="var45_result" as="xs:boolean+">
          <xsl:for-each select="$var8_resultof_group_items"> <xsl:sequence select="fn:exists(Y2)"/> </xsl:for-each>
         </xsl:variable>
         <xsl:if test="fn:exists($var45_result[.])">
          <xsl:variable name="var46_result" as="xs:boolean+">
           <xsl:for-each select="$var8_resultof_group_items"> <xsl:sequence select="fn:exists(Y2)"/>    </xsl:for-each>
          </xsl:variable>
          <xsl:if test="fn:exists($var46_result[.])">
           <xsl:variable name="var47_result" as="xs:decimal*">
```

```
                        <xsl:for-each select="$var8_resultof_group_items/Y2">
                          <xsl:sequence select="xs:decimal(fn:string(.))"/>
                        </xsl:for-each>
                      </xsl:variable>
                      <xsl:variable name="var48_result" as="xs:decimal*">
                        <xsl:for-each select="$var8_resultof_group_items/Y2">
                          <xsl:sequence select="xs:decimal(fn:string(.))"/>
                        </xsl:for-each>
                      </xsl:variable>
                      <xsl:variable name="var49_resultof_subtract" as="xs:decimal" select="(fn:max($var47_result) -
fn:min($var48_result))"/>
                      <xsl:variable name="var50_result" as="xs:decimal*">
                        <xsl:for-each select="$var8_resultof_group_items/X2">
                          <xsl:sequence select="xs:decimal(fn:string(.))"/>
                        </xsl:for-each>
                      </xsl:variable>
                      <xsl:variable name="var51_result" as="xs:decimal*">
                        <xsl:for-each select="$var8_resultof_group_items/X2">
                          <xsl:sequence select="xs:decimal(fn:string(.))"/>
                        </xsl:for-each>
                      </xsl:variable>
                      <xsl:variable name="var52_resultof_subtract" as="xs:decimal" select="(fn:max($var50_result) -
fn:min($var51_result))"/>
                      <xsl:variable name="var53_result" as="xs:decimal">
                        <xsl:choose>
                          <xsl:when test="($var52_resultof_subtract &gt;= $var49_resultof_subtract)">
                            <xsl:sequence select="$var52_resultof_subtract"/>
                          </xsl:when>
                          <xsl:otherwise>
                            <xsl:sequence select="$var49_resultof_subtract"/>
                          </xsl:otherwise>
                        </xsl:choose>
                      </xsl:variable>
                      <xsl:variable name="var54_resultof_firstCharacter" as="xs:string">
                        <xsl:call-template name="core:firstCharacter">
                          <xsl:with-param name="value" select="'.'" as="xs:string"/>
                          <xsl:with-param name="default" select="'.'" as="xs:string"/>
                        </xsl:call-template>
                      </xsl:variable>
                      <xsl:variable name="var55_resultof_firstCharacter" as="xs:string">
                        <xsl:call-template name="core:firstCharacter">
                          <xsl:with-param name="value" select="','" as="xs:string"/>
                          <xsl:with-param name="default" select="','" as="xs:string"/>
                        </xsl:call-template>
                      </xsl:variable>
                      <xsl:sequence select="fn:translate(format-number($var53_result, '#0.00#'), '.,',
fn:concat($var54_resultof_firstCharacter, $var55_resultof_firstCharacter))"/>
                    </xsl:if>
                  </xsl:if>
                </xsl:if>
              </xsl:if>
            </xsl:if>
          </xsl:variable>
          <xsl:sequence select="xs:string(xs:double($var56_result))"/>
        </gco:Distance>
      </xsl:if>
    </gmd:distance>
  </gmd:MD_Resolution>
</gn:mostDetailedViewingResolution>
<xsl:for-each-group select="$var8_resultof_group_items" group-by="grp:var59_function(.)">
  <xsl:variable name="var60_resultof_group_items" as="node()+" select="current-group()"/>
  <gn:name>
    <gn:GeographicalName>
      <gn:language>DUT</gn:language>
      <gn:nativeness><xsl:sequence select="$var10_resultof_create_attribute"/></gn:nativeness>
      <gn:nameStatus><xsl:sequence select="$var10_resultof_create_attribute"/>  </gn:nameStatus>
      <xsl:variable name="var64_result" as="xs:boolean+">
        <xsl:for-each select="$var60_resultof_group_items">
          <xsl:variable name="var63_cur" as="node()" select="."/>
          <xsl:variable name="var62_result" as="xs:boolean?">
            <xsl:for-each select="Source2">
              <xsl:variable name="var61_resultof_cast" as="xs:string" select="fn:string(.)"/>
              <xsl:sequence select="((('LD' = $var61_resultof_cast) and fn:exists($var63_cur/WBHCode2)) or (fn:not(('LD' =
$var61_resultof_cast)) and (fn:not(('WFD_BW' = $var61_resultof_cast)) or fn:exists($var63_cur/WBHCode2))))"/>
            </xsl:for-each>
          </xsl:variable>
```

```
    <xsl:sequence select="fn:exists($var62_result[.])"/>
  </xsl:for-each>
</xsl:variable>
<xsl:variable name="var68_result" as="node()+">
  <xsl:choose>
    <xsl:when test="fn:exists($var64_result[.])">
      <xsl:for-each select="$var60_resultof_group_items">
        <xsl:variable name="var66_cur" as="node()" select="."/>
        <xsl:for-each select="Source2">
          <xsl:choose>
            <xsl:when test="('LD' = fn:string(.))">
              <xsl:for-each select="$var66_cur/WBHCode2">
                <dummy>
                  <xsl:call-template name="vmf:vmf1_inputtoresult">
                    <xsl:with-param name="input" select="fn:string(.)" as="xs:string"/>
                  </xsl:call-template>
                </dummy>
              </xsl:for-each>
            </xsl:when>
            <xsl:when test="('WFD_BW' = fn:string(.))">
              <xsl:for-each select="$var66_cur/WBHCode2">
                <dummy>
                  <xsl:call-template name="vmf:vmf2_inputtoresult">
                    <xsl:with-param name="input" select="fn:string(.)" as="xs:string"/>
                  </xsl:call-template>
                </dummy>
              </xsl:for-each>
            </xsl:when>
            <xsl:otherwise>
              <xsl:variable name="var65_let" as="xs:string?">
                <xsl:for-each select="$var66_cur/WBHCode2">  <xsl:sequence select="fn:string(.)"/>  </xsl:for-each>
              </xsl:variable>
              <dummy>
                <xsl:choose>
                  <xsl:when test="fn:exists($var65_let)">
                    <xsl:sequence select="$var65_let"/>
                  </xsl:when>
                  <xsl:otherwise>  <xsl:sequence select="'other:unknown'"/>  </xsl:otherwise>
                </xsl:choose>
              </dummy>
            </xsl:otherwise>
          </xsl:choose>
        </xsl:for-each>
      </xsl:for-each>
    </xsl:when>
    <xsl:otherwise>
      <xsl:variable name="var67_resultof_create_element" as="node()">
        <dummy>   <xsl:attribute name="xsi:nil" select="'true'"/>   </dummy>
      </xsl:variable>
      <xsl:sequence select="$var67_resultof_create_element"/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:variable>
<xsl:for-each select="$var68_result">
  <gn:sourceOfName>
    <xsl:choose>
      <xsl:when test="(fn:translate(fn:string(@xsi:nil), 'true ', '1') = '1')">
        <xsl:attribute name="xsi:nil" select="'true'"/>
      </xsl:when>
      <xsl:otherwise>
        <xsl:sequence select="fn:string(.)"/>
      </xsl:otherwise>
    </xsl:choose>
  </gn:sourceOfName>
</xsl:for-each>
<gn:pronunciation>  <xsl:sequence select="$var10_resultof_create_attribute"/></gn:pronunciation>
<gn:spelling>
  <gn:SpellingOfName>
    <xsl:variable name="var69_result" as="xs:boolean+">
      <xsl:for-each select="$var60_resultof_group_items">  <xsl:sequence select="fn:exists(Naam2)"/></xsl:for-each>
    </xsl:variable>
    <xsl:variable name="var71_result" as="node()+">
      <xsl:choose>
        <xsl:when test="fn:exists($var69_result[.])">
          <xsl:for-each select="$var60_resultof_group_items/Naam2">
            <dummy>  <xsl:sequence select="fn:string(.)"/>  </dummy>
```

```
                </xsl:for-each>
              </xsl:when>
              <xsl:otherwise>
                <xsl:variable name="var70_resultof_create_element" as="node()">
                  <dummy>   <xsl:attribute name="xsi:nil" select="'true'"/>     </dummy>
                </xsl:variable>
                <xsl:sequence select="$var70_resultof_create_element"/>
              </xsl:otherwise>
            </xsl:choose>
          </xsl:variable>
          <xsl:for-each select="$var71_result">
            <gn:text>
              <xsl:if test="fn:not((fn:translate(fn:string(@xsi:nil), 'true ', '1') = '1'))">
                <xsl:sequence select="fn:string(.)"/>
              </xsl:if>
            </gn:text>
          </xsl:for-each>
          <gn:script>Latn</gn:script>
        </gn:SpellingOfName>
      </gn:spelling>
      <gn:grammaticalGender>   <xsl:sequence select="$var10_resultof_create_attribute"/>  </gn:grammaticalGender>
      <xsl:for-each select="$var60_resultof_group_items">
        <xsl:variable name="var72_cur" as="node()" select="."/>
        <xsl:for-each select="LocalID2">
          <gn:grammaticalNumber>
            <xsl:for-each select="$var72_cur/Source2">
              <xsl:attribute name="codeSpace" select="xs:string(xs:anyURI(fn:string(.)))"/>
            </xsl:for-each>
            <xsl:sequence select="fn:string(.)"/>
          </gn:grammaticalNumber>
        </xsl:for-each>
      </xsl:for-each>
    </gn:GeographicalName>
  </gn:name>
</xsl:for-each-group>
<gn:relatedSpatialObject>
  <xsl:sequence select="$var10_resultof_create_attribute"/>
</gn:relatedSpatialObject>
<gn:type>
  <xsl:sequence select="$var10_resultof_create_attribute"/>
</gn:type>
</gn:NamedPlace>
</xsl:for-each-group>
</gml:featureMembers>
</gml:FeatureCollection>
</xsl:template>
</xsl:stylesheet>
```

F.3 OML: Geographical Names

OML (part): http://gima.lekkerkerk.info/ProofOfConcept/ReferenceSet/MappingOML_HALE.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<hale-alignment>
  <cell transformation="eu.esdihumboldt.hale.align.retype">
    <source>
      <class>
        <type>Row</type>
        <filter>CQL:Source1 &lt;&gt; 'WFD_BW' AND Source1 &lt;&gt; 'LD'</filter>
      </class>
    </source>
    <target>
      <class>  <type>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}NamedPlaceType</type>          </class>
    </target>
  </cell>
  <cell transformation="eu.esdihumboldt.hale.align.retype">
    <source>
      <class>
        <type>Row</type>
        <filter>CQL:Source1='WFD_BW'</filter>
      </class>
    </source>
    <target>
      <class><type>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}NamedPlaceType</type>          </class>
    </target>
  </cell>
  <cell transformation="eu.esdihumboldt.hale.align.retype">
    <source>
      <class>
        <type>Row</type>
        <filter>CQL:Source1='LD'</filter>
      </class>
    </source>
    <target>
      <class>   <type>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}NamedPlaceType</type>          </class>
    </target>
  </cell>
  <cell transformation="eu.esdihumboldt.hale.align.rename">
    <source>
      <property>
        <type>Row</type>
        <filter>CQL:Source1 &lt;&gt; 'WFD_BW' AND Source1 &lt;&gt; 'LD'</filter>
        <child>ID1</child>
      </property>
    </source>
    <target>
      <property>
        <type>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}NamedPlaceType</type>
        <child>{http://www.opengis.net/gml/3.2}id</child>
      </property>
    </target>
    <parameter name="structuralRename" value="false"/>
  </cell>
  <cell transformation="eu.esdihumboldt.hale.align.rename">
    <source>
      <property>
        <type>Row</type>
        <filter>CQL:Source1 &lt;&gt; 'WFD_BW' AND Source1 &lt;&gt; 'LD'</filter>
        <child>Ident1</child>
      </property>
    </source>
    <target>
      <property>
        <type>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}NamedPlaceType</type>
        <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}inspireId</child>
        <child>{urn:x-inspire:specification:gmlas:BaseTypes:3.2}Identifier</child>
        <child>{urn:x-inspire:specification:gmlas:BaseTypes:3.2}localId</child>
      </property>
    </target>
    <parameter name="structuralRename" value="false"/>
  </cell>
  <cell transformation="eu.esdihumboldt.hale.align.rename">
    <source>
```

```
        <property>
          <type>Row</type>
          <filter>CQL:Source1 &lt;&gt; 'WFD_BW' AND Source1 &lt;&gt; 'LD'</filter>
          <child>WBHCode1</child>
        </property>
      </source>
      <target>
        <property>
          <type>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}NamedPlaceType</type>
          <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}inspireId</child>
          <child>{urn:x-inspire:specification:gmlas:BaseTypes:3.2}Identifier</child>
          <child>{urn:x-inspire:specification:gmlas:BaseTypes:3.2}namespace</child>
        </property>
      </target>
      <parameter name="structuralRename" value="false"/>
    </cell>
    <cell transformation="eu.esdihumboldt.hale.align.rename">
      <source>
        <property>
          <type>Row</type>
          <filter>CQL:Source1 &lt;&gt; 'WFD_BW' AND Source1 &lt;&gt; 'LD'</filter>
          <child>Naam2</child>
        </property>
      </source>
      <target>
        <property>
          <type>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}NamedPlaceType</type>
          <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}name</child>
          <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}GeographicalName</child>
          <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}spelling</child>
          <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}SpellingOfName</child>
          <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}text</child>
        </property>
      </target>
      <parameter name="structuralRename" value="false"/>
    </cell>
    <cell transformation="eu.esdihumboldt.hale.align.rename">
      <source>
        <property>
          <type>Row</type>
          <filter>CQL:Source1 &lt;&gt; 'WFD_BW' AND Source1 &lt;&gt; 'LD'</filter>
          <child>Source2</child>
        </property>
      </source>
      <target>
        <property>
          <type>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}NamedPlaceType</type>
          <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}name</child>
          <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}GeographicalName</child>
          <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}grammaticalNumber</child>
          <child>codeSpace</child>
        </property>
      </target>
      <parameter name="structuralRename" value="false"/>
    </cell>
    <cell transformation="eu.esdihumboldt.hale.align.rename">
      <source>
        <property>
          <type>Row</type>
          <filter>CQL:Source1 &lt;&gt; 'WFD_BW' AND Source1 &lt;&gt; 'LD'</filter>
          <child>LocalID2</child>
        </property>
      </source>
      <target>
        <property>
          <type>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}NamedPlaceType</type>
          <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}name</child>
          <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}GeographicalName</child>
          <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}grammaticalNumber</child>
        </property>
      </target>
      <parameter name="structuralRename" value="false"/>
    </cell>
    <cell transformation="eu.esdihumboldt.hale.align.rename">
      <source>
        <property>
```

```
            <type>Row</type>
            <filter>CQL:Source1 &lt;&gt; 'WFD_BW' AND Source1 &lt;&gt; 'LD'</filter>
            <child>Descr2</child>
         </property>
      </source>
      <target>
         <property>
            <type>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}NamedPlaceType</type>
            <child>{http://www.opengis.net/gml/3.2}description</child>
         </property>
      </target>
      <parameter name="structuralRename" value="false"/>
   </cell>
   <cell transformation="eu.esdihumboldt.hale.align.rename">
      <source>
         <property>
            <type>Row</type>
            <filter>CQL:Source1='WFD_BW'</filter>
            <child>ID1</child>
         </property>
      </source>
      <target>
         <property>
            <type>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}NamedPlaceType</type>
            <child>{http://www.opengis.net/gml/3.2}id</child>
         </property>
      </target>
      <parameter name="structuralRename" value="false"/>
   </cell>
   <cell transformation="eu.esdihumboldt.hale.align.rename">
      <source>
         <property>
            <type>Row</type>
            <filter>CQL:Source1='WFD_BW'</filter>
            <child>Ident1</child>
         </property>
      </source>
      <target>
         <property>
            <type>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}NamedPlaceType</type>
            <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}inspireId</child>
            <child>{urn:x-inspire:specification:gmlas:BaseTypes:3.2}Identifier</child>
            <child>{urn:x-inspire:specification:gmlas:BaseTypes:3.2}localId</child>
         </property>
      </target>
      <parameter name="structuralRename" value="false"/>
   </cell>
   <cell transformation="eu.esdihumboldt.hale.align.classification">
      <source>
         <property>
            <type>Row</type>
            <filter>CQL:Source1='WFD_BW'</filter>
            <child>WBHCode1</child>
         </property>
      </source>
      <target>
         <property>
            <type>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}NamedPlaceType</type>
            <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}inspireId</child>
            <child>{urn:x-inspire:specification:gmlas:BaseTypes:3.2}Identifier</child>
            <child>{urn:x-inspire:specification:gmlas:BaseTypes:3.2}namespace</child>
         </property>
      </target>
      <parameter name="notClassifiedAction" value="source"/>
      <parameter name="classificationMapping" value="34 01"/>
   </cell>
   <cell transformation="eu.esdihumboldt.hale.align.rename">
      <source>
         <property>
            <type>Row</type>
            <filter>CQL:Source1='WFD_BW'</filter>
            <child>LocalID2</child>
         </property>
      </source>
      <target>
         <property>
```

```
            <type>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}NamedPlaceType</type>
            <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}name</child>
            <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}GeographicalName</child>
            <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}grammaticalNumber</child>
         </property>
      </target>
      <parameter name="structuralRename" value="false"/>
   </cell>
   <cell transformation="eu.esdihumboldt.hale.align.rename">
      <source>
         <property>
            <type>Row</type>
            <filter>CQL:Source1='WFD_BW'</filter>
            <child>Source2</child>
         </property>
      </source>
      <target>
         <property>
            <type>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}NamedPlaceType</type>
            <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}name</child>
            <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}GeographicalName</child>
            <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}grammaticalNumber</child>
            <child>codeSpace</child>
         </property>
      </target>
      <parameter name="structuralRename" value="false"/>
   </cell>
   <cell transformation="eu.esdihumboldt.hale.align.rename">
      <source>
         <property>
            <type>Row</type>
            <filter>CQL:Source1='WFD_BW'</filter>
            <child>Descr2</child>
         </property>
      </source>
      <target>
         <property>
            <type>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}NamedPlaceType</type>
            <child>{http://www.opengis.net/gml/3.2}description</child>
         </property>
      </target>
      <parameter name="structuralRename" value="false"/>
   </cell>
   <cell transformation="eu.esdihumboldt.hale.align.rename">
      <source>
         <property>
            <type>Row</type>
            <filter>CQL:Source1='WFD_BW'</filter>
            <child>Naam2</child>
         </property>
      </source>
      <target>
         <property>
            <type>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}NamedPlaceType</type>
            <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}name</child>
            <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}GeographicalName</child>
            <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}spelling</child>
            <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}SpellingOfName</child>
            <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}text</child>
         </property>
      </target>
      <parameter name="structuralRename" value="false"/>
   </cell>
   <cell transformation="eu.esdihumboldt.hale.align.rename">
      <source>
         <property>
            <type>Row</type>
            <filter>CQL:Source1='LD'</filter>
            <child>ID1</child>
         </property>
      </source>
      <target>
         <property>
            <type>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}NamedPlaceType</type>
            <child>{http://www.opengis.net/gml/3.2}id</child>
         </property>
```

```
    </target>
    <parameter name="structuralRename" value="false"/>
</cell>
<cell transformation="eu.esdihumboldt.hale.align.rename">
    <source>
        <property>
            <type>Row</type>
            <filter>CQL:Source1='LD'</filter>
            <child>Descr2</child>
        </property>
    </source>
    <target>
        <property>
            <type>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}NamedPlaceType</type>
            <child>{http://www.opengis.net/gml/3.2}description</child>
        </property>
    </target>
    <parameter name="structuralRename" value="false"/>
</cell>
<cell transformation="eu.esdihumboldt.hale.align.rename">
    <source>
        <property>
            <type>Row</type>
            <filter>CQL:Source1='LD'</filter>
            <child>Ident1</child>
        </property>
    </source>
    <target>
        <property>
            <type>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}NamedPlaceType</type>
            <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}inspireId</child>
            <child>{urn:x-inspire:specification:gmlas:BaseTypes:3.2}Identifier</child>
            <child>{urn:x-inspire:specification:gmlas:BaseTypes:3.2}localId</child>
        </property>
    </target>
    <parameter name="structuralRename" value="false"/>
</cell>
<cell transformation="eu.esdihumboldt.hale.align.classification">
    <source>
        <property>
            <type>Row</type>
            <filter>CQL:Source1='LD'</filter>
            <child>WBHCode1</child>
        </property>
    </source>
    <target>
        <property>
            <type>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}NamedPlaceType</type>
            <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}inspireId</child>
            <child>{urn:x-inspire:specification:gmlas:BaseTypes:3.2}Identifier</child>
            <child>{urn:x-inspire:specification:gmlas:BaseTypes:3.2}namespace</child>
        </property>
    </target>
    <parameter name="notClassifiedAction" value="null"/>
    <parameter name="classificationMapping" value="02 WF"/>
    <parameter name="classificationMapping" value="34 WN"/>
</cell>
<cell transformation="eu.esdihumboldt.hale.align.rename">
    <source>
        <property>
            <type>Row</type>
            <filter>CQL:Source1='LD'</filter>
            <child>LocalID2</child>
        </property>
    </source>
    <target>
        <property>
            <type>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}NamedPlaceType</type>
            <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}name</child>
            <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}GeographicalName</child>
            <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}grammaticalNumber</child>
        </property>
    </target>
    <parameter name="structuralRename" value="false"/>
</cell>
<cell transformation="eu.esdihumboldt.hale.align.rename">
```

```
    <source>
      <property>
        <type>Row</type>
        <filter>CQL:Source1='LD'</filter>
        <child>Source2</child>
      </property>
    </source>
    <target>
      <property>
        <type>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}NamedPlaceType</type>
        <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}name</child>
        <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}GeographicalName</child>
        <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}grammaticalNumber</child>
        <child>codeSpace</child>
      </property>
    </target>
    <parameter name="structuralRename" value="false"/>
  </cell>
  <cell transformation="eu.esdihumboldt.hale.align.rename">
    <source>
      <property>
        <type>Row</type>
        <filter>CQL:Source1='LD'</filter>
        <child>Naam2</child>
      </property>
    </source>
    <target>
      <property>
        <type>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}NamedPlaceType</type>
        <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}name</child>
        <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}GeographicalName</child>
        <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}spelling</child>
        <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}SpellingOfName</child>
        <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}text</child>
      </property>
    </target>
    <parameter name="structuralRename" value="false"/>
  </cell>
  <cell transformation="eu.esdihumboldt.cst.functions.geometric.ordinates_to_point">
    <source name="y">
      <property>
        <type>Row</type>
        <filter>CQL:Source1 &lt;&gt; 'WFD_BW' AND Source1 &lt;&gt; 'LD'</filter>
        <child>Y2</child>
      </property>
    </source>
    <source name="x">
      <property>
        <type>Row</type>
        <filter>CQL:Source1 &lt;&gt; 'WFD_BW' AND Source1 &lt;&gt; 'LD'</filter>
        <child>X2</child>
      </property>
    </source>
    <target>
      <property>
        <type>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}NamedPlaceType</type>
        <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}geometry</child>
        <child>{http://www.opengis.net/gml/3.2/AbstractGeometry}choice</child>
        <child>{http://www.opengis.net/gml/3.2}Point</child>
      </property>
    </target>
    <parameter name="referenceSystem" value="epsg:28992"/>
  </cell>
  <cell transformation="eu.esdihumboldt.cst.functions.geometric.ordinates_to_point">
    <source name="y">
      <property>
        <type>Row</type>
        <filter>CQL:Source1='LD'</filter>
        <child>Y2</child>
      </property>
    </source>
    <source name="x">
      <property>
        <type>Row</type>
        <filter>CQL:Source1='LD'</filter>
        <child>X2</child>
```

```
            </property>
        </source>
        <target>
            <property>
                <type>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}NamedPlaceType</type>
                <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}geometry</child>
                <child>{http://www.opengis.net/gml/3.2/AbstractGeometry}choice</child>
                <child>{http://www.opengis.net/gml/3.2}Point</child>
            </property>
        </target>
        <parameter name="referenceSystem" value="epsg:28992"/>
    </cell>
    <cell transformation="eu.esdihumboldt.cst.functions.geometric.ordinates_to_point">
        <source name="y">
            <property>
                <type>Row</type>
                <filter>CQL:Source1='WFD_BW'</filter>
                <child>Y2</child>
            </property>
        </source>
        <source name="x">
            <property>
                <type>Row</type>
                <filter>CQL:Source1='WFD_BW'</filter>
                <child>X2</child>
            </property>
        </source>
        <target>
            <property>
                <type>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}NamedPlaceType</type>
                <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}geometry</child>
                <child>{http://www.opengis.net/gml/3.2/AbstractGeometry}choice</child>
                <child>{http://www.opengis.net/gml/3.2}Point</child>
            </property>
        </target>
        <parameter name="referenceSystem" value="epsg:28992"/>
    </cell>
    <cell transformation="eu.esdihumboldt.hale.align.assign">
        <target>
            <property>
                <type>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}NamedPlaceType</type>
                <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}name</child>
                <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}GeographicalName</child>
                <child>{urn:x-inspire:specification:gmlas:GeographicalNames:3.0}language</child>
            </property>
        </target>
        <parameter name="value" value="Dut"/>
    </cell>
</hale-alignment>
```

Appendix G: **XSLT2: WATER DATABASE ETL**

This Appendix describes the ETL process (XSLT2 transformation) to create the harmonised Water Database as dicussed in Chapter 7. The Water Database uses the reference set as produced by the code from Appendix F as well as the monitoring program and surface water bodies from the WFD Database as described in Appendix A. The results are transformed to the WQR schema as described in Appendix B.

The full code can be found on the Internet.

XSLT2: http://gima.lekkerkerk.info/ProofOfConcept/WaterDatabase/MappingMapTowaterdatabase.xslt

HTML: http://gima.lekkerkerk.info/ProofOfConcept/WaterDatabase/waterdatabase.html

XML results: http://gima.lekkerkerk.info/ProofOfConcept/WaterDatabase/waterdatabase_RDIJ.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!--This file was generated by Altova MapForce 2011r3

YOU SHOULD NOT MODIFY THIS FILE, BECAUSE IT WILL BE
OVERWRITTEN WHEN YOU RE-RUN CODE GENERATION.-->

<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:tbf="http://www.altova.com/MapForce/UDF/tbf" xmlns:vmf="http://www.altova.com/MapForce/UDF/vmf"
xmlns:grp="http://www.altova.com/MapForce/grouping" xmlns:gco="http://www.isotc211.org/2005/gco"
xmlns:gmd="http://www.isotc211.org/2005/gmd" xmlns:ns0="http://www.opengis.net/gml/3.2"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:fn="http://www.w3.org/2005/xpath-functions" xmlns:base="urn:x-
inspire:specification:gmlas:BaseTypes:3.2" xmlns:gn="urn:x-inspire:specification:gmlas:GeographicalNames:3.0" exclude-result-
prefixes="tbf vmf grp xs fn">
 <xsl:template name="tbf:tbf1_GeometryPropertyType">  <xsl:param name="input" select="()"/>
  <xsl:for-each select="$input/node()">  <xsl:if test="fn:boolean(self::ns0:Point)">
    <xsl:element name="{node-name(.)}" namespace="{namespace-uri(.)}">
     <xsl:call-template name="tbf:tbf15_PointType"><xsl:with-param name="input" select="." as="node()"/> </xsl:call-template>
    </xsl:element>  </xsl:if>
  </xsl:for-each>
 </xsl:template>
 <xsl:template name="tbf:tbf7_CodeWithAuthorityType">  <xsl:param name="input" select="()"/>
  <xsl:for-each select="$input/@codeSpace">  <xsl:attribute name="codeSpace" select="fn:string(.)"/>  </xsl:for-each>
  <xsl:sequence select="fn:string($input)"/>
 </xsl:template>
 <xsl:template name="tbf:tbf8_CodeType">  <xsl:param name="input" select="()"/>
  <xsl:for-each select="$input/@codeSpace">  <xsl:attribute name="codeSpace" select="fn:string(.)"/>  </xsl:for-each>
  <xsl:sequence select="fn:string($input)"/>
 </xsl:template>
  <xsl:if test="fn:boolean(self::ns0:Point)">
   <xsl:element name="{node-name(.)}" namespace="{namespace-uri(.)}">
    <xsl:call-template name="tbf:tbf15_PointType"><xsl:with-param name="input" select="." as="node()"/> </xsl:call-template>
   </xsl:element>
  </xsl:if>
 </xsl:for-each>
</xsl:template>
<xsl:template name="tbf:tbf15_PointType">
 <xsl:param name="input" select="()"/>
 <xsl:for-each select="$input/@ns0:id">  <xsl:attribute name="ns0:id" select="fn:string(.)"/>  </xsl:for-each>
 <xsl:for-each select="$input/@srsName">  <xsl:attribute name="srsName" select="fn:string(.)"/>  </xsl:for-each>
 <xsl:for-each select="$input/@srsDimension">  <xsl:attribute name="srsDimension" select="fn:string(.)"/>  </xsl:for-each>
 <xsl:for-each select="$input/@axisLabels">  <xsl:attribute name="axisLabels" select="fn:string(.)"/>  </xsl:for-each>
 <xsl:for-each select="$input/@uomLabels">  <xsl:attribute name="uomLabels" select="fn:string(.)"/>  </xsl:for-each>
</xsl:template>
<xsl:template name="vmf:vmf1_inputtoresult">
 <xsl:param name="input" select="()"/>
 <xsl:choose>
  <xsl:when test="$input='VIS'"> <xsl:value-of select="'nl:aquo:krwKwaliteitsElement:code'"/>  </xsl:when>
  <xsl:when test="$input='VIS_ABUN'">  <xsl:value-of select="'nl:aquo:krwKwaliteitsElement:code'"/>  </xsl:when>
  <xsl:when test="$input='VIS_LTOB'">  <xsl:value-of select="'nl:aquo:krwKwaliteitsElement:code'"/>  </xsl:when>
  <xsl:when test="$input='VIS_SRTS'">  <xsl:value-of select="'nl:aquo:krwKwaliteitsElement:code'"/>  </xsl:when>
  <xsl:when test="$input='ZGV_AREA'">  <xsl:value-of select="'nl:aquo:krwKwaliteitsElement:code'"/>  </xsl:when>
  <xsl:when test="$input='ZGV_DSRT'">  <xsl:value-of select="'nl:aquo:krwKwaliteitsElement:code'"/>  </xsl:when>
  <xsl:when test="$input='STOFEU'">  <xsl:value-of select="'nl:aquo:krwKwaliteitsElement:code'"/>  </xsl:when>
  <xsl:when test="$input='STOFOV'">  <xsl:value-of select="'nl:aquo:krwKwaliteitsElement:code'"/>  </xsl:when>
  <xsl:when test="$input='STOFPR'">  <xsl:value-of select="'nl:aquo:krwKwaliteitsElement:code'"/>  </xsl:when>
  <xsl:when test="$input='FYTOBEN'">  <xsl:value-of select="'nl:aquo:krwKwaliteitsElement:code'"/>  </xsl:when>
```

```
    <xsl:when test="$input='KWD_AREA'">      <xsl:value-of select="'nl:aquo:krwKwaliteitsElement:code'"/>   </xsl:when>
    <xsl:when test="$input='KWD_KWAL'">      <xsl:value-of select="'nl:aquo:krwKwaliteitsElement:code'"/>   </xsl:when>
    <xsl:when test="$input='HMFREG_AAN'">     <xsl:value-of select="'nl:wfd:domgwcod:code'"/>   </xsl:when>
    <xsl:when test="$input='HMFREG_AFM'">     <xsl:value-of select="'nl:wfd:domgwcod:code'"/>   </xsl:when>
    <xsl:when test="$input='HMFGET'">      <xsl:value-of select="'nl:aquo:krwKwaliteitsElement:code'"/>   </xsl:when>
    <xsl:when test="$input='HMFGET_DZW'">     <xsl:value-of select="'nl:wfd:domgwcod:code'"/>   </xsl:when>
    <xsl:when test="$input='HMFGET_GTZ'">     <xsl:value-of select="'nl:wfd:domgwcod:code'"/>   </xsl:when>
    <xsl:when test="$input='HMFMOR_HCD'">      <xsl:value-of select="'nl:wfd:domgwcod:code'"/>   </xsl:when>
    <xsl:when test="$input='HMFMOR'">     <xsl:value-of select="'nl:aquo:krwKwaliteitsElement:code'"/>   </xsl:when>
    <xsl:when test="$input='HMFREG'">     <xsl:value-of select="'nl:aquo:krwKwaliteitsElement:code'"/>   </xsl:when>
    <xsl:when test="$input='FYTOPL'">     <xsl:value-of select="'nl:aquo:krwKwaliteitsElement:code'"/>   </xsl:when>
    <xsl:when test="$input='MAFAUNA'">     <xsl:value-of select="'nl:aquo:krwKwaliteitsElement:code'"/>   </xsl:when>
    <xsl:when test="$input='MFT_ABGV'">     <xsl:value-of select="'nl:aquo:krwKwaliteitsElement:code'"/>   </xsl:when>
    <xsl:when test="$input='MFT_SRTS'">     <xsl:value-of select="'nl:aquo:krwKwaliteitsElement:code'"/>   </xsl:when>
    <xsl:when test="$input='OVWFLORA'">     <xsl:value-of select="'nl:aquo:krwKwaliteitsElement:code'"/>   </xsl:when>
    <xsl:when test="$input='HMFMOR_OEV'">     <xsl:value-of select="'nl:wfd:domgwcod:code'"/>   </xsl:when>
    <xsl:when test="$input='HMFREG_WAM'">      <xsl:value-of select="'nl:wfd:domgwcod:code'"/>   </xsl:when>
    <xsl:otherwise>      <xsl:value-of select="'nl:aquo:grootheid:code'"/>   </xsl:otherwise>
  </xsl:choose>
</xsl:template>
<xsl:template name="vmf:vmf2_inputtoresult">
  <xsl:param name="input" select="()"/>
  <xsl:choose>
    <xsl:when test="$input='T'">     <xsl:value-of select="'T'"/>   </xsl:when>
    <xsl:when test="$input='ZICHT'">      <xsl:value-of select="'ZICHT'"/>    </xsl:when>
    <xsl:when test="$input='HH'">      <xsl:value-of select="'HH'"/>    </xsl:when>
    <xsl:when test="$input='SALNTT'">     <xsl:value-of select="'SALNTT'"/>    </xsl:when>
    <xsl:otherwise>      <xsl:value-of select="'CONCTTE'"/>   </xsl:otherwise>
  </xsl:choose>
</xsl:template>
<xsl:template name="vmf:vmf3_inputtoresult">  <xsl:param name="input" select="()"/>
  <xsl:choose>
    <xsl:when test="$input='N'">     <xsl:value-of select="'Ntot'"/>    </xsl:when>
    <xsl:when test="$input='P'">     <xsl:value-of select="'Ptot'"/>    </xsl:when>
    <xsl:when test="$input='cbedzm'">      <xsl:value-of select="'carbdzm'"/>    </xsl:when>
    <xsl:when test="$input='2Clptlidne'">     <xsl:value-of select="'2Cl4C1yAn'"/>    </xsl:when>
    <xsl:when test="$input='bisCliC3yEtr'">     <xsl:value-of select="'DClDiC3yEtr'"/>    </xsl:when>
    <xsl:when test="$input='C2ypton'">      <xsl:value-of select="'C2yprton'"/>    </xsl:when>
    <xsl:when test="$input='Clprfs'">      <xsl:value-of select="'C2yClprfs'"/>    </xsl:when>
    <xsl:when test="$input='coumfs'">      <xsl:value-of select="'cumfs'"/>   </xsl:when>
    <xsl:when test="$input='DOC'">     <xsl:value-of select="'OC'"/>    </xsl:when>
    <xsl:when test="$input='doDne'">      <xsl:value-of select="'dodne'"/>    </xsl:when>
    <xsl:when test="$input='metzCl'">      <xsl:value-of select="'mzCl'"/>    </xsl:when>
    <xsl:when test="$input='pirmfC1y'">     <xsl:value-of select="'C1yprmfs'"/>   </xsl:when>
    <xsl:when test="$input='ptonC1y'">      <xsl:value-of select="'C1yprton'"/>   </xsl:when>
    <xsl:when test="$input='sDDT4'">      <xsl:value-of select="'sDDX4'"/>   </xsl:when>
    <xsl:when test="$input='DIN'">     <xsl:value-of select="'Ntot'"/>    </xsl:when>
    <xsl:when test="$input='sDDTX'">      <xsl:value-of select="'sDDX4'"/>   </xsl:when>
    <xsl:when test="$input='ZN'">     <xsl:value-of select="'Zn'"/>    </xsl:when>
    <xsl:otherwise>      <xsl:value-of select="'NVT'"/>   </xsl:otherwise>
  </xsl:choose>
</xsl:template>
<xsl:template name="vmf:vmf4_inputtoresult">  <xsl:param name="input" select="()"/>
  <xsl:choose>
    <xsl:when test="$input='ZS'">     <xsl:value-of select="'ZS'"/>    </xsl:when>
    <xsl:otherwise>      <xsl:value-of select="'NVT'"/>   </xsl:otherwise>
  </xsl:choose>
</xsl:template>
<xsl:template name="vmf:vmf5_inputtoresult">  <xsl:param name="input" select="()"/>
  <xsl:choose>
    <xsl:when test="$input='1'">      <xsl:value-of select="'10'"/>   </xsl:when>
    <xsl:when test="$input='2'">      <xsl:value-of select="'100'"/>   </xsl:when>
    <xsl:when test="$input='3'">      <xsl:value-of select="'1000'"/>    </xsl:when>
    <xsl:when test="$input='4'">      <xsl:value-of select="'10000'"/>    </xsl:when>
    <xsl:when test="$input='5'">      <xsl:value-of select="'100000'"/>    </xsl:when>
    <xsl:when test="$input='6'">      <xsl:value-of select="'1000000'"/>    </xsl:when>
    <xsl:when test="$input='0'">      <xsl:value-of select="'1'"/>   </xsl:when>
    <xsl:otherwise>      <xsl:value-of select="'1000000'"/>    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
<xsl:output method="xml" encoding="UTF-8" indent="yes"/>
<xsl:param name="WFD_MLC2" select="'../1-PreIntegration/WFD_MLC_NL_20090930_RDIJ.xml'"/>
<xsl:param name="WFD_MLC_PAR2" select="'../1-PreIntegration/WFD_MLC_PAR_NL_20090930_RDIJ.xml'"/>
<xsl:param name="WFD_OWM_Admin2" select="'../1-PreIntegration/WFD_OWM_Admin_RDIJ.xml'"/>
<xsl:param name="WFD_OWM_SGBP2" select="'../1-PreIntegration/WFD_OWM_SGBP_RDIJ.xml'"/>
<xsl:function name="grp:var16_function">
```

```
    <xsl:param name="var15_param" as="node()"/>
    <xsl:variable name="var12_GeographicalName" as="node()?" select="$var15_param/gn:GeographicalName"/>
    <xsl:variable name="var14_result" as="xs:boolean?">
      <xsl:for-each select="$var12_GeographicalName">
        <xsl:variable name="var13_result" as="xs:boolean?">
          <xsl:for-each select="gn:grammaticalNumber">        <xsl:sequence select="fn:exists(@codeSpace)"/>        </xsl:for-each>
        </xsl:variable>
        <xsl:sequence select="fn:exists($var13_result[.])"/>
      </xsl:for-each>
    </xsl:variable>
    <xsl:choose>
      <xsl:when test="fn:exists($var14_result[.])">
        <xsl:sequence select="fn:distinct-
values(xs:string(xs:anyURI(fn:string($var12_GeographicalName/gn:grammaticalNumber/@codeSpace))))"/>
      </xsl:when>
      <xsl:otherwise>
        <xsl:sequence select="fn:distinct-values(())"/>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:function>
  <xsl:function name="grp:var30_function">  <xsl:param name="var29_param" as="node()"/>
    <xsl:for-each select="$var29_param/MLCIDENT">     <xsl:sequence select="fn:string(.)"/>    </xsl:for-each>
  </xsl:function>
  <xsl:function name="grp:var37_function">  <xsl:param name="var36_param" as="node()"/>
    <xsl:for-each select="$var36_param/MLCSOORT">     <xsl:sequence select="fn:string(.)"/>    </xsl:for-each>
  </xsl:function>
  <xsl:function name="grp:var39_function">  <xsl:param name="var38_param" as="node()"/>
    <xsl:for-each select="$var38_param/DOMGWCOD">     <xsl:sequence select="fn:string(.)"/>    </xsl:for-each>
  </xsl:function>
  <xsl:function name="grp:var56_function">  <xsl:param name="var55_param" as="node()"/>
    <xsl:for-each select="$var55_param/OWMIDENT">    <xsl:sequence select="fn:string(.)"/>    </xsl:for-each>
  </xsl:function>
  <xsl:function name="grp:var91_function">  <xsl:param name="var90_param" as="node()"/>
    <xsl:variable name="var87_GeographicalName" as="node()?" select="$var90_param/gn:GeographicalName"/>
    <xsl:variable name="var89_result" as="xs:boolean?">
      <xsl:for-each select="$var87_GeographicalName">
        <xsl:variable name="var88_result" as="xs:boolean?">
          <xsl:for-each select="gn:grammaticalNumber">        <xsl:sequence select="fn:exists(@codeSpace)"/>        </xsl:for-each>
        </xsl:variable>
        <xsl:sequence select="fn:exists($var88_result[.])"/>
      </xsl:for-each>
    </xsl:variable>
    <xsl:choose>
      <xsl:when test="fn:exists($var89_result[.])">
        <xsl:sequence select="fn:distinct-
values(xs:string(xs:anyURI(fn:string($var87_GeographicalName/gn:grammaticalNumber/@codeSpace))))"/>
      </xsl:when>
      <xsl:otherwise>
        <xsl:sequence select="fn:distinct-values(())"/>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:function>
  <xsl:function name="grp:var102_function">  <xsl:param name="var101_param" as="node()"/>
    <xsl:for-each select="$var101_param/MLCIDENT">     <xsl:sequence select="fn:string(.)"/>    </xsl:for-each>
  </xsl:function>
  <xsl:function name="grp:var105_function">  <xsl:param name="var104_param" as="node()"/>
    <xsl:for-each select="$var104_param/MLCIDENT">     <xsl:sequence select="fn:string(.)"/>    </xsl:for-each>
  </xsl:function>
  <xsl:function name="grp:var108_function">  <xsl:param name="var107_param" as="node()"/>    <xsl:for-each
select="$var107_param/MLCIDENT">
      <xsl:sequence select="fn:string(.)"/>
    </xsl:for-each>
  </xsl:function>
  <xsl:template match="/">
    <xsl:variable name="var1_FeatureCollection" as="node()?" select="ns0:FeatureCollection"/>
    <xsl:variable name="var2_Import" as="node()?" select="fn:doc($WFD_MLC_PAR2)/Import"/>
    <gml:FeatureCollection xmlns:gml="http://www.opengis.net/gml/3.2" xmlns:gco="http://www.isotc211.org/2005/gco"
xmlns:gmd="http://www.isotc211.org/2005/gmd" xmlns:gsr="http://www.isotc211.org/2005/gsr"
xmlns:gss="http://www.isotc211.org/2005/gss" xmlns:gts="http://www.isotc211.org/2005/gts"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:wdb="urn:ihw:gmlas:waterdatabase" xmlns:base="urn:x-
inspire:specification:gmlas:BaseTypes:3.2" xmlns:gn="urn:x-inspire:specification:gmlas:GeographicalNames:3.0">
      <xsl:attribute name="xsi:schemaLocation" select="'http://www.opengis.net/gml/3.2 Y:/projects/P0902-GIMA/MODULE~9/03-
SOLL/waterdatabase/waterdatabase.xsd'"/>
      <xsl:attribute name="gml:id" select="'_12345'"/>
      <gml:featureMembers>
        <xsl:for-each select="$var1_FeatureCollection/gml:featureMembers/gn:NamedPlace">
```

```
<xsl:variable name="var3_name" as="node()*" select="gn:name"/>
<xsl:variable name="var4_result" as="xs:string*">
  <xsl:for-each select="$var3_name/gn:GeographicalName/gn:grammaticalNumber/@codeSpace">
    <xsl:sequence select="xs:string(xs:anyURI(fn:string(.)))"/>
  </xsl:for-each>
</xsl:variable>
<xsl:variable name="var5_result" as="xs:boolean*">
  <xsl:for-each select="fn:distinct-values($var4_result)">            <xsl:sequence select="(. = 'WFD_SW')"/>        </xsl:for-each>
</xsl:variable>
<xsl:if test="fn:not(fn:exists($var5_result[.]))">
  <xsl:variable name="var6_val" as="item()*" select="()"/>
  <xsl:variable name="var7_beginLifespanVersion" as="node()?" select="gn:beginLifespanVersion"/>
  <xsl:variable name="var8_resultof_create_attribute" as="item()*">
    <xsl:attribute name="xsi:nil" select="'true'"/>
  </xsl:variable>
  <xsl:variable name="var9_inspireId" as="node()?" select="gn:inspireId"/>
  <xsl:variable name="var10_resultof_cast" as="xs:string" select="fn:string(@gml:id)"/>
  <wdb:EnvironmentalMonitoringFacility>
    <xsl:attribute name="gml:id" select="$var10_resultof_cast"/>
    <xsl:for-each select="$var9_inspireId">
      <wdb:inspireId>              <xsl:sequence select="(./@node(), ./node())"/>            </wdb:inspireId>
    </xsl:for-each>
    <xsl:variable name="var11_result" as="xs:string*">
      <xsl:for-each select="$var3_name/gn:GeographicalName/gn:spelling/gn:SpellingOfName/gn:text">
        <xsl:sequence select="fn:lower-case(fn:string(.))"/>
      </xsl:for-each>
    </xsl:variable>
    <xsl:for-each select="fn:distinct-values($var11_result)">
      <wdb:name>            <xsl:sequence select="."/>            </wdb:name>
    </xsl:for-each>
    <xsl:for-each select="gml:description">
      <wdb:additionalDescription>              <xsl:sequence select="fn:string(.)"/>            </wdb:additionalDescription>
    </xsl:for-each>
    <wdb:mediaMonitored>
      <xsl:attribute name="codeSpace" select="xs:string(xs:anyURI('urn:aquo:compartiment:code'))"/>
      <xsl:sequence select="'OW'"/>
    </wdb:mediaMonitored>
    <xsl:for-each select="gn:geometry">
      <wdb:geometry>
        <xsl:call-template name="tbf:tbf1_GeometryPropertyType">
          <xsl:with-param name="input" select="." as="node()"/>
        </xsl:call-template>
      </wdb:geometry>
    </xsl:for-each>
    <xsl:for-each-group select="$var3_name" group-by="grp:var16_function(.)">
      <xsl:variable name="var17_resultof_group_items" as="node()+" select="current-group()"/>
      <xsl:variable name="var18_result" as="xs:string*">
        <xsl:for-each select="$var17_resultof_group_items/gn:GeographicalName/gn:grammaticalNumber/@codeSpace">
          <xsl:sequence select="xs:string(xs:anyURI(fn:string(.)))"/>
        </xsl:for-each>
      </xsl:variable>
      <xsl:variable name="var19_result" as="xs:boolean*">
        <xsl:for-each select="fn:distinct-values($var18_result)">
          <xsl:sequence select="(((. = 'WFD_BW') or (. = 'BULK')) or (. = 'WFD_SW'))"/>
        </xsl:for-each>
      </xsl:variable>
      <xsl:if test="fn:exists($var19_result[.])">
        <wdb:legalBackground>
          <xsl:attribute name="xlink:type" select="'simple'"/>
          <xsl:variable name="var20_result" as="xs:string*">
            <xsl:for-each select="$var17_resultof_group_items/gn:GeographicalName/gn:grammaticalNumber/@codeSpace">
              <xsl:sequence select="xs:string(xs:anyURI(fn:string(.)))"/>
            </xsl:for-each>
          </xsl:variable>
          <xsl:for-each select="fn:distinct-values($var20_result)">
            <xsl:variable name="var21_resultof_equal" as="xs:boolean" select="(. = 'WFD_BW')"/>
            <xsl:if test="($var21_resultof_equal or ((. = 'BULK') or (. = 'WFD_SW')))">
              <xsl:variable name="var22_result" as="xs:string">
                <xsl:choose>
                  <xsl:when test="$var21_resultof_equal">    <xsl:sequence select="'BW'"/>      </xsl:when>
                  <xsl:when test="(. = 'BULK')"> <xsl:sequence select="'LEW'"/>                  </xsl:when>
                  <xsl:otherwise> <xsl:if test="(. = 'WFD_SW')"> <xsl:sequence select="'WFD'"/> </xsl:if>  </xsl:otherwise>
                </xsl:choose>
              </xsl:variable>
              <xsl:attribute name="xlink:href" select="xs:string(xs:anyURI($var22_result))"/>
            </xsl:if>
```

```
            </xsl:for-each>
          </wdb:legalBackground>
        </xsl:if>
      </xsl:for-each-group>
      <xsl:for-each select="$var9_inspireId/base:Identifier/base:namespace">
        <wdb:responsibleParty>
          <xsl:attribute name="codeSpace" select="xs:string(xs:anyURI('urn:aquo:waterbeheerder:code'))"/>
          <xsl:sequence select="fn:string(.)"/>
        </wdb:responsibleParty>
      </xsl:for-each>
      <wdb:onlineResource>          <xsl:sequence select="$var8_resultof_create_attribute"/>          </wdb:onlineResource>
      <wdb:purpose>          <xsl:sequence select="$var8_resultof_create_attribute"/>          </wdb:purpose>
      <wdb:reportedTo>          <xsl:sequence select="$var8_resultof_create_attribute"/>          </wdb:reportedTo>
      <wdb:relatedObservation>          <xsl:sequence select="$var6_val"/>          </wdb:relatedObservation>
      <wdb:sampledFeature>          <xsl:sequence select="$var8_resultof_create_attribute"/>          </wdb:sampledFeature>
      <xsl:for-each select="gn:mostDetailedViewingResolution">
        <xsl:variable name="var23_MDResolution" as="node()?" select="gmd:MD_Resolution"/>
        <xsl:variable name="var26_result" as="xs:boolean?">
          <xsl:for-each select="$var23_MDResolution">
            <xsl:variable name="var25_result" as="xs:boolean?">
              <xsl:for-each select="gmd:distance">
                <xsl:variable name="var24_nilReason" as="node()?" select="@gco:nilReason"/>
                <xsl:sequence select="(fn:exists($var24_nilReason) and (fn:string($var24_nilReason) = 'other:undetermined'))"/>
              </xsl:for-each>
            </xsl:variable>
            <xsl:sequence select="fn:exists($var25_result[.])"/>
          </xsl:for-each>
        </xsl:variable>
        <xsl:if test="fn:not(fn:exists($var26_result[.]))">
          <wdb:positionalAccuracy>
            <xsl:attribute name="xsi:type" select="xs:QName('gmd:DQ_RelativeInternalPositionalAccuracy_Type')"/>
            <gmd:result>
              <gmd:DQ_QuantitativeResult>
                <gmd:valueUnit>
                  <xsl:attribute name="xlink:type" select="'simple'"/>
                  <xsl:for-each select="$var23_MDResolution/gmd:distance/gco:Distance">
                    <xsl:attribute name="xlink:href" select="xs:string(xs:anyURI(fn:string(@uom)))"/>
                  </xsl:for-each>
                </gmd:valueUnit>
                <gmd:value>
                  <gco:Record>
                    <xsl:for-each select="$var23_MDResolution/gmd:distance/gco:Distance">
                      <xsl:sequence select="xs:string(xs:double(fn:string(.)))"/>
                    </xsl:for-each>
                  </gco:Record>
                </gmd:value>
              </gmd:DQ_QuantitativeResult>
            </gmd:result>
          </wdb:positionalAccuracy>
        </xsl:if>
      </xsl:for-each>
      <wdb:hostedProcedure>          <xsl:sequence select="$var6_val"/>          </wdb:hostedProcedure>
      <wdb:measurementRegime>demand driven data collection</wdb:measurementRegime>
      <wdb:mobile>          <xsl:sequence select="xs:string(fn:false())"/>          </wdb:mobile>
      <wdb:resultAcquisitionSource>in-situ</wdb:resultAcquisitionSource>
      <wdb:speciallisedEMFType>  <xsl:sequence select="$var8_resultof_create_attribute"/>    </wdb:speciallisedEMFType>
      <xsl:variable name="var27_result" as="xs:boolean?">
        <xsl:for-each select="$var7_beginLifespanVersion">
          <xsl:sequence select="(fn:not((fn:translate(fn:string(@xsi:nil), 'true ', '1') = '1')) and (xs:dateTime(fn:string(.)) =
xs:dateTime('9999-01-01T00:00:00')))"/>
        </xsl:for-each>
      </xsl:variable>
      <xsl:if test="fn:exists($var27_result[.])">
        <wdb:operationalActivityPeriod>
          <xsl:attribute name="nilReason" select="'other:undetermined'"/>
        </wdb:operationalActivityPeriod>
      </xsl:if>
      <xsl:variable name="var28_result" as="xs:boolean?">
        <xsl:for-each select="$var7_beginLifespanVersion">
          <xsl:sequence select="(fn:not((fn:translate(fn:string(@xsi:nil), 'true ', '1') = '1')) and (xs:dateTime(fn:string(.)) =
xs:dateTime('9999-01-01T00:00:00')))"/>
        </xsl:for-each>
      </xsl:variable>
      <xsl:if test="fn:not(fn:exists($var28_result[.]))">
        <wdb:operationalActivityPeriod>
          <wdb:OperationalActivityPeriod>
```

```
            <wdb:activityTime>
              <xsl:attribute name="xsi:type" select="xs:QName('gml:TimePeriodType')"/>
              <xsl:attribute name="gml:id" select="fn:concat($var10_resultof_cast, '_oat')"/>
              <xsl:for-each select="$var7_beginLifespanVersion">
                <gml:beginPosition>
                  <xsl:if test="fn:not((fn:translate(fn:string(@xsi:nil), 'true ', '1') = '1'))">
                    <xsl:sequence select="xs:string(xs:dateTime(fn:string(.)))"/>
                  </xsl:if>
                </gml:beginPosition>
              </xsl:for-each>
              <xsl:for-each select="gn:endLifespanVersion">
                <gml:endPosition>
                  <xsl:if test="fn:not((fn:translate(fn:string(@xsi:nil), 'true ', '1') = '1'))">
                    <xsl:sequence select="xs:string(xs:dateTime(fn:string(.)))"/>
                  </xsl:if>
                </gml:endPosition>
              </xsl:for-each>
            </wdb:activityTime>
          </wdb:OperationalActivityPeriod>
        </wdb:operationalActivityPeriod>
      </xsl:if>
    </wdb:EnvironmentalMonitoringFacility>
  </xsl:if>
</xsl:for-each>
<xsl:for-each-group select="$var2_Import/Row" group-by="grp:var30_function(.)">
  <xsl:variable name="var31_resultof_concat" as="xs:string" select="fn:concat(current-grouping-key(), '_oc')"/>
  <xsl:variable name="var32_resultof_group_items" as="node()+" select="current-group()"/>
  <xsl:variable name="var33_resultof_create_attribute" as="item()*">
    <xsl:attribute name="xsi:nil" select="'true'"/>
  </xsl:variable>
  <xsl:variable name="var34_Import" as="node()?" select="fn:doc($WFD_MLC2)/Import"/>
  <wdb:ObservingCapability>
    <xsl:for-each select="$var34_Import/Row/MLCIDENT">
      <xsl:variable name="var35_cur" as="node()" select="."/>
      <xsl:for-each select="$var32_resultof_group_items/MLCIDENT[(fn:string($var35_cur) = fn:string(.))]">
        <xsl:attribute name="gml:id" select="$var31_resultof_concat"/>
      </xsl:for-each>
    </xsl:for-each>
    <wdb:observingTime>
      <xsl:attribute name="xsi:type" select="xs:QName('gml:TimeInstantType')"/>
      <xsl:attribute name="gml:id" select="fn:concat($var31_resultof_concat, '_bt')"/>
      <gml:timePosition>
        <xsl:sequence select="xs:string(xs:dateTime('9999-01-01T00:00:00'))"/>
      </gml:timePosition>
    </wdb:observingTime>
    <xsl:for-each-group select="$var32_resultof_group_items" group-by="grp:var37_function(.)">
      <xsl:variable name="var51_resultof_grouping_key" as="xs:string" select="current-grouping-key()"/>
      <xsl:for-each-group select="current-group()" group-by="grp:var39_function(.)">
        <xsl:variable name="var40_resultof_grouping_key" as="xs:string" select="current-grouping-key()"/>
        <xsl:variable name="var41_resultof_group_items" as="node()+" select="current-group()"/>
        <xsl:variable name="var42_resultof_vmf__inputtoresult" as="xs:string">
          <xsl:call-template name="vmf:vmf1_inputtoresult">
            <xsl:with-param name="input" select="$var40_resultof_grouping_key" as="xs:string"/>
          </xsl:call-template>
        </xsl:variable>
        <xsl:variable name="var43_val" as="xs:string">
          <xsl:choose>
            <xsl:when test="('urn:aquo:grootheid:code' = $var42_resultof_vmf__inputtoresult)">
              <xsl:call-template name="vmf:vmf2_inputtoresult">
                <xsl:with-param name="input" select="$var40_resultof_grouping_key" as="xs:string"/>
              </xsl:call-template>
            </xsl:when>
            <xsl:otherwise>                <xsl:sequence select="$var40_resultof_grouping_key"/>                </xsl:otherwise>
          </xsl:choose>
        </xsl:variable>
        <xsl:variable name="var44_resultof_equal" as="xs:boolean" select="($var43_val = 'CONCTTE')"/>
        <wdb:monitoredParameter>
          <wdb:quantity>
            <xsl:attribute name="codeSpace" select="xs:string(xs:anyURI($var42_resultof_vmf__inputtoresult))"/>
            <xsl:sequence select="$var43_val"/>
          </wdb:quantity>
          <xsl:variable name="var50_result" as="xs:boolean">
            <xsl:choose>
              <xsl:when test="$var44_resultof_equal">                <xsl:sequence select="fn:true()"/>                </xsl:when>
              <xsl:otherwise>                <xsl:variable name="var45_resultof_vmf__inputtoresult" as="xs:string">
                <xsl:call-template name="vmf:vmf4_inputtoresult">
```

```
            <xsl:with-param name="input" select="$var40_resultof_grouping_key" as="xs:string"/>
          </xsl:call-template>
        </xsl:variable>
        <xsl:sequence select="fn:not(('NVT' = $var45_resultof_vmf__inputtoresult))"/>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:variable>
  <xsl:if test="$var50_result">
    <wdb:parameter>
      <xsl:variable name="var47_result" as="xs:string">
        <xsl:choose>
          <xsl:when test="$var44_resultof_equal"> <xsl:sequence select="'urn:aquo:chemischeStof:code'"/> </xsl:when>
          <xsl:otherwise>
            <xsl:variable name="var46_resultof_vmf__inputtoresult" as="xs:string">
              <xsl:call-template name="vmf:vmf4_inputtoresult">
                <xsl:with-param name="input" select="$var40_resultof_grouping_key" as="xs:string"/>
              </xsl:call-template>
            </xsl:variable>
            <xsl:if test="fn:not(('NVT' = $var46_resultof_vmf__inputtoresult))">
              <xsl:sequence select="'urn:aquo:object:code'"/>
            </xsl:if>
          </xsl:otherwise>
        </xsl:choose>
      </xsl:variable>
      <xsl:attribute name="codeSpace" select="xs:string(xs:anyURI($var47_result))"/>
      <xsl:choose>
        <xsl:when test="$var44_resultof_equal">
          <xsl:variable name="var48_resultof_vmf__inputtoresult" as="xs:string">
            <xsl:call-template name="vmf:vmf3_inputtoresult">
              <xsl:with-param name="input" select="$var40_resultof_grouping_key" as="xs:string"/>
            </xsl:call-template>
          </xsl:variable>
          <xsl:choose>
            <xsl:when test="($var48_resultof_vmf__inputtoresult = 'NVT')">
              <xsl:sequence select="$var40_resultof_grouping_key"/>
            </xsl:when>
            <xsl:otherwise>
              <xsl:sequence select="$var48_resultof_vmf__inputtoresult"/>
            </xsl:otherwise>
          </xsl:choose>
        </xsl:when>
        <xsl:otherwise>
          <xsl:variable name="var49_resultof_vmf__inputtoresult" as="xs:string">
            <xsl:call-template name="vmf:vmf4_inputtoresult">
              <xsl:with-param name="input" select="$var40_resultof_grouping_key" as="xs:string"/>
            </xsl:call-template>
          </xsl:variable>
          <xsl:if test="fn:not(('NVT' = $var49_resultof_vmf__inputtoresult))">
            <xsl:sequence select="$var49_resultof_vmf__inputtoresult"/>
          </xsl:if>
        </xsl:otherwise>
      </xsl:choose>
    </wdb:parameter>
  </xsl:if>
  <wdb:resultNature>processed</wdb:resultNature>
  <xsl:for-each select="$var41_resultof_group_items/MONFREQ">
    <wdb:frequency>
      <xsl:attribute name="uom" select="'urn:aquo:eenheid:n/a'"/>
      <xsl:sequence select="xs:string(xs:double(xs:integer(fn:string(.))))"/>
    </wdb:frequency>
  </xsl:for-each>
  <xsl:for-each select="$var41_resultof_group_items/MONCYCLUS">
    <wdb:cycle>
      <xsl:attribute name="uom" select="'urn:aquo:eenheid:a'"/>
      <xsl:sequence select="xs:string(xs:double(xs:integer(fn:string(.))))"/>
    </wdb:cycle>
  </xsl:for-each>
  <wdb:parameterUse>
    <xsl:attribute name="codeSpace" select="xs:string(xs:anyURI('urn:aquo:krwSoortDoelMeetlocatie:code'))"/>
    <xsl:sequence select="$var51_resultof_grouping_key"/>
  </wdb:parameterUse>
 </wdb:monitoredParameter>
</xsl:for-each-group>
</xsl:for-each-group>
<wdb:onlineResource>          <xsl:sequence select="$var33_resultof_create_attribute"/>        </wdb:onlineResource>
<wdb:procedure>          <xsl:sequence select="$var33_resultof_create_attribute"/>        </wdb:procedure>
```

```
        <wdb:featureOfInterest>
          <xsl:attribute name="xlink:type" select="'simple'"/>
          <xsl:for-each select="$var34_Import/Row">
            <xsl:variable name="var54_cur" as="node()" select="."/>
            <xsl:for-each select="MLCIDENT">
              <xsl:variable name="var53_cur" as="node()" select="."/>
              <xsl:for-each select="$var32_resultof_group_items/MLCIDENT[((fn:string($var53_cur) = fn:string(.)) and
fn:exists($var54_cur/OWMIDENT))]">
                <xsl:variable name="var52_result" as="xs:string">
                  <xsl:if test="(fn:string($var53_cur) = fn:string(.))">
                    <xsl:for-each select="$var54_cur/OWMIDENT">
                      <xsl:sequence select="fn:concat('#', fn:string(.))"/>
                    </xsl:for-each>
                  </xsl:if>
                </xsl:variable>
                <xsl:attribute name="xlink:href" select="xs:string(xs:anyURI($var52_result))"/>
              </xsl:for-each>
            </xsl:for-each>
          </xsl:for-each>
        </wdb:featureOfInterest>
      </wdb:ObservingCapability>
    </xsl:for-each-group>
    <xsl:for-each-group select="fn:doc($WFD_OWM_Admin2)/Import/Row" group-by="grp:var56_function(.)">
      <xsl:variable name="var57_resultof_grouping_key" as="xs:string" select="current-grouping-key()"/>
      <xsl:variable name="var58_val" as="item()*" select="()"/>
      <xsl:variable name="var59_resultof_group_items" as="node()+" select="current-group()"/>
      <xsl:variable name="var60_resultof_substring" as="xs:string" select="fn:substring($var57_resultof_grouping_key,
xs:double(xs:decimal('3')), xs:double(xs:decimal('2')))"/>
      <xsl:variable name="var61_resultof_create_attribute" as="item()*">
        <xsl:attribute name="xsi:nil" select="'true'"/>
      </xsl:variable>
      <xsl:variable name="var62_resultof_distinct_values" as="xs:string*" select="fn:distinct-values($var60_resultof_substring)"/>
      <xsl:variable name="var63_Import" as="node()?" select="fn:doc($WFD_OWM_SGBP2)/Import"/>
      <wdb:WFDSurfaceWaterBody>
        <xsl:attribute name="gml:id" select="$var57_resultof_grouping_key"/>
        <xsl:variable name="var64_result" as="xs:string*">
          <xsl:for-each select="$var59_resultof_group_items/OWMNAAM">
            <xsl:sequence select="fn:string(.)"/>
          </xsl:for-each>
        </xsl:variable>
        <wdb:geographicalName>
          <gmd:PT_FreeText>
            <gmd:textGroup>
              <xsl:for-each select="fn:distinct-values($var64_result)">
                <gmd:LocalisedCharacterString>
                  <xsl:for-each select="$var62_resultof_distinct_values">
                    <xsl:attribute name="locale" select="xs:string(xs:anyURI(.))"/>
                  </xsl:for-each>
                  <xsl:sequence select="."/>
                </gmd:LocalisedCharacterString>
              </xsl:for-each>
            </gmd:textGroup>
          </gmd:PT_FreeText>
        </wdb:geographicalName>
        <wdb:hydroId>          <xsl:sequence select="$var61_resultof_create_attribute"/>        </wdb:hydroId>
        <wdb:authority>
          <xsl:attribute name="codeSpace" select="xs:string(xs:anyURI('urn:aquo:wbh:code'))"/>
          <xsl:sequence select="$var60_resultof_substring"/>
        </wdb:authority>
        <wdb:relatedHydroObject>          <xsl:sequence select="$var58_val"/>        </wdb:relatedHydroObject>
        <wdb:beginLifespanVersion>        <xsl:sequence select="$var61_resultof_create_attribute"/></wdb:beginLifespanVersion>
        <wdb:endLifespanVersion>          <xsl:sequence select="$var61_resultof_create_attribute"/></wdb:endLifespanVersion>
        <wdb:inspireId>
          <base:Identifier>
            <xsl:for-each select="fn:distinct-values(substring($var57_resultof_grouping_key, xs:double(xs:decimal('5'))))">
              <base:localId>              <xsl:sequence select="."/>              </base:localId>
            </xsl:for-each>
            <xsl:for-each select="$var62_resultof_distinct_values">
              <base:namespace>              <xsl:sequence select="."/>              </base:namespace>
            </xsl:for-each>
          </base:Identifier>
        </wdb:inspireId>
        <wdb:geometry>          <xsl:sequence select="$var58_val"/>        </wdb:geometry>
        <xsl:variable name="var65_result" as="xs:string*">
          <xsl:for-each select="$var59_resultof_group_items/OWMSTAT">
            <xsl:sequence select="fn:string(.)"/>
```

```
 </xsl:for-each>
</xsl:variable>
<xsl:for-each select="fn:distinct-values($var65_result)">
 <wdb:artificial>  <xsl:sequence select="xs:string((. = 'K'))"/>  </wdb:artificial>
</xsl:for-each>
<xsl:variable name="var66_result" as="xs:string*">
 <xsl:for-each select="$var59_resultof_group_items/OWMSTAT">
  <xsl:sequence select="fn:string(.)"/>
 </xsl:for-each>
</xsl:variable>
<xsl:for-each select="fn:distinct-values($var66_result)">
 <wdb:heavilyModified>  <xsl:sequence select="xs:string((. = 'S'))"/>  </wdb:heavilyModified>
</xsl:for-each>
<xsl:variable name="var67_result" as="xs:string*">
 <xsl:for-each select="$var59_resultof_group_items/OWMTYPE">
  <xsl:sequence select="fn:string(.)"/>
 </xsl:for-each>
</xsl:variable>
<xsl:for-each select="fn:distinct-values($var67_result)">
 <wdb:NLTypeCurrent>
  <xsl:attribute name="codeSpace" select="xs:string(xs:anyURI('urn:aquo:krwtype:code'))"/>
  <xsl:sequence select="."/>
 </wdb:NLTypeCurrent>
</xsl:for-each>
<xsl:variable name="var68_result" as="xs:string*">
 <xsl:for-each select="$var59_resultof_group_items/OWMTYPER">
  <xsl:sequence select="fn:string(.)"/>
 </xsl:for-each>
</xsl:variable>
<xsl:for-each select="fn:distinct-values($var68_result)">
 <wdb:NLTypeReference>
  <xsl:attribute name="codeSpace" select="xs:string(xs:anyURI('urn:aquo:krwtype:code'))"/>
  <xsl:sequence select="."/>
 </wdb:NLTypeReference>
</xsl:for-each>
<xsl:variable name="var69_result" as="xs:string*">
 <xsl:for-each select="$var59_resultof_group_items/OWMTYPINT">
  <xsl:sequence select="fn:string(.)"/>
 </xsl:for-each>
</xsl:variable>
<xsl:for-each select="fn:distinct-values($var69_result)">
 <wdb:internationalType>  <xsl:sequence select="."/>  </wdb:internationalType>
</xsl:for-each>
<wdb:representativePoint>
 <gml:Point>
  <xsl:for-each select="$var63_Import/Row/OWMIDENT">
   <xsl:variable name="var70_resultof_cast" as="xs:string" select="fn:string(.)"/>
   <xsl:if test="($var57_resultof_grouping_key = $var70_resultof_cast)">
    <xsl:attribute name="gml:id" select="fn:concat($var70_resultof_cast, '.xy')"/>
   </xsl:if>
  </xsl:for-each>
  <xsl:for-each select="$var63_Import/Row">
   <xsl:variable name="var77_cur" as="node()" select="."/>
   <xsl:for-each select="OWMIDENT">
    <xsl:variable name="var71_resultof_equal" as="xs:boolean" select="($var57_resultof_grouping_key = fn:string(.))"/>
    <xsl:variable name="var76_result" as="xs:boolean">
     <xsl:choose>
      <xsl:when test="$var71_resultof_equal">
       <xsl:variable name="var72_result" as="xs:boolean?">
        <xsl:for-each select="$var77_cur/X_SGBP">
         <xsl:sequence select="fn:exists($var77_cur/Y_SGBP)"/>
        </xsl:for-each>
       </xsl:variable>
       <xsl:sequence select="fn:exists($var72_result[.])"/>
      </xsl:when>
      <xsl:otherwise>  <xsl:sequence select="fn:false()"/>  </xsl:otherwise>
     </xsl:choose>
    </xsl:variable>
    <xsl:if test="$var76_result">
     <gml:pos>
      <xsl:if test="$var71_resultof_equal">
       <xsl:for-each select="$var77_cur/X_SGBP">
        <xsl:variable name="var75_cur" as="node()" select="."/>
        <xsl:for-each select="$var77_cur/Y_SGBP">
         <xsl:variable name="var73_resultof_vmf__inputtoresult" as="xs:string">
          <xsl:call-template name="vmf:vmf5_inputtoresult">
```

```
                        <xsl:with-param name="input" select="xs:string(xs:decimal('3'))" as="xs:string"/>
                      </xsl:call-template>
                    </xsl:variable>
                    <xsl:variable name="var74_" as="xs:decimal" select="xs:decimal($var73_resultof_vmf__inputtoresult)"/>
                    <xsl:sequence select="fn:concat(fn:concat(xs:string((fn:round((xs:decimal(fn:string($var75_cur)) * $var74_)) div
$var74_)), ' '), xs:string((fn:round((xs:decimal(fn:string(.)) * $var74_)) div $var74_)))"/>
                  </xsl:for-each>
                </xsl:for-each>
              </xsl:if>
            </gml:pos>
          </xsl:if>
        </xsl:for-each>
      </xsl:for-each>
    </gml:Point>
  </wdb:representativePoint>
 </wdb:WFDSurfaceWaterBody>
</xsl:for-each-group>
<xsl:for-each select="$var1_FeatureCollection/gml:featureMembers/gn:NamedPlace">
 <xsl:variable name="var78_name" as="node()*" select="gn:name"/>
 <xsl:variable name="var79_result" as="xs:string*">
  <xsl:for-each select="$var78_name/gn:GeographicalName/gn:grammaticalNumber/@codeSpace">
   <xsl:sequence select="xs:string(xs:anyURI(fn:string(.)))"/>
  </xsl:for-each>
 </xsl:variable>
 <xsl:variable name="var80_result" as="xs:boolean*">
  <xsl:for-each select="fn:distinct-values($var79_result)"> <xsl:sequence select="(. = 'WFD_SW')"/></xsl:for-each>
 </xsl:variable>
 <xsl:if test="fn:exists($var80_result[.])">
  <xsl:variable name="var81_beginLifespanVersion" as="node()?" select="gn:beginLifespanVersion"/>
  <xsl:variable name="var82_inspireId" as="node()?" select="gn:inspireId"/>
  <xsl:variable name="var83_resultof_create_attribute" as="item()*">
   <xsl:attribute name="xsi:nil" select="'true'"/>
  </xsl:variable>
  <xsl:variable name="var84_resultof_create_attribute" as="node()">
   <xsl:attribute name="xlink:type" select="'simple'"/>
  </xsl:variable>
  <xsl:variable name="var85_resultof_cast" as="xs:string" select="fn:string(@gml:id)"/>
  <wdb:WFD_SW_MonitoringStation>
   <xsl:attribute name="gml:id" select="$var85_resultof_cast"/>
   <xsl:for-each select="$var82_inspireId">
    <wdb:inspireId>  <xsl:sequence select="(./@node(), ./node())"/>  </wdb:inspireId>
   </xsl:for-each>
   <xsl:variable name="var86_result" as="xs:string*">
    <xsl:for-each select="$var78_name/gn:GeographicalName/gn:spelling/gn:SpellingOfName/gn:text">
     <xsl:sequence select="fn:lower-case(fn:string(.))"/>
    </xsl:for-each>
   </xsl:variable>
   <xsl:for-each select="fn:distinct-values($var86_result)">
    <wdb:name>  <xsl:sequence select="."/>  </wdb:name>
   </xsl:for-each>
   <xsl:for-each select="gml:description">
    <wdb:additionalDescription>
     <xsl:sequence select="fn:string(.)"/>
    </wdb:additionalDescription>
   </xsl:for-each>
   <wdb:mediaMonitored>
    <xsl:attribute name="codeSpace" select="xs:string(xs:anyURI('urn:aquo:compartiment:code'))"/>
    <xsl:sequence select="'OW'"/>
   </wdb:mediaMonitored>
   <xsl:for-each-group select="$var78_name" group-by="grp:var91_function(.)">
    <xsl:variable name="var92_resultof_group_items" as="node()+" select="current-group()"/>
    <xsl:variable name="var93_result" as="xs:string*">
     <xsl:for-each select="$var92_resultof_group_items/gn:GeographicalName/gn:grammaticalNumber/@codeSpace">
      <xsl:sequence select="xs:string(xs:anyURI(fn:string(.)))"/>
     </xsl:for-each>
    </xsl:variable>
    <xsl:variable name="var94_result" as="xs:boolean*">
     <xsl:for-each select="fn:distinct-values($var93_result)">
      <xsl:sequence select="(((. = 'WFD_BW') or (. = 'BULK')) or (. = 'WFD_SW'))"/>
     </xsl:for-each>
    </xsl:variable>
    <xsl:if test="fn:exists($var94_result[.])">
     <wdb:legalBackground>
      <xsl:sequence select="$var84_resultof_create_attribute"/>
      <xsl:variable name="var95_result" as="xs:string*">
       <xsl:for-each select="$var92_resultof_group_items/gn:GeographicalName/gn:grammaticalNumber/@codeSpace">
```

```
                <xsl:sequence select="xs:string(xs:anyURI(fn:string(.)))"/>
              </xsl:for-each>
            </xsl:variable>
            <xsl:for-each select="fn:distinct-values($var95_result)">
              <xsl:variable name="var96_resultof_equal" as="xs:boolean" select="(. = 'WFD_BW')"/>
              <xsl:if test="($var96_resultof_equal or ((. = 'BULK') or (. = 'WFD_SW')))">
                <xsl:variable name="var97_result" as="xs:string">
                  <xsl:choose>
                    <xsl:when test="$var96_resultof_equal">                    <xsl:sequence select="'BW'"/>
</xsl:when>
                    <xsl:when test="(. = 'BULK')">                    <xsl:sequence select="'LEW'"/>                    </xsl:when>
                    <xsl:otherwise>  <xsl:if test="(. = 'WFD_SW')"><xsl:sequence select="'WFD'"/> </xsl:if>        </xsl:otherwise>
                  </xsl:choose>
                </xsl:variable>
                <xsl:attribute name="xlink:href" select="xs:string(xs:anyURI($var97_result))"/>
              </xsl:if>
            </xsl:for-each>
          </wdb:legalBackground>
        </xsl:if>
      </xsl:for-each-group>
      <xsl:for-each select="$var82_inspireId/base:Identifier/base:namespace">
        <wdb:responsibleParty>
          <xsl:attribute name="codeSpace" select="xs:string(xs:anyURI('urn:aquo:waterbeheerder:code'))"/>
          <xsl:sequence select="fn:string(.)"/>
        </wdb:responsibleParty>
      </xsl:for-each>
      <wdb:onlineResource>            <xsl:sequence select="$var83_resultof_create_attribute"/>          </wdb:onlineResource>
      <xsl:for-each select="$var78_name/gn:GeographicalName/gn:grammaticalNumber">
        <xsl:variable name="var100_cur" as="node()" select="."/>
        <xsl:if test="fn:not((fn:translate(fn:string(@xsi:nil), 'true ', '1') = '1'))">
          <xsl:for-each select="fn:doc($WFD_MLC2)/Import/Row">
            <xsl:variable name="var99_cur" as="node()" select="."/>
            <xsl:for-each select="MLCIDENT">
              <xsl:variable name="var98_resultof_equal" as="xs:boolean" select="(fn:string($var100_cur) = fn:string(.))"/>
              <xsl:if test="($var98_resultof_equal and fn:exists($var99_cur/MLCDOEL))">
                <wdb:purpose>
                  <xsl:attribute name="codeSpace" select="xs:string(xs:anyURI('urn:aquo:krwSoortDoelMeetlocatie:code'))"/>
                  <xsl:if test="$var98_resultof_equal">
                    <xsl:for-each select="$var99_cur/MLCDOEL">
                      <xsl:sequence select="fn:string(.)"/>
                    </xsl:for-each>
                  </xsl:if>
                </wdb:purpose>
              </xsl:if>
            </xsl:for-each>
          </xsl:for-each>
        </xsl:if>
      </xsl:for-each>
      <wdb:observingCapability>
        <xsl:sequence select="$var84_resultof_create_attribute"/>
        <xsl:for-each select="$var78_name/gn:GeographicalName/gn:grammaticalNumber">
          <xsl:variable name="var113_cur" as="node()" select="."/>
          <xsl:if test="fn:not((fn:translate(fn:string(@xsi:nil), 'true ', '1') = '1'))">
            <xsl:for-each select="$var2_Import/Row">
              <xsl:variable name="var112_cur" as="node()" select="."/>
              <xsl:for-each select="MLCIDENT">
                <xsl:variable name="var111_result" as="xs:boolean">
                  <xsl:choose>
                    <xsl:when test="(fn:string($var113_cur) = fn:string(.))">
                      <xsl:variable name="var103_resultof_group_by" as="item()*">
                        <xsl:for-each-group select="$var112_cur" group-by="grp:var102_function(.)">
                          <xsl:sequence select="."/>
                        </xsl:for-each-group>
                      </xsl:variable>
                      <xsl:sequence select="fn:exists($var103_resultof_group_by)"/>
                    </xsl:when>
                    <xsl:otherwise>
                      <xsl:sequence select="fn:false()"/>
                    </xsl:otherwise>
                  </xsl:choose>
                </xsl:variable>
                <xsl:if test="$var111_result">
                  <xsl:variable name="var110_result" as="xs:string">
                    <xsl:if test="(fn:string($var113_cur) = fn:string(.))">
                      <xsl:variable name="var106_resultof_group_by" as="item()*">
                        <xsl:for-each-group select="$var112_cur" group-by="grp:var105_function(.)">
```

```
                <xsl:sequence select="."/>
               </xsl:for-each-group>
              </xsl:variable>
              <xsl:if test="fn:exists($var106_resultof_group_by)">
               <xsl:variable name="var109_result" as="xs:string*">
                <xsl:for-each-group select="$var112_cur" group-by="grp:var108_function(.)">
                 <xsl:sequence select="fn:concat(fn:concat('#', current-grouping-key()), '_oc')"/>
                </xsl:for-each-group>
               </xsl:variable>
               <xsl:sequence select="xs:string(fn:string-join(for $x in $var109_result return xs:string($x), ' '))"/>
              </xsl:if>
             </xsl:if>
            </xsl:variable>
            <xsl:attribute name="xlink:href" select="xs:string(xs:anyURI($var110_result))"/>
           </xsl:if>
          </xsl:for-each>
         </xsl:for-each>
        </xsl:if>
       </xsl:for-each>
      </wdb:observingCapability>
      <wdb:reportedTo>
       <xsl:sequence select="$var83_resultof_create_attribute"/>
      </wdb:reportedTo>
      <wdb:sampledFeature>
       <xsl:sequence select="$var84_resultof_create_attribute"/>
       <xsl:for-each select="$var78_name/gn:GeographicalName/gn:grammaticalNumber">
        <xsl:variable name="var117_cur" as="node()" select="."/>
        <xsl:if test="fn:not((fn:translate(fn:string(@xsi:nil), 'true ', '1') = '1'))">
         <xsl:for-each select="fn:doc($WFD_MLC2)/Import/Row">
          <xsl:variable name="var116_cur" as="node()" select="."/>
          <xsl:for-each select="MLCIDENT">
           <xsl:variable name="var114_resultof_equal" as="xs:boolean" select="(fn:string($var117_cur) = fn:string(.))"/>
           <xsl:if test="($var114_resultof_equal and fn:exists($var116_cur/OWAIDENT))">
            <xsl:variable name="var115_result" as="xs:string">
             <xsl:if test="$var114_resultof_equal">
              <xsl:for-each select="$var116_cur/OWAIDENT">
               <xsl:sequence select="fn:string(.)"/>
              </xsl:for-each>
             </xsl:if>
            </xsl:variable>
            <xsl:attribute name="xlink:href" select="xs:string(xs:anyURI(fn:concat('#', $var115_result)))"/>
           </xsl:if>
          </xsl:for-each>
         </xsl:for-each>
        </xsl:if>
       </xsl:for-each>
      </wdb:sampledFeature>
      <xsl:for-each select="gn:mostDetailedViewingResolution">
       <xsl:variable name="var118_MDResolution" as="node()?" select="gmd:MD_Resolution"/>
       <xsl:variable name="var121_result" as="xs:boolean?">
        <xsl:for-each select="$var118_MDResolution">
         <xsl:variable name="var120_result" as="xs:boolean?">
          <xsl:for-each select="gmd:distance">
           <xsl:variable name="var119_nilReason" as="node()?" select="@gco:nilReason"/>
           <xsl:sequence select="(fn:exists($var119_nilReason) and (fn:string($var119_nilReason) = 'other:undetermined'))"/>
          </xsl:for-each>
         </xsl:variable>
         <xsl:sequence select="fn:exists($var120_result[.])"/>
        </xsl:for-each>
       </xsl:variable>
       <xsl:if test="fn:not(fn:exists($var121_result[.]))">
        <wdb:positionalAccuracy>
         <xsl:attribute name="xsi:type" select="xs:QName('gmd:DQ_RelativeInternalPositionalAccuracy_Type')"/>
         <gmd:result>
          <gmd:DQ_QuantitativeResult>
           <gmd:valueUnit>
            <xsl:sequence select="$var84_resultof_create_attribute"/>
            <xsl:for-each select="$var118_MDResolution/gmd:distance/gco:Distance">
             <xsl:attribute name="xlink:href" select="xs:string(xs:anyURI(fn:string(@uom)))"/>
            </xsl:for-each>
           </gmd:valueUnit>
           <gmd:value>
            <gco:Record>
             <xsl:for-each select="$var118_MDResolution/gmd:distance/gco:Distance">
              <xsl:sequence select="xs:string(xs:double(fn:string(.)))"/>
             </xsl:for-each>
```

```
                </gco:Record>
               </gmd:value>
              </gmd:DQ_QuantitativeResult>
             </gmd:result>
            </wdb:positionalAccuracy>
           </xsl:if>
          </xsl:for-each>
          <wdb:hostedProcedure>
           <xsl:sequence select="()"/>
          </wdb:hostedProcedure>
          <xsl:for-each select="gn:geometry/gml:Point">
           <wdb:representativePoint>
            <xsl:sequence select="@xsi:nil"/>
            <xsl:call-template name="tbf:tbf15_PointType">
             <xsl:with-param name="input" select="." as="node()"/>
            </xsl:call-template>
           </wdb:representativePoint>
          </xsl:for-each>
          <wdb:measurementRegime>demand driven data collection</wdb:measurementRegime>
          <wdb:mobile>          <xsl:sequence select="xs:string(fn:false())"/>          </wdb:mobile>
          <wdb:resultAcquisitionSource>in-situ</wdb:resultAcquisitionSource>
          <wdb:speciallisedEMFType><xsl:sequence select="$var83_resultof_create_attribute"/></wdb:speciallisedEMFType>
          <xsl:variable name="var122_result" as="xs:boolean?">
           <xsl:for-each select="$var81_beginLifespanVersion">
            <xsl:sequence select="(fn:not((fn:translate(fn:string(@xsi:nil), 'true ', '1') = '1')) and (xs:dateTime(fn:string(.)) =
xs:dateTime('9999-01-01T00:00:00')))"/>
           </xsl:for-each>
          </xsl:variable>
          <xsl:if test="fn:exists($var122_result[.])">
           <wdb:operationalActivityPeriod>
            <xsl:attribute name="nilReason" select="'other:undetermined'"/>
           </wdb:operationalActivityPeriod>
          </xsl:if>
          <xsl:variable name="var123_result" as="xs:boolean?">
           <xsl:for-each select="$var81_beginLifespanVersion">
            <xsl:sequence select="(fn:not((fn:translate(fn:string(@xsi:nil), 'true ', '1') = '1')) and (xs:dateTime(fn:string(.)) =
xs:dateTime('9999-01-01T00:00:00')))"/>
           </xsl:for-each>
          </xsl:variable>
          <xsl:if test="fn:not(fn:exists($var123_result[.]))">
           <wdb:operationalActivityPeriod>
            <wdb:OperationalActivityPeriod>
             <wdb:activityTime>
              <xsl:attribute name="xsi:type" select="xs:QName('gml:TimePeriodType')"/>
              <xsl:attribute name="gml:id" select="fn:concat($var85_resultof_cast, '_oat')"/>
              <xsl:for-each select="$var81_beginLifespanVersion">
               <gml:beginPosition>
                <xsl:if test="fn:not((fn:translate(fn:string(@xsi:nil), 'true ', '1') = '1'))">
                 <xsl:sequence select="xs:string(xs:dateTime(fn:string(.)))"/>
                </xsl:if>
               </gml:beginPosition>
              </xsl:for-each>
              <xsl:for-each select="gn:endLifespanVersion">
               <gml:endPosition>
                <xsl:if test="fn:not((fn:translate(fn:string(@xsi:nil), 'true ', '1') = '1'))">
                 <xsl:sequence select="xs:string(xs:dateTime(fn:string(.)))"/>
                </xsl:if>
               </gml:endPosition>
              </xsl:for-each>
             </wdb:activityTime>
            </wdb:OperationalActivityPeriod>
           </wdb:operationalActivityPeriod>
          </xsl:if>
          <xsl:for-each select="$var78_name/gn:GeographicalName/gn:grammaticalNumber">
           <xsl:variable name="var127_cur" as="node()" select="."/>
           <xsl:if test="fn:not((fn:translate(fn:string(@xsi:nil), 'true ', '1') = '1'))">
            <xsl:for-each select="fn:doc($WFD_MLC2)/Import/Row">
             <xsl:variable name="var126_cur" as="node()" select="."/>
             <xsl:for-each select="MLCIDENT">
              <xsl:variable name="var124_resultof_equal" as="xs:boolean" select="(fn:string($var127_cur) = fn:string(.))"/>
              <xsl:if test="($var124_resultof_equal and fn:exists($var126_cur/MPN_AANTAL))">
               <wdb:subsites>
                <xsl:if test="$var124_resultof_equal">
                 <xsl:attribute name="codeSpace" select="xs:string(xs:anyURI('urn:eu:wfd:subsites:description'))"/>
                </xsl:if>
                <xsl:if test="$var124_resultof_equal">
```

```
                    <xsl:for-each select="$var126_cur/MPN_AANTAL">
                     <xsl:variable name="var125_resultof_cast" as="xs:decimal" select="xs:integer(fn:string(.))"/>
                     <xsl:choose>
                      <xsl:when test="($var125_resultof_cast = xs:decimal('-1'))">
                       <xsl:sequence select="'transect'"/>
                      </xsl:when>
                      <xsl:when test="($var125_resultof_cast &gt; xs:decimal('1'))">
                       <xsl:sequence select="'multipoint'"/>
                      </xsl:when>
                      <xsl:otherwise>
                       <xsl:sequence select="'not applicable/none'"/>
                      </xsl:otherwise>
                     </xsl:choose>
                    </xsl:for-each>
                   </xsl:if>
                  </wdb:subsites>
                 </xsl:if>
                </xsl:for-each>
               </xsl:for-each>
              </xsl:if>
             </xsl:for-each>
             <xsl:for-each select="$var78_name/gn:GeographicalName/gn:grammaticalNumber">
              <xsl:variable name="var131_cur" as="node()" select="."/>
              <xsl:if test="fn:not((fn:translate(fn:string(@xsi:nil), 'true ', '1') = '1'))">
               <xsl:for-each select="fn:doc($WFD_MLC2)/Import/Row">
                <xsl:variable name="var130_cur" as="node()" select="."/>
                <xsl:for-each select="MLCIDENT">
                 <xsl:variable name="var128_resultof_equal" as="xs:boolean" select="(fn:string($var131_cur) = fn:string(.))"/>
                 <xsl:if test="($var128_resultof_equal and fn:exists($var130_cur/MPN_AANTAL))">
                  <xsl:variable name="var129_result" as="xs:decimal">
                   <xsl:if test="$var128_resultof_equal">
                    <xsl:for-each select="$var130_cur/MPN_AANTAL">
                     <xsl:sequence select="xs:integer(fn:string(.))"/>
                    </xsl:for-each>
                   </xsl:if>
                  </xsl:variable>
                  <wdb:numberOfPointsInSubsite>
                   <xsl:sequence select="xs:string(xs:integer($var129_result))"/>
                  </wdb:numberOfPointsInSubsite>
                 </xsl:if>
                </xsl:for-each>
               </xsl:for-each>
              </xsl:if>
             </xsl:for-each>
             <xsl:for-each select="$var78_name/gn:GeographicalName/gn:grammaticalNumber">
              <xsl:variable name="var134_cur" as="node()" select="."/>
              <xsl:if test="fn:not((fn:translate(fn:string(@xsi:nil), 'true ', '1') = '1'))">
               <xsl:for-each select="fn:doc($WFD_MLC2)/Import/Row">
                <xsl:variable name="var133_cur" as="node()" select="."/>
                <xsl:for-each select="MLCIDENT">
                 <xsl:variable name="var132_resultof_equal" as="xs:boolean" select="(fn:string($var134_cur) = fn:string(.))"/>
                 <xsl:if test="($var132_resultof_equal and fn:exists($var133_cur/MLCSOORT))">
                  <wdb:monitoringUse>
                   <xsl:if test="$var132_resultof_equal">
                    <xsl:for-each select="$var133_cur/MLCSOORT">
                     <xsl:sequence select="fn:string(.)"/>
                    </xsl:for-each>
                   </xsl:if>
                  </wdb:monitoringUse>
                 </xsl:if>
                </xsl:for-each>
               </xsl:for-each>
              </xsl:if>
             </xsl:for-each>
            </wdb:WFD_SW_MonitoringStation>
           </xsl:if>
          </xsl:for-each>
         </gml:featureMembers>
        </gml:FeatureCollection>
       </xsl:template>
      </xsl:stylesheet>
```

Appendix H: **MEDIATED SCHEMA CODE**

In this Appendix the XQuery and XSLT2 code is given for the mediated schema solution for the WQR. The three steps (mapping of query parameters, querying of the data sources and integration into the target schema) are given in separate sub appendixes. For more information on the working of the code and results obtained, see Chapter 7. The full transformation code can be found on the internet.

H.1 Map query parameters

XQuery: http://gima.lekkerkerk.info/ProofOfConcept/MediatedSchema/MappingMapToMPN_Query_Results.xq

HTML:
http://gima.lekkerkerk.info/ProofOfConcept/MediatedSchema/query_federated_waterdatabase_1_querydef.html

XML input: http://gima.lekkerkerk.info/ProofOfConcept/MediatedSchema/query_settings.xml

XML output: http://gima.lekkerkerk.info/ProofOfConcept/MediatedSchema/MPN_Query_Results.xml

```
(: This file was generated by Altova MapForce 2011r3

YOU SHOULD NOT MODIFY THIS FILE, BECAUSE IT WILL BE
OVERWRITTEN WHEN YOU RE-RUN CODE GENERATION.:)

xquery version "1.0";

declare namespace vmf = 'http://www.altova.com/MapForce/UDF/vmf';
declare namespace xs = 'http://www.w3.org/2001/XMLSchema';
declare namespace fn = 'http://www.w3.org/2005/xpath-functions';
declare variable $BULK_WNS2 as xs:string := "../05-Integration/1-PreIntegration/BULK_WNS_RDIJ.xml";
declare variable $GeographicalNames2 as xs:string := "../05-Integration/2-ReferenceSet/4-
Integration/GeographicalNames/Altova/MPN_GeographicalNames_Altova_RDIJ.xml";
declare variable $LD_PAR2 as xs:string := "../05-Integration/1-PreIntegration/LD_PAR_RDIJ.xml";
declare function vmf:vmf1_inputtoresult( $input as xs:string ) as xs:string?
{
  if ( $input = 'VOLMFTE' ) then 'stof' else
  if ( $input = 'MASSA' ) then 'stof' else
  if ( $input = 'CONCTTE' ) then 'stof' else
  if ( $input = 'MASSFTE' ) then 'stof' else
  if ( $input = 'Q' ) then 'stof' else
  if ( $input = 'AANTL' ) then 'stof' else
  if ( $input = 'VOLME' ) then 'stof' else
  if ( $input = 'AANTPVLME' ) then 'stof' else
  if ( $input = 'AANTLOPVTE' ) then 'stof' else ''
};

declare function vmf:vmf2_inputtoresult( $input as xs:string ) as xs:string?
{
  if ( $input = 'mzCl' ) then 'metzCl' else
  if ( $input = 'C1yprton' ) then 'ptonC1y' else
  if ( $input = 'terbtn' ) then 'terbtne' else
  if ( $input = 'Corg' ) then 'DOC' else
  if ( $input = 'Ntot' ) then 'N' else
  if ( $input = 'Porg' ) then 'POC' else
  if ( $input = 'C2yprton' ) then 'C2ypton' else
  if ( $input = 'sprton2' ) then 'spton2' else
  if ( $input = 'carbdzm' ) then 'cbedzm' else
  if ( $input = 'sorgSn' ) then 'sorgSn' else
  if ( $input = 'mmtn' ) then 'mtmtn' else
  if ( $input = 'etofcbSO' ) then 'etofcbsO' else
  if ( $input = 'alDcsfn' ) then 'alDcbsfn' else
  if ( $input = 'carbrn' ) then 'cbfrn' else
  if ( $input = 'carbrl' ) then 'cbrl' else
  if ( $input = 'Clxrn' ) then 'Clxrn' else
  if ( $input = 'metocbSO' ) then 'metocbsO' else
  if ( $input = 'C1yprmfs' ) then 'pirmfC1y' else
  if ( $input = 'dodne' ) then 'doDne' else
  if ( $input = 'carbtAd' ) then 'cbeAd' else
  if ( $input = 'sulcton' ) then 'sulctone' else
  if ( $input = 'carbOxn' ) then 'cbOxn' else
  if ( $input = 'Ptot' ) then 'P' else
```

```
  if ( $input = '26DCl4NO2An' ) then 'Dcrn' else
  if ( $input = 'Dffncn' ) then 'dffncn' else
  if ( $input = 'mecpp' ) then 'mecpp' else
  if ( $input = 'MICCTNS' ) then 'MCCYSTe' else  'NVT'
};

declare function vmf:vmf3_inputtoresult( $input as xs:string ) as xs:string?
{
  if ( $input = 'VOLMFTE' ) then 'stof' else
  if ( $input = 'MASSA' ) then 'stof' else
  if ( $input = 'CONCTTE' ) then 'stof' else
  if ( $input = 'MASSFTE' ) then 'stof' else
  if ( $input = 'Q' ) then 'stof' else
  if ( $input = 'AANTL' ) then 'stof' else
  if ( $input = 'VOLME' ) then 'stof' else
  if ( $input = 'AANTPVLME' ) then 'stof' else
  if ( $input = 'AANTLOPVTE' ) then 'stof' else  'grootheid'
};

declare function vmf:vmf4_inputtoresult( $input as xs:string ) as xs:string?
{
  if ( $input = 'O2' ) then 'Zuurstofverzadigingspercentage' else
  if ( $input = 'BZV' ) then 'Biochemisch zuurstofverbruik over 5 dage' else
  if ( $input = 'CZV' ) then 'Chemisch zuurstofverbruik' else
  if ( $input = 'Ptot' ) then 'Fosfor-totaal' else
  if ( $input = 'PO4' ) then 'ortho-Fosfaat' else
  if ( $input = 'Ntot' ) then 'Stikstof-totaal' else
  if ( $input = 'NKj' ) then 'Kjeldahl stikstof' else
  if ( $input = 'NH3' ) then 'Ammoniak' else
  if ( $input = 'NH4' ) then 'Ammonium' else
  if ( $input = 'NO2' ) then 'Nitriet' else
  if ( $input = 'NO3' ) then 'Nitraat' else
  if ( $input = 'sNO3NO2' ) then 'Nitriet+nitraat' else
  if ( $input = 'CHLFa' ) then 'Chlorofyl-a' else
  if ( $input = 'sFEO' ) then 'Pheo (Faeofytine)' else
  if ( $input = 'K' ) then 'Kalium' else
  if ( $input = 'Ca' ) then 'Calcium' else
  if ( $input = 'Fe' ) then 'IJzer' else
  if ( $input = 'Mg' ) then 'Magnesium' else
  if ( $input = 'Na' ) then 'Natrium' else
  if ( $input = 'Cl' ) then 'Chloride' else
  if ( $input = 'SO4' ) then 'Sulfaat' else
  if ( $input = 'CO3' ) then 'Bicarbonaat' else
  if ( $input = 'Al' ) then 'Aluminium' else
  if ( $input = 'As' ) then 'Arseen' else
  if ( $input = 'Cd' ) then 'Cadmium' else
  if ( $input = 'Cr' ) then 'Chroom' else
  if ( $input = 'Cu' ) then 'Koper' else
  if ( $input = 'Hg' ) then 'Kwik' else
  if ( $input = 'Ni' ) then 'Nikkel' else
  if ( $input = 'Pb' ) then 'Lood' else
  if ( $input = 'Zn' ) then 'Zink' else
  if ( $input = 'Si' ) then 'Silicium' else
  if ( $input = 'BaA' ) then 'BaA (Benzo(a)antraceen)' else
  if ( $input = 'BaP' ) then 'BaP (Benzo(a)pyreen)' else
  if ( $input = 'BbF' ) then 'BbF (Benzo(b)fluorantheen)' else
  if ( $input = 'BkF' ) then 'BkF (Benzo(k)fluorantheen)' else
  if ( $input = 'Chr' ) then 'Chr (Chryseen)' else
  if ( $input = 'Ant' ) then 'Ant  (Anthraceen)' else
  if ( $input = 'Fle' ) then 'Fle (Fluoreen)' else
  if ( $input = 'Flu' ) then 'Flu (Fluorantheen)' else
  if ( $input = 'Naf' ) then 'Naf  (Naftaleen)' else
  if ( $input = 'Pyr' ) then 'Pyr (Pyreen)' else
  if ( $input = 'Fen' ) then 'Fen (Fenanthreen)' else
  if ( $input = 'InP' ) then 'InP (Indeno(1,2,3-c,d)pyreen)' else
  if ( $input = 'AcNe' ) then 'AcNe (Acenafteen)' else
  if ( $input = 'AcNy' ) then 'AcNy (Acenaftyleen)' else
  if ( $input = 'BghiPe' ) then 'BghiP (Benzo(ghi)peryleen)' else
  if ( $input = 'DBahAnt' ) then 'DBahAnt (Dibenzo(a,h)antraceen)' else
  if ( $input = 'Co' ) then 'Co Cobalt' else
  if ( $input = 'Li' ) then 'Lithium' else
  if ( $input = 'Sn' ) then 'Sn (Tin)' else
  if ( $input = 'Ald' ) then 'Ald (Aldrin)' else
  if ( $input = 'aedsfn' ) then 'aEndo (alfa-Endosulfan)' else
  if ( $input = 'aHCH' ) then 'aHCH (alfa-Hexachloorcyclohexaan)' else
  if ( $input = 'atzne' ) then 'Atrazine' else
```

```
  if ( $input = 'bHCH' ) then 'bHCH (beta-Hexachloorcyclohexaan)' else
  if ( $input = 'dHCH' ) then 'dHCH (delta-Hexachloorcyclohexaan)' else
  if ( $input = 'Daznn' ) then 'Diazinon' else
  if ( $input = 'Dld' ) then 'Dld (Dieldrin)' else
  if ( $input = 'Dmtat' ) then 'Dimethoaat' else
  if ( $input = 'endn' ) then 'End (Endrin)' else
  if ( $input = 'cHCH' ) then 'cHCH (gamma-Hexachloorcyclohexaan)' else
  if ( $input = 'HpCl' ) then 'Hepta (Heptachloor)' else
  if ( $input = 'sHpCl2' ) then 'Hepo (Heptachloorepoxide)' else
  if ( $input = 'HCB' ) then 'HCB (Hexachloorbenzeen)' else
  if ( $input = 'idn' ) then 'Isd (Isodrin)' else
  if ( $input = 'malton' ) then 'Malathion' else
  if ( $input = 'tolcfsC1y' ) then 'Methyl tolclofos' else
  if ( $input = 'C1yprton' ) then 'Methylparathion' else
  if ( $input = 'propzne' ) then 'Propazine' else
  if ( $input = 'simzne' ) then 'Simazine' else
  if ( $input = '24DDD' ) then 'opDDD (2,4-dichloordifenyldichloorethaan' else
  if ( $input = '24DDE' ) then 'opDDE (2,4-dichloordifenyldichlooretheen' else
  if ( $input = '44DDD' ) then 'ppDDD(4,4-dichloordifenyldichloorethaan)' else
  if ( $input = 'sPAK6' ) then 'som PAK 6 Borneff' else
  if ( $input = 'sPAK10' ) then 'som PAK 10 (VROM/Leidraad)' else
  if ( $input = '44DDE' ) then 'ppDDE (4,4-dichloordifenyldichlooretheen' else
  if ( $input = '44DDT' ) then 'ppDDT(4,4-dichloordifenyltrichloorethaan' else
  if ( $input = 'Clfvfs' ) then 'Chloorfenvinfos' else
  if ( $input = 'DCLvs' ) then 'Dichloorvos' else
  if ( $input = 'mevfs' ) then 'Mev (Mevinfos)' else
  if ( $input = 'C2yprton' ) then 'Ethylparathion' else
  if ( $input = '24DDT' ) then 'opDDT(2,4-dichloordifenyltrichloorethaan' else
  if ( $input = 'Ag' ) then 'Ag (Zilver)' else
  if ( $input = 'Corg' ) then 'DOC opgelost organisch koolstof' else
  if ( $input = 'Ctot' ) then 'TOC totaal organisch koolstof' else
  if ( $input = 'Mn' ) then 'Mn Mangaan' else
  if ( $input = 'ZS' ) then 'Zwevende stof' else
  if ( $input = 'GR' ) then 'Gloeirest' else   'NVT'
};

declare function vmf:vmf5_inputtoresult( $input as xs:string ) as xs:string?
{
  if ( $input = 'T' ) then 'Temperatuur' else
  if ( $input = 'pH' ) then 'Zuurgraad (veldmeting)' else
  if ( $input = 'GELDHD' ) then 'Electrisch Geleidingsvermogen (veldmetin' else
  if ( $input = 'ZICHT' ) then 'Doorzicht' else
  if ( $input = 'DIEPTE' ) then 'Diepte' else
  if ( $input = 'BREEDTE' ) then 'Breedte' else
  if ( $input = 'ALKLTT' ) then 'Alkaliniteit' else   ()
};

let $var1_Settings := ./Settings
return
  <QueryResults>
    {  (
      attribute xsi:noNamespaceSchemaLocation
        {      'Y:/projects/P0902-GIMA/MODULE~9/06-ProofOfConcept/MPN_Query_Results.xsd'       },
      for $var47_cur in fn:doc($GeographicalNames2)/*:FeatureCollection[fn:namespace-uri() eq 'http://www.opengis.net/gml/3.2']
      return      for $var46_cur in $var47_cur/*:featureMembers[fn:namespace-uri() eq 'http://www.opengis.net/gml/3.2']
        return
          for $var45_cur in $var46_cur/*:NamedPlace[fn:namespace-uri() eq 'urn:x-inspire:specification:gmlas:GeographicalNames:3.0']
          return          (if (fn:exists(
              for $var23_cur in $var45_cur/*:geometry[fn:namespace-uri() eq 'urn:x-inspire:specification:gmlas:GeographicalNames:3.0']
              return              fn:exists(
                for $var22_cur in $var23_cur/*:Point[fn:namespace-uri() eq 'http://www.opengis.net/gml/3.2']
                return                fn:exists(
                  for $var21_cur in $var22_cur/*:pos[fn:namespace-uri() eq 'http://www.opengis.net/gml/3.2']
                  return    fn:exists(                for $var20_cur in $var1_Settings
                    return                  fn:exists(                  for $var19_cur in $var20_cur/Q_X
                      return                    fn:exists(                    for $var18_cur in $var20_cur/Q_Dist_From_XY
                        return                      fn:exists(                      for $var17_cur in $var20_cur/Q_Y
                          return                        fn:exists(
                            for $var16_cur in $var45_cur/*:name[fn:namespace-uri() eq 'urn:x-
inspire:specification:gmlas:GeographicalNames:3.0']
                            return                          fn:exists(
                              for $var15_cur in $var16_cur/*:GeographicalName[fn:namespace-uri() eq 'urn:x-
inspire:specification:gmlas:GeographicalNames:3.0']
                              return                            fn:exists(
                                for $var14_cur in $var15_cur/*:spelling[fn:namespace-uri() eq 'urn:x-
inspire:specification:gmlas:GeographicalNames:3.0']
```

```
                    return                    fn:exists(
                      for $var13_cur in $var14_cur/*:SpellingOfName[fn:namespace-uri() eq 'urn:x-
inspire:specification:gmlas:GeographicalNames:3.0']
                      return                    fn:exists(
                        for $var12_cur in $var13_cur/*:text[fn:namespace-uri() eq 'urn:x-
inspire:specification:gmlas:GeographicalNames:3.0']
                        return                    fn:exists(
                          for $var11_cur in $var20_cur/Q_Name
                          return                    fn:exists(
                            for $var10_cur in $var15_cur/*:grammaticalNumber[fn:namespace-uri() eq 'urn:x-
inspire:specification:gmlas:GeographicalNames:3.0']
                            return
                              (if ((fn:translate(fn:string($var10_cur/@xsi:nil), 'true ', '1') = '1')) then fn:false() else fn:exists(
                                for $var9_cur in $var20_cur/Q_ID
                                let $var2_resultof_cast := fn:string($var9_cur),
                                  $var3_resultof_cast := fn:string($var21_cur),
                                  $var4_resultof_number := xs:decimal(fn:substring-after($var3_resultof_cast, ' ')),
                                  $var5_resultof_number := xs:decimal(fn:substring-before($var3_resultof_cast, ' ')),
                                  $var6_resultof_cast := xs:integer(xs:decimal(fn:string($var18_cur))),
                                  $var7_resultof_cast := xs:integer(xs:decimal(fn:string($var17_cur))),
                                  $var8_resultof_cast := xs:integer(xs:decimal(fn:string($var19_cur)))
                                return
                                  ((((((($var5_resultof_number >= ($var8_resultof_cast - $var6_resultof_cast)) and ($var5_resultof_number
<= ($var8_resultof_cast + $var6_resultof_cast))) and (($var4_resultof_number >= ($var7_resultof_cast - $var6_resultof_cast)) and
($var4_resultof_number <= ($var7_resultof_cast + $var6_resultof_cast)))) or fn:matches(fn:string($var12_cur),
fn:string($var11_cur), 'ix')) or fn:matches(fn:string($var10_cur), $var2_resultof_cast, 'ix')) or
fn:matches(fn:string($var45_cur/@*:id[fn:namespace-uri() eq 'http://www.opengis.net/gml/3.2']), $var2_resultof_cast, 'ix'))
                                [.]) ) [.]) [.]) [.]) [.]) [.]) [.]) [.]) [.]) [.]) [.]) [.]) [.])          [.]))
          then
           let $var24_name := $var45_cur/*:name[fn:namespace-uri() eq 'urn:x-inspire:specification:gmlas:GeographicalNames:3.0']
           return
            <MPN>            {            (
              <WDB_Ident>     { fn:string($var45_cur/@*:id[fn:namespace-uri() eq 'http://www.opengis.net/gml/3.2'])} </WDB_Ident>,
              for $var29_cur in $var24_name
              return
                for $var28_cur in $var29_cur/*:GeographicalName[fn:namespace-uri() eq 'urn:x-
inspire:specification:gmlas:GeographicalNames:3.0']
                return
                  for $var27_cur in $var28_cur/*:grammaticalNumber[fn:namespace-uri() eq 'urn:x-
inspire:specification:gmlas:GeographicalNames:3.0']
                  let $var25_codeSpace := $var27_cur/@codeSpace
                  return
                    (if (fn:exists($var25_codeSpace)) then
                      let $var26_resultof_equal := (xs:string(xs:anyURI(fn:string($var25_codeSpace))) = 'BULK')
                      return
                        (if ((if ($var26_resultof_equal) then fn:not((fn:translate(fn:string($var27_cur/@xsi:nil), 'true ', '1') = '1'))
                          else fn:false())
                        ) then
                          <BULK_Ident>
                            {
                            (if ($var26_resultof_equal) then (if ((fn:translate(fn:string($var27_cur/@xsi:nil), 'true ', '1') = '1')) then ()
                              else fn:string($var27_cur)                    )
                             else ()                    )                    }
                          </BULK_Ident>
                         else ()                )
                       else ()                ),
                for $var34_cur in $var24_name
                return
                  for $var33_cur in $var34_cur/*:GeographicalName[fn:namespace-uri() eq 'urn:x-
inspire:specification:gmlas:GeographicalNames:3.0']
                  return
                    for $var32_cur in $var33_cur/*:grammaticalNumber[fn:namespace-uri() eq 'urn:x-
inspire:specification:gmlas:GeographicalNames:3.0']
                    let $var30_codeSpace := $var32_cur/@codeSpace
                    return
                      (if (fn:exists($var30_codeSpace)) then
                        let $var31_resultof_equal := (xs:string(xs:anyURI(fn:string($var30_codeSpace))) = 'WFD_SW')
                        return
                          (if ((if ($var31_resultof_equal) then fn:not((fn:translate(fn:string($var32_cur/@xsi:nil), 'true ', '1') = '1'))
                            else fn:false())
                          ) then
                            <WFD_SW_Ident>                    {
                              (if ($var31_resultof_equal) then (if ((fn:translate(fn:string($var32_cur/@xsi:nil), 'true ', '1') = '1')) then ()
                                else fn:string($var32_cur)                    )
                               else ()                    )                    }
                            </WFD_SW_Ident>
```

```
                    else ()                    )
                  else ()                ),
              for $var39_cur in $var24_name
              return
                for $var38_cur in $var39_cur/*:GeographicalName[fn:namespace-uri() eq 'urn:x-
inspire:specification:gmlas:GeographicalNames:3.0']
                return
                  for $var37_cur in $var38_cur/*:grammaticalNumber[fn:namespace-uri() eq 'urn:x-
inspire:specification:gmlas:GeographicalNames:3.0']
                  let $var35_codeSpace := $var37_cur/@codeSpace
                  return
                    (if (fn:exists($var35_codeSpace)) then
                      let $var36_resultof_equal := (xs:string(xs:anyURI(fn:string($var35_codeSpace))) = 'WFD_BW')
                      return
                        (if ((if ($var36_resultof_equal) then fn:not((fn:translate(fn:string($var37_cur/@xsi:nil), 'true ', '1') = '1'))
                          else fn:false())
                        ) then
                          <WFD_BW_Ident>                    {
                            (if ($var36_resultof_equal) then (if ((fn:translate(fn:string($var37_cur/@xsi:nil), 'true ', '1') = '1')) then ()
                              else fn:string($var37_cur)                    )
                            else ()                    )                    }
                          </WFD_BW_Ident>
                        else ()                )
                    else ()                ),
              for $var44_cur in $var24_name
              return
                for $var43_cur in $var44_cur/*:GeographicalName[fn:namespace-uri() eq 'urn:x-
inspire:specification:gmlas:GeographicalNames:3.0']
                return
                  for $var42_cur in $var43_cur/*:grammaticalNumber[fn:namespace-uri() eq 'urn:x-
inspire:specification:gmlas:GeographicalNames:3.0']
                  let $var40_codeSpace := $var42_cur/@codeSpace
                  return
                    (if (fn:exists($var40_codeSpace)) then
                      let $var41_resultof_equal := (xs:string(xs:anyURI(fn:string($var40_codeSpace))) = 'LD')
                      return
                        (if ((if ($var41_resultof_equal) then fn:not((fn:translate(fn:string($var42_cur/@xsi:nil), 'true ', '1') = '1'))
                          else fn:false())
                        ) then
                          <LD_Ident>                    {
                            (if ($var41_resultof_equal) then (if ((fn:translate(fn:string($var42_cur/@xsi:nil), 'true ', '1') = '1')) then ()
                              else fn:string($var42_cur)                    )
                            else ()                    )                    }
                          </LD_Ident>
                        else ()                )
                    else ()                )          )          }
          </MPN>
        else ()        ),
    for $var62_cur in fn:doc($BULK_WNS2)/Import
    return    for $var61_cur in $var62_cur/Row
      return      (if (fn:exists(        for $var55_cur in $var1_Settings
        return        fn:exists(        for $var54_cur in $var55_cur/Q_Grootheid
          let $var48_resultof_cast := fn:string($var54_cur),
            $var49_resultof_equal := ('stof' = vmf:vmf1_inputtoresult($var48_resultof_cast))
          return
            (if ((if ($var49_resultof_equal) then fn:exists($var55_cur/Q_Parameter)
              else fn:true())
            ) then fn:exists(
              for $var53_cur in $var61_cur/BV_MPS_DOMGWCOD
              return
                ((if ($var49_resultof_equal) then
                  for $var52_cur in $var55_cur/Q_Parameter
                  let $var50_resultof_cast := fn:string($var52_cur),
                    $var51_resultof_vmf__inputtoresult := vmf:vmf2_inputtoresult($var50_resultof_cast)
                  return
                    (if (($var51_resultof_vmf__inputtoresult = 'NVT')) then $var50_resultof_cast else $var51_resultof_vmf__inputtoresult)
                else $var48_resultof_cast)
                = fn:string($var53_cur))              [.])
            else fn:false())          [.])
        [.])) then
        <PAR>          {          (
          for $var57_cur in $var1_Settings
          return          for $var56_cur in $var57_cur/Q_Grootheid
            return          <WDB_Grootheid>          {          fn:string($var56_cur)          }  </WDB_Grootheid>,
          for $var59_cur in $var1_Settings
          return          for $var58_cur in $var59_cur/Q_Parameter
```

```
          return          <WDB_Parameter>          {          fn:string($var58_cur)          }  </WDB_Parameter>,
        for $var60_cur in $var61_cur/WNS_ID
        return          <BULK_WNS>{ xs:string(xs:integer(xs:decimal(fn:string($var60_cur))))          }  </BULK_WNS>
      )          }
    </PAR>
   else ()       ),
for $var78_cur in fn:doc($LD_PAR2)/Import
return     for $var77_cur in $var78_cur/Row
 return       (if (fn:exists(          for $var71_cur in $var1_Settings
    return          fn:exists(          for $var70_cur in $var71_cur/Q_Grootheid
     let $var63_resultof_cast := fn:string($var70_cur),
      $var64_resultof_equal := ('stof' = vmf:vmf3_inputtoresult($var63_resultof_cast))
     return
      (if ((if ($var64_resultof_equal) then fn:exists($var71_cur/Q_Parameter)
        else fn:exists(vmf:vmf5_inputtoresult($var63_resultof_cast))          )
      ) then fn:exists(
       for $var69_cur in $var77_cur/Naam
       return
        ((if ($var64_resultof_equal) then
         for $var67_cur in $var71_cur/Q_Parameter
         let $var65_resultof_cast := fn:string($var67_cur),
          $var66_resultof_vmf__inputtoresult := vmf:vmf4_inputtoresult($var65_resultof_cast)
         return
          (if (($var66_resultof_vmf__inputtoresult = 'NVT')) then $var65_resultof_cast else $var66_resultof_vmf__inputtoresult)
        else
         let $var68_resultof_vmf__inputtoresult := vmf:vmf5_inputtoresult($var63_resultof_cast)
         return
          (if (fn:exists($var68_resultof_vmf__inputtoresult)) then $var68_resultof_vmf__inputtoresult
           else ()          )          )
        = fn:string($var69_cur))          [.])
     else fn:false())          [.])
   [.]))
   then
    <PAR>          {          (
     for $var73_cur in $var1_Settings
     return          for $var72_cur in $var73_cur/Q_Grootheid
      return
       <WDB_Grootheid>          {          fn:string($var72_cur)          }          </WDB_Grootheid>,
     for $var75_cur in $var1_Settings
     return          for $var74_cur in $var75_cur/Q_Parameter
      return
       <WDB_Parameter>          {          fn:string($var74_cur)          }          </WDB_Parameter>,
     for $var76_cur in $var77_cur/Parnr
     return
      <LD_PAR>          {          xs:string(xs:integer(xs:decimal(fn:string($var76_cur))))          }  </LD_PAR>
    )          }
   </PAR>
   else ()
  ),
<TME>     {     (
  for $var80_cur in $var1_Settings
  return       for $var79_cur in $var80_cur/Q_BeginTime
   return       <beginTime>{xs:string(xs:date(fn:string($var79_cur)))          }          </beginTime>,
  for $var82_cur in $var1_Settings
  return       for $var81_cur in $var82_cur/Q_EndTime
   return     <endTime>  { xs:string(xs:date(fn:string($var81_cur)))          }          </endTime>      )     }
</TME>
)  } </QueryResults>
```

H.2 Data retrieval

Select Observations from Bulkdatabase

XQuery: http://gima.lekkerkerk.info/ProofOfConcept/MediatedSchema/MappingMapToBULK_MWA.xq

XQuery: http://gima.lekkerkerk.info/ProofOfConcept/MediatedSchema/MappingMapToBULK_WNS.xq

HTML:
http://gima.lekkerkerk.info/ProofOfConcept/MediatedSchema/query_federated_waterdatabase_2_select_BULK.html

```
(: This file was generated by Altova MapForce 2011r3

YOU SHOULD NOT MODIFY THIS FILE, BECAUSE IT WILL BE
OVERWRITTEN WHEN YOU RE-RUN CODE GENERATION. :)

xquery version "1.0";

declare namespace xs = 'http://www.w3.org/2001/XMLSchema';
declare namespace fn = 'http://www.w3.org/2005/xpath-functions';
declare variable $BULK_MPN2 as xs:string := "../05-Integration/1-PreIntegration/BULK_MPN_RDIJ.xml";
declare variable $BULK_MWA2 as xs:string := "../05-Integration/1-PreIntegration/BULK_MWA_RDIJ - detail.xml";
let $initial := .
return
 <Import>  {  (
    attribute xsi:noNamespaceSchemaLocation
     {      'Y:/projects/P0902-GIMA/MODULE~9/05-Integration/1-PreIntegration/BULK_MWA.xsd'       },
    for $var49_cur in fn:doc($BULK_MWA2)/Import
    return      for $var48_cur in $var49_cur/Row       let $var1_QueryResults := $initial/QueryResults
     return       (if (fn:exists(
        for $var21_cur in $var1_QueryResults          return          fn:exists(
         for $var20_cur in $var21_cur/PAR          return          fn:exists(
          for $var19_cur in $var20_cur/BULK_WNS            return            fn:exists(
           for $var18_cur in $var48_cur/WNS_ID            return            fn:exists(
            for $var17_cur in $var21_cur/MPN            return            fn:exists(
             for $var16_cur in $var17_cur/BULK_Ident              return              fn:exists(
              for $var15_cur in fn:doc($BULK_MPN2)/Import              return              fn:exists(
               for $var14_cur in $var15_cur/Row              return              fn:exists(
                for $var13_cur in $var14_cur/MPNIDENT              return              fn:exists(
                 for $var12_cur in $var48_cur/MPN_ID              return              fn:exists(
                  for $var11_cur in $var14_cur/MPN_ID              return              fn:exists(
                   for $var10_cur in $var48_cur/MWADTMB              return              fn:exists(
                    for $var9_cur in $var21_cur/TME              return              fn:exists(
                     for $var8_cur in $var9_cur/beginTime              return              fn:exists(
                      for $var7_cur in $var9_cur/endTime              let $var2_resultof_cast := fn:string($var10_cur),
                     $var3_resultof_substring_after := fn:substring-after($var2_resultof_cast, '-'),
                      $var4_resultof_substring_before := fn:substring-before($var2_resultof_cast, '-'),
                      $var5_resultof_substring_before := fn:substring-before($var3_resultof_substring_after, '-'),
                      $var6_resultof_concat := fn:concat(fn:concat(fn:concat(fn:concat(fn:substring-before(fn:substring-
after($var3_resultof_substring_after, '-'), ' '), '-'), (if ((fn:string-length($var5_resultof_substring_before) = xs:decimal('1'))) then
fn:concat('0', $var5_resultof_substring_before)
                       else $var5_resultof_substring_before)
                      ), '-'), (if ((fn:string-length($var4_resultof_substring_before) = xs:decimal('1'))) then fn:concat('0',
$var4_resultof_substring_before)
                       else $var4_resultof_substring_before)
                      )
                     return
                      (((((fn:string($var19_cur) = xs:string(xs:integer(xs:decimal(fn:string($var18_cur))))) and
(fn:string($var16_cur) = fn:string($var13_cur))) and (xs:integer(xs:decimal(fn:string($var12_cur))) =
xs:integer(xs:decimal(fn:string($var11_cur))))) and ($var6_resultof_concat >= xs:string(xs:date(fn:string($var8_cur))))) and
($var6_resultof_concat <= xs:string(xs:date(fn:string($var7_cur)))))
                     [.]) [.]) [.]) [.]) [.]) [.]) [.]) [.]) [.]) [.]) [.]) [.]) [.]) [.])          [.]))
       then
        let $var22_WNSID := $var48_cur/WNS_ID,$var23_MPNID := $var48_cur/MPN_ID,$var24_MWADTMB :=
           $var48_cur/MWADTMB
        return
         <Row>          {          (
           for $var25_cur in $var48_cur/MWA_ID
           return              <MWA_ID>{ xs:string(xs:integer(xs:decimal(fn:string($var25_cur))))   }</MWA_ID>,
           for $var26_cur in $var23_MPNID
           return              <MPN_ID> { xs:string(xs:integer(xs:decimal(fn:string($var26_cur))))   } </MPN_ID>,
```

```
            for $var27_cur in $var22_WNSID
            return              <WNS_ID>{xs:string(xs:integer(xs:decimal(fn:string($var27_cur))))  }</WNS_ID>,
            for $var28_cur in $var48_cur/MRSINKWA_ID
            return              <MRSINKWA_ID>{xs:string(xs:integer(xs:decimal(fn:string($var28_cur))))}</MRSINKWA_ID>,
            for $var29_cur in $var48_cur/MWAWRDEN
            return              <MWAWRDEN>{xs:string(xs:double(fn:string($var29_cur)))}</MWAWRDEN>,
            for $var30_cur in $var48_cur/MWAWRDEA
            return              <MWAWRDEA>{  fn:string($var30_cur)          }             </MWAWRDEA>,
            for $var31_cur in $var24_MWADTMB
            return              <MWADTMB>   {  fn:string($var31_cur)          }             </MWADTMB>,
            for $var32_cur in $var48_cur/MWATIJDB
            return              <MWATIJDB>   {  fn:string($var32_cur)          }             </MWATIJDB>,
            for $var47_cur in $var1_QueryResults
            return              for $var46_cur in $var47_cur/PAR                return
              for $var45_cur in $var46_cur/BULK_WNS,
                $var44_cur in $var22_WNSID,
                $var43_cur in $var47_cur/MPN,
                $var42_cur in $var43_cur/BULK_Ident,
                $var41_cur in fn:doc($BULK_MPN2)/Import,
                $var40_cur in $var41_cur/Row,
                $var39_cur in $var40_cur/MPNIDENT,
                $var38_cur in $var23_MPNID,
                $var37_cur in $var40_cur/MPN_ID,
                $var36_cur in $var24_MWADTMB,
                $var35_cur in $var47_cur/TME,
                $var34_cur in $var35_cur/beginTime,
                $var33_cur in $var35_cur/endTime
              return
                (if ((((((fn:string($var45_cur) = xs:string(xs:integer(xs:decimal(fn:string($var44_cur))))) and (fn:string($var42_cur) =
fn:string($var39_cur))) and (xs:integer(xs:decimal(fn:string($var38_cur))) = xs:integer(xs:decimal(fn:string($var37_cur))))) and
(fn:concat(fn:concat(fn:concat(fn:concat(fn:substring-before(fn:substring-after(fn:string($var36_cur), '-'), '-'), ' '), '-'),
(if ((fn:string-length(fn:substring-before(fn:substring-after(fn:string($var36_cur), '-'), '-')) = xs:decimal('1'))) then fn:concat('0',
fn:substring-before(fn:substring-after(fn:string($var36_cur), '-'), '-')) else fn:substring-before(fn:substring-after(fn:string($var36_cur),
'-'), '-'))
                ), '-'), (if ((fn:string-length(fn:substring-before(fn:string($var36_cur), '-')) = xs:decimal('1'))) then fn:concat('0',
fn:substring-before(fn:string($var36_cur), '-')) else fn:substring-before(fn:string($var36_cur), '-'))
                ) >= xs:string(xs:date(fn:string($var34_cur))))) and (fn:concat(fn:concat(fn:concat(fn:concat(fn:substring-
before(fn:substring-after(fn:substring-after(fn:string($var36_cur), '-'), '-'), ' '), '-'), (if ((fn:string-length(fn:substring-before(fn:substring-
after(fn:string($var36_cur), '-'), '-')) = xs:decimal('1'))) then fn:concat('0', fn:substring-before(fn:substring-after(fn:string($var36_cur), '-
'), '-')) else fn:substring-before(fn:substring-after(fn:string($var36_cur), '-'), '-'))
                ), '-'), (if ((fn:string-length(fn:substring-before(fn:string($var36_cur), '-')) = xs:decimal('1'))) then fn:concat('0',
fn:substring-before(fn:string($var36_cur), '-')) else fn:substring-before(fn:string($var36_cur), '-'))
                ) <= xs:string(xs:date(fn:string($var33_cur)))))) then
                <MWADTME>                   {                   fn:string($var39_cur)                   }                   </MWADTME>
              else ()               )           )           }
        </Row>
      else ()       )  )  }
</Import>
```


Select WNS from Bulkdatabase

```
(: This file was generated by Altova MapForce 2011r3

YOU SHOULD NOT MODIFY THIS FILE, BECAUSE IT WILL BE
OVERWRITTEN WHEN YOU RE-RUN CODE GENERATION.:)

xquery version "1.0";

declare namespace xs = 'http://www.w3.org/2001/XMLSchema';
declare namespace fn = 'http://www.w3.org/2005/xpath-functions';
declare variable $BULK_MWA3 as xs:string := "BULK_MWA_query.xml";
<Import>
 {
 (
  attribute xsi:noNamespaceSchemaLocation
   {
   'Y:/projects/P0902-GIMA/MODULE~9/05-Integration/1-PreIntegration/BULK_WNS.xsd'
   },
  for $var10_cur in ./Import
  return   for $var9_cur in $var10_cur/Row
   return     (if (fn:exists(       for $var4_cur in fn:doc($BULK_MWA3)/Import
      return         fn:exists(         for $var3_cur in $var4_cur/Row
       return          fn:exists(          for $var2_cur in $var3_cur/WNS_ID
        return           fn:exists(            for $var1_cur in $var9_cur/WNS_ID
         return             (xs:integer(xs:decimal(fn:string($var2_cur))) = xs:integer(xs:decimal(fn:string($var1_cur))))
```

```
            [.])          [.])          [.])        [.]))
        then
        <Row>        {        (
          for $var5_cur in $var9_cur/WNS_ID
          return          <WNS_ID>{xs:string(xs:integer(xs:decimal(fn:string($var5_cur))))      }  </WNS_ID>,
          for $var6_cur in $var9_cur/BV_MPS_DOMGWCOD
          return          <BV_MPS_DOMGWCOD>{fn:string($var6_cur)    }          </BV_MPS_DOMGWCOD>,
          for $var7_cur in $var9_cur/BV_MEP_DOMGWCOD
          return          <BV_MEP_DOMGWCOD>            {            fn:string($var7_cur) }</BV_MEP_DOMGWCOD>,
          for $var8_cur in $var9_cur/BV_HOE_DOMGWCOD
          return          <BV_HOE_DOMGWCOD>          {            fn:string($var8_cur)   }</BV_HOE_DOMGWCOD>
        )        }
        </Row>
       else ()     ) ) }
</Import>
```

Select observations from Limnodata

XQuery: http://gima.lekkerkerk.info/ProofOfConcept/MediatedSchema/MappingMapToLD_MON2.xq

XQuery: http://gima.lekkerkerk.info/ProofOfConcept/MediatedSchema/MappingMapToLD_PAR.xq

HTML:
http://gima.lekkerkerk.info/ProofOfConcept/MediatedSchema/query_federated_waterdatabase_2_select_LD.html

```
(: This file was generated by Altova MapForce 2011r3

YOU SHOULD NOT MODIFY THIS FILE, BECAUSE IT WILL BE
OVERWRITTEN WHEN YOU RE-RUN CODE GENERATION.:)

xquery version "1.0";

declare namespace xs = 'http://www.w3.org/2001/XMLSchema';
declare namespace fn = 'http://www.w3.org/2005/xpath-functions';
declare variable $LD_MON22 as xs:string := "../05-Integration/1-PreIntegration/LD_MON_RDIJ-detail.xml";
declare variable $LD_MP2 as xs:string := "../05-Integration/1-PreIntegration/LD_MP_RDIJ.xml";
let $initial := .
return
 <Import>
  {
  (
   attribute xsi:noNamespaceSchemaLocation
    {
    'Y:/projects/P0902-GIMA/MODULE~9/05-Integration/1-PreIntegration/LD_MON2.xsd'
    },
   for $var49_cur in fn:doc($LD_MON22)/Import
   return      for $var48_cur in $var49_cur/Row      let $var1_QueryResults := $initial/QueryResults
    return       (if (fn:exists(          for $var21_cur in $var1_QueryResults
       return          fn:exists(          for $var20_cur in $var21_cur/PAR
        return          fn:exists(          for $var19_cur in $var20_cur/LD_PAR
         return          fn:exists(          for $var18_cur in $var48_cur/Parnr
          return          fn:exists(          for $var17_cur in $var21_cur/MPN
           return          fn:exists(          for $var16_cur in $var17_cur/LD_Ident
            return          fn:exists(          for $var15_cur in fn:doc($LD_MP2)/Import
             return          fn:exists(          for $var14_cur in $var15_cur/Row
              return          fn:exists(          for $var13_cur in $var14_cur/Mp
               return          fn:exists(          for $var12_cur in $var14_cur/Recnum
                return          fn:exists(          for $var11_cur in $var48_cur/Mprec
                 return          fn:exists(          for $var10_cur in $var48_cur/Datum
                  return          fn:exists(          for $var9_cur in $var21_cur/TME
                   return          fn:exists(          for $var8_cur in $var9_cur/beginTime
                    return          fn:exists(          for $var7_cur in $var9_cur/endTime
                     let $var2_resultof_cast := fn:string($var10_cur),
                      $var3_resultof_substring_before := fn:substring-before($var2_resultof_cast, '-'),
                      $var4_resultof_substring_after := fn:substring-after($var2_resultof_cast, '-'),
                      $var5_resultof_substring_before := fn:substring-before($var4_resultof_substring_after, '-'),
                      $var6_resultof_concat := fn:concat(fn:concat(fn:concat(fn:concat(fn:substring-before(fn:substring-
after($var4_resultof_substring_after, '-'), ' '), '-'), (if ((fn:string-length($var5_resultof_substring_before) = xs:decimal('1'))) then
fn:concat('0', $var5_resultof_substring_before)
                                 else $var5_resultof_substring_before)
                                ), '-'), (if ((fn:string-length($var3_resultof_substring_before) = xs:decimal('1'))) then fn:concat('0',
$var3_resultof_substring_before)
```

```
                    else $var3_resultof_substring_before)
                  )
                return
                 (((((fn:string($var19_cur) = xs:string(xs:integer(xs:decimal(fn:string($var18_cur))))) and
(fn:string($var16_cur) = fn:string($var13_cur))) and (xs:integer(xs:decimal(fn:string($var12_cur))) =
xs:integer(xs:decimal(fn:string($var11_cur))))) and ($var6_resultof_concat >= xs:string(xs:date(fn:string($var8_cur))))) and
($var6_resultof_concat <= xs:string(xs:date(fn:string($var7_cur)))))
                [.]) [.]) [.]) [.]) [.]) [.]) [.]) [.]) [.]) [.]) [.]) [.]) [.]) [.])      [.]))
    then
      let $var22_Datum := $var48_cur/Datum,
       $var23_Parnr := $var48_cur/Parnr,
       $var24_Mprec := $var48_cur/Mprec
      return
       <Row>      {      (
         for $var25_cur in $var48_cur/Recnum
         return      <Recnum> {    xs:string(xs:integer(xs:decimal(fn:string($var25_cur))))    }    </Recnum>,
         for $var26_cur in $var24_Mprec
         return      <Mprec>    {    xs:string(xs:integer(xs:decimal(fn:string($var26_cur))))    }    </Mprec>,
         for $var27_cur in $var22_Datum
         return      <Datum>    {    fn:string($var27_cur)          }          </Datum>,
         for $var28_cur in $var48_cur/Tijd
         return      <Tijd>          {          fn:string($var28_cur)          }          </Tijd>,
         for $var29_cur in $var48_cur/Detec
         return      <Detec>          {          fn:string($var29_cur)          }          </Detec>,
         for $var30_cur in $var48_cur/Waarde
         return      <Waarde>          {          xs:string(xs:double(fn:string($var30_cur)))  }</Waarde>,
         for $var45_cur in $var1_QueryResults
         return      for $var44_cur in $var45_cur/PAR
          return      for $var43_cur in $var44_cur/LD_PAR,
            $var42_cur in $var23_Parnr,
            $var41_cur in $var45_cur/MPN,
            $var40_cur in $var41_cur/LD_Ident,
            $var39_cur in fn:doc($LD_MP2)/Import,
            $var38_cur in $var39_cur/Row,
            $var37_cur in $var38_cur/Mp,
            $var36_cur in $var38_cur/Recnum,
            $var35_cur in $var24_Mprec,
            $var34_cur in $var22_Datum,
            $var33_cur in $var45_cur/TME,
            $var32_cur in $var33_cur/beginTime,
            $var31_cur in $var33_cur/endTime
           return
            (if ((((((fn:string($var43_cur) = xs:string(xs:integer(xs:decimal(fn:string($var42_cur))))) and (fn:string($var40_cur) =
fn:string($var37_cur))) and (xs:integer(xs:decimal(fn:string($var36_cur))) = xs:integer(xs:decimal(fn:string($var35_cur))))) and
(fn:concat(fn:concat(fn:concat(fn:concat(fn:substring-before(fn:substring-after(fn:substring-after(fn:string($var34_cur), '-'), '-'), ' '), '-'),
(if ((fn:string-length(fn:substring-before(fn:substring-after(fn:string($var34_cur), '-'), '-')) = xs:decimal('1'))) then fn:concat('0',
fn:substring-before(fn:substring-after(fn:string($var34_cur), '-'), '-')) else fn:substring-before(fn:substring-after(fn:string($var34_cur),
'-'), '-'))
                ), '-'), (if ((fn:string-length(fn:substring-before(fn:string($var34_cur), '-')) = xs:decimal('1'))) then fn:concat('0',
fn:substring-before(fn:string($var34_cur), '-')) else fn:substring-before(fn:string($var34_cur), '-'))
                ) >= xs:string(xs:date(fn:string($var32_cur))))) and (fn:concat(fn:concat(fn:concat(fn:concat(fn:substring-
before(fn:substring-after(fn:substring-after(fn:string($var34_cur), '-'), '-'), ' '), '-'), (if ((fn:string-length(fn:substring-before(fn:substring-
after(fn:string($var34_cur), '-'), '-')) = xs:decimal('1'))) then fn:concat('0', fn:substring-before(fn:substring-after(fn:string($var34_cur), '-
'), '-')) else fn:substring-before(fn:substring-after(fn:string($var34_cur), '-'), '-'))
                ), '-'), (if ((fn:string-length(fn:substring-before(fn:string($var34_cur), '-')) = xs:decimal('1'))) then fn:concat('0',
fn:substring-before(fn:string($var34_cur), '-')) else fn:substring-before(fn:string($var34_cur), '-'))
                ) <= xs:string(xs:date(fn:string($var31_cur)))))) then
                <Lab>          {          fn:string($var37_cur)          }          </Lab>
               else ()          ),
         for $var46_cur in $var48_cur/Labmeth
         return      <Labmeth>{          fn:string($var46_cur)          }          </Labmeth>,
         for $var47_cur in $var23_Parnr
         return      <Parnr>{          xs:string(xs:integer(xs:decimal(fn:string($var47_cur))))  }  </Parnr> )          }
       </Row>
     else ()    )  )  }
 </Import>
```

Select parameters from Limnodata

```
(: This file was generated by Altova MapForce 2011r3

YOU SHOULD NOT MODIFY THIS FILE, BECAUSE IT WILL BE
OVERWRITTEN WHEN YOU RE-RUN CODE GENERATION.:)

xquery version "1.0";
```

```
declare namespace xs = 'http://www.w3.org/2001/XMLSchema';
declare namespace fn = 'http://www.w3.org/2005/xpath-functions';
declare variable $LD_MON23 as xs:string := "LD_MON2_query.xml";
<Import> { (
   attribute xsi:noNamespaceSchemaLocation
     {    'Y:/projects/P0902-GIMA/MODULE~9/05-Integration/1-PreIntegration/LD_PAR.xsd'     },
   for $var10_cur in ./Import
   return    for $var9_cur in $var10_cur/Row
    return      (if (fn:exists(        for $var4_cur in fn:doc($LD_MON23)/Import
       return         fn:exists(         for $var3_cur in $var4_cur/Row
         return           fn:exists(           for $var2_cur in $var3_cur/Parnr
           return             fn:exists(             for $var1_cur in $var9_cur/Parnr
             return               (xs:integer(xs:decimal(fn:string($var2_cur))) = xs:integer(xs:decimal(fn:string($var1_cur))))   [.]) [.]) [.]) [.]))
     then
      <Row>        {        (
         for $var5_cur in $var9_cur/Recnum
         return            <Recnum>              {              xs:string(xs:integer(xs:decimal(fn:string($var5_cur))))  }</Recnum>,
         for $var6_cur in $var9_cur/Parnr
         return            <Parnr>              {              xs:string(xs:integer(xs:decimal(fn:string($var6_cur))))       } </Parnr>,
         for $var7_cur in $var9_cur/Naam
         return            <Naam>              {              fn:string($var7_cur)              }            </Naam>,
         for $var8_cur in $var9_cur/EenheidOw
         return            <EenheidOw>              {              fn:string($var8_cur)              }            </EenheidOw>          )        }
      </Row>
      else ()     ) ) }
</Import>
```

Retrieve monitoring program from Water Database:

XQuery: http://gima.lekkerkerk.info/ProofOfConcept/MediatedSchema/MappingMapToWDB.xq

HTML:
http://gima.lekkerkerk.info/ProofOfConcept/MediatedSchema/query_federated_waterdatabase_2_select_WDB.html

H.3 Integrate results into WQR

XSLT: http://gima.lekkerkerk.info/ProofOfConcept/MediatedSchema/MappingMapTowaterdatabase.xslt

HTML:
http://gima.lekkerkerk.info/ProofOfConcept/MediatedSchema/query_federated_waterdatabase_3_integrate.html

XML output: http://gima.lekkerkerk.info/ProofOfConcept/MediatedSchema/waterdatabase_RDIJ_query-total.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!--This file was generated by Altova MapForce 2011r3

YOU SHOULD NOT MODIFY THIS FILE, BECAUSE IT WILL BE
OVERWRITTEN WHEN YOU RE-RUN CODE GENERATION.-->
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:vmf="http://www.altova.com/MapForce/UDF/vmf" xmlns:gmd="http://www.isotc211.org/2005/gmd"
xmlns:ns0="http://www.opengis.net/gml/3.2" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:fn="http://www.w3.org/2005/xpath-functions" xmlns:wdb="urn:ihw:gmlas:waterdatabase" exclude-result-prefixes="vmf xs fn">
 <xsl:template name="vmf:vmf1_inputtoresult">
  <xsl:param name="input" select="()"/>
  <xsl:choose>
   <xsl:when test="$input='mg P/l'">  <xsl:value-of select="'mg/l'"/>  </xsl:when>
   <xsl:when test="$input='mg N/l'">  <xsl:value-of select="'mg/l'"/>  </xsl:when>
   <xsl:when test="$input='-'">  <xsl:value-of select="'DMSLS'"/>  </xsl:when>
   <xsl:otherwise>  <xsl:value-of select="'NVT'"/>  </xsl:otherwise>
  </xsl:choose>
 </xsl:template>
 <xsl:template name="vmf:vmf2_inputtoresult">
  <xsl:param name="input" select="()"/>
  <xsl:choose>
   <xsl:when test="$input=xs:short('3')">  <xsl:value-of select="'&lt;'"/>  </xsl:when>
   <xsl:when test="$input=xs:short('2')">  <xsl:value-of select="'&gt;'"/>  </xsl:when>
  </xsl:choose>
 </xsl:template>
 <xsl:template name="vmf:vmf3_inputtoresult">
  <xsl:param name="input" select="()"/>
  <xsl:choose>
   <xsl:when test="$input='Temperatuur'">  <xsl:value-of select="'T'"/>  </xsl:when>
   <xsl:when test="$input='Zuurgraad (veldmeting)'">  <xsl:value-of select="'pH'"/>  </xsl:when>
   <xsl:when test="$input='Electrisch Geleidingsvermogen (veldmetin'">  <xsl:value-of select="'GELDHD'"/>  </xsl:when>
   <xsl:when test="$input='Doorzicht'">  <xsl:value-of select="'ZICHT'"/>  </xsl:when>
   <xsl:when test="$input='Diepte'">  <xsl:value-of select="'DIEPTE'"/>  </xsl:when>
   <xsl:when test="$input='Breedte'">  <xsl:value-of select="'BREEDTE'"/>  </xsl:when>
   <xsl:when test="$input='Alkaliniteit'">  <xsl:value-of select="'ALKLTT'"/>  </xsl:when>
   <xsl:when test="$input='Fenolftaline alkaliniteit'">  <xsl:value-of select="'ALKLTT'"/>  </xsl:when>
   <xsl:otherwise>  <xsl:value-of select="'NVT'"/>  </xsl:otherwise>
  </xsl:choose>
 </xsl:template>
 <xsl:template name="vmf:vmf6_inputtoresult">
  <xsl:param name="input" select="()"/>
  <xsl:choose>
   <xsl:when test="$input='Zuurstof'">  <xsl:value-of select="'O2'"/>  </xsl:when>
   <xsl:when test="$input='Zuurstofverzadigingspercentage'">  <xsl:value-of select="'O2'"/>  </xsl:when>
   <xsl:when test="$input='Biochemisch zuurstofverbruik over 5 dage'">  <xsl:value-of select="'BZV'"/>  </xsl:when>
   <xsl:when test="$input='Chemisch zuurstofverbruik'">  <xsl:value-of select="'CZV'"/>  </xsl:when>
   <xsl:when test="$input='Fosfor-totaal'">  <xsl:value-of select="'Ptot'"/>  </xsl:when>
   <xsl:when test="$input='ortho-Fosfaat'">  <xsl:value-of select="'PO4'"/>  </xsl:when>
   <xsl:when test="$input='Stikstof-totaal'">  <xsl:value-of select="'Ntot'"/>  </xsl:when>
   <xsl:when test="$input='Kjeldahl stikstof'">  <xsl:value-of select="'NKj'"/>  </xsl:when>
   <xsl:when test="$input='Ammoniak'">  <xsl:value-of select="'NH3'"/>  </xsl:when>
   <xsl:when test="$input='Ammonium'">  <xsl:value-of select="'NH4'"/>  </xsl:when>
   <xsl:when test="$input='Nitriet'">  <xsl:value-of select="'NO2'"/>  </xsl:when>
   <xsl:when test="$input='Nitraat'">  <xsl:value-of select="'NO3'"/>  </xsl:when>
   <xsl:when test="$input='Nitriet+nitraat'">  <xsl:value-of select="'sNO3NO2'"/>  </xsl:when>
   <xsl:when test="$input='Chlorofyl-a'">  <xsl:value-of select="'CHLFa'"/>  </xsl:when>
   <xsl:when test="$input='Pheo (Faeofytine)'">  <xsl:value-of select="'sFEO'"/>  </xsl:when>
   <xsl:when test="$input='Kalium'">  <xsl:value-of select="'K'"/>  </xsl:when>
   <xsl:when test="$input='Calcium'">  <xsl:value-of select="'Ca'"/>  </xsl:when>
   <xsl:when test="$input='IJzer'">  <xsl:value-of select="'Fe'"/>  </xsl:when>
   <xsl:when test="$input='Magnesium'">  <xsl:value-of select="'Mg'"/>  </xsl:when>
   <xsl:when test="$input='Natrium'">  <xsl:value-of select="'Na'"/>  </xsl:when>
   <xsl:when test="$input='Chloride'">  <xsl:value-of select="'Cl'"/>  </xsl:when>
   <xsl:when test="$input='Sulfaat'">  <xsl:value-of select="'SO4'"/>  </xsl:when>
```

```
<xsl:when test="$input='Bicarbonaat'">      <xsl:value-of select="'CO3'"/>    </xsl:when>
<xsl:when test="$input='Al-filtraat'">     <xsl:value-of select="'Al'"/>    </xsl:when>
<xsl:when test="$input='Cd-filtraat'">      <xsl:value-of select="'Cd'"/>     </xsl:when>
<xsl:when test="$input='Cu-filtraat'">      <xsl:value-of select="'Cu'"/>     </xsl:when>
<xsl:when test="$input='Fe-filtraat'">      <xsl:value-of select="'Fe'"/>     </xsl:when>
<xsl:when test="$input='Zn-filtraat'">      <xsl:value-of select="'Zn'"/>     </xsl:when>
<xsl:when test="$input='As-filtraat'">      <xsl:value-of select="'As'"/>     </xsl:when>
<xsl:when test="$input='Ca-filtraat'">      <xsl:value-of select="'Ca'"/>     </xsl:when>
<xsl:when test="$input='Co-filtraat'">      <xsl:value-of select="'Co'"/>     </xsl:when>
<xsl:when test="$input='Cr-filtraat'">      <xsl:value-of select="'Cr'"/>   </xsl:when>
<xsl:when test="$input='Hg-filtraat'">      <xsl:value-of select="'Hg'"/>     </xsl:when>
<xsl:when test="$input='K-filtraat'">     <xsl:value-of select="'K'"/>     </xsl:when>
<xsl:when test="$input='Mg-filtraat'">      <xsl:value-of select="'Mg'"/>     </xsl:when>
<xsl:when test="$input='Mn-filtraat'">      <xsl:value-of select="'Mn'"/>     </xsl:when>
<xsl:when test="$input='Na-filtraat'">      <xsl:value-of select="'Na'"/>     </xsl:when>
<xsl:when test="$input='Ni-filtraat'">     <xsl:value-of select="'Ni'"/>    </xsl:when>
<xsl:when test="$input='Pb-filtraat'">      <xsl:value-of select="'Pb'"/>     </xsl:when>
<xsl:when test="$input='Sn-filtraat'">      <xsl:value-of select="'Sn'"/>     </xsl:when>
<xsl:when test="$input='Ag filtraat'">      <xsl:value-of select="'Ag'"/>     </xsl:when>
<xsl:when test="$input='Aluminium'">       <xsl:value-of select="'Al'"/>     </xsl:when>
<xsl:when test="$input='Arseen'">      <xsl:value-of select="'As'"/>     </xsl:when>
<xsl:when test="$input='Cadmium'">      <xsl:value-of select="'Cd'"/>     </xsl:when>
<xsl:when test="$input='Chroom'">      <xsl:value-of select="'Cr'"/>    </xsl:when>
<xsl:when test="$input='Koper'">      <xsl:value-of select="'Cu'"/>     </xsl:when>
<xsl:when test="$input='Kwik'">      <xsl:value-of select="'Hg'"/>     </xsl:when>
<xsl:when test="$input='Nikkel'">      <xsl:value-of select="'Ni'"/>     </xsl:when>
<xsl:when test="$input='Lood'">      <xsl:value-of select="'Pb'"/>     </xsl:when>
<xsl:when test="$input='Zink'">      <xsl:value-of select="'Zn'"/>     </xsl:when>
<xsl:when test="$input='Silicium'">      <xsl:value-of select="'Si'"/>     </xsl:when>
<xsl:when test="$input='BaA (Benzo(a)antraceen)'">      <xsl:value-of select="'BaA'"/>     </xsl:when>
<xsl:when test="$input='BaP (Benzo(a)pyreen)'">      <xsl:value-of select="'BaP'"/>     </xsl:when>
<xsl:when test="$input='BbF (Benzo(b)fluorantheen)'">      <xsl:value-of select="'BbF'"/>     </xsl:when>
<xsl:when test="$input='BkF (Benzo(k)fluorantheen)'">      <xsl:value-of select="'BkF'"/>     </xsl:when>
<xsl:when test="$input='Chr (Chryseen)'">      <xsl:value-of select="'Chr'"/>     </xsl:when>
<xsl:when test="$input='Ant  (Anthraceen)'">      <xsl:value-of select="'Ant'"/>     </xsl:when>
<xsl:when test="$input='Fle (Fluoreen)'">      <xsl:value-of select="'Fle'"/>     </xsl:when>
<xsl:when test="$input='Flu (Fluorantheen)'">      <xsl:value-of select="'Flu'"/>     </xsl:when>
<xsl:when test="$input='Naf  (Naftaleen)'">      <xsl:value-of select="'Naf'"/>     </xsl:when>
<xsl:when test="$input='Pyr (Pyreen)'">      <xsl:value-of select="'Pyr'"/>     </xsl:when>
<xsl:when test="$input='Fen (Fenanthreen)'">      <xsl:value-of select="'Fen'"/>     </xsl:when>
<xsl:when test="$input='InP (Indeno(1,2,3-c,d)pyreen)'">      <xsl:value-of select="'InP'"/>     </xsl:when>
<xsl:when test="$input='AcNe (Acenafteen)'">      <xsl:value-of select="'AcNe'"/>     </xsl:when>
<xsl:when test="$input='AcNy (Acenaftyleen)'">      <xsl:value-of select="'AcNy'"/>     </xsl:when>
<xsl:when test="$input='BghiP (Benzo(ghi)peryleen)'">      <xsl:value-of select="'BghiPe'"/>     </xsl:when>
<xsl:when test="$input='DBahAnt (Dibenzo(a,h)antraceen)'">      <xsl:value-of select="'DBahAnt'"/>     </xsl:when>
<xsl:when test="$input='Co Cobalt'">      <xsl:value-of select="'Co'"/>     </xsl:when>
<xsl:when test="$input='Lithium'">      <xsl:value-of select="'Li'"/>     </xsl:when>
<xsl:when test="$input='Sn (Tin)'">      <xsl:value-of select="'Sn'"/>     </xsl:when>
<xsl:when test="$input='Ald (Aldrin)'">      <xsl:value-of select="'Ald'"/>     </xsl:when>
<xsl:when test="$input='aEndo (alfa-Endosulfan)'">      <xsl:value-of select="'aedsfn'"/>     </xsl:when>
<xsl:when test="$input='aHCH (alfa-Hexachloorcyclohexaan)'">      <xsl:value-of select="'aHCH'"/>     </xsl:when>
<xsl:when test="$input='Atrazine'">      <xsl:value-of select="'atzne'"/>     </xsl:when>
<xsl:when test="$input='bHCH (beta-Hexachloorcyclohexaan)'">      <xsl:value-of select="'bHCH'"/>     </xsl:when>
<xsl:when test="$input='dHCH (delta-Hexachloorcyclohexaan)'">      <xsl:value-of select="'dHCH'"/>     </xsl:when>
<xsl:when test="$input='Diazinon'">      <xsl:value-of select="'Dazn'"/>     </xsl:when>
<xsl:when test="$input='Dld (Dieldrin)'">      <xsl:value-of select="'Dld'"/>     </xsl:when>
<xsl:when test="$input='Dimethoaat'">      <xsl:value-of select="'Dmtat'"/>     </xsl:when>
<xsl:when test="$input='End (Endrin)'">      <xsl:value-of select="'endn'"/>     </xsl:when>
<xsl:when test="$input='cHCH (gamma-Hexachloorcyclohexaan)'">      <xsl:value-of select="'cHCH'"/>     </xsl:when>
<xsl:when test="$input='Hepta (Heptachloor)'">      <xsl:value-of select="'HpCl'"/>     </xsl:when>
<xsl:when test="$input='Hepo (Heptachloorepoxide)'">      <xsl:value-of select="'sHpCl2'"/>     </xsl:when>
<xsl:when test="$input='HCB (Hexachloorbenzeen)'">      <xsl:value-of select="'HCB'"/>     </xsl:when>
<xsl:when test="$input='Isd (Isodrin)'">      <xsl:value-of select="'idn'"/>     </xsl:when>
<xsl:when test="$input='Malathion'">     <xsl:value-of select="'malton'"/>     </xsl:when>
<xsl:when test="$input='Methyl tolclofos'">      <xsl:value-of select="'tolcfsC1y'"/>     </xsl:when>
<xsl:when test="$input='Methylparathion'">      <xsl:value-of select="'C1yprton'"/>     </xsl:when>
<xsl:when test="$input='Propazine'">      <xsl:value-of select="'propzne'"/>     </xsl:when>
<xsl:when test="$input='Simazine'">      <xsl:value-of select="'simzne'"/>     </xsl:when>
<xsl:when test="$input='opDDD (2,4-dichloordifenyldichloorethaan'">      <xsl:value-of select="'24DDD'"/>     </xsl:when>
<xsl:when test="$input='opDDE (2,4-dichloordifenyldichlooretheen'">      <xsl:value-of select="'24DDE'"/>     </xsl:when>
<xsl:when test="$input='ppDDD(4,4-dichloordifenyldichloorethaan)'">      <xsl:value-of select="'44DDD'"/>     </xsl:when>
<xsl:when test="$input='som PAK 6 Borneff'">      <xsl:value-of select="'sPAK6'"/>     </xsl:when>
<xsl:when test="$input='som PAK 10 (VROM/Leidraad)'">      <xsl:value-of select="'sPAK10'"/>     </xsl:when>
<xsl:when test="$input='ppDDE (4,4-dichloordifenyldichlooretheen'">      <xsl:value-of select="'44DDE'"/>     </xsl:when>
<xsl:when test="$input='ppDDT(4,4-dichloordifenyltrichloorethaan'">      <xsl:value-of select="'44DDT'"/>     </xsl:when>
<xsl:when test="$input='Chloorfenvinfos'">      <xsl:value-of select="'Clfvfs'"/>     </xsl:when>
```

```
    <xsl:when test="$input='Dichloorvos'">      <xsl:value-of select="'DCLvs'"/>    </xsl:when>
    <xsl:when test="$input='Mev (Mevinfos)'">      <xsl:value-of select="'mevfs'"/>    </xsl:when>
    <xsl:when test="$input='Ethylparathion'">      <xsl:value-of select="'C2yprton'"/>    </xsl:when>
    <xsl:when test="$input='opDDT(2,4-dichloordifenyltrichloorethaan'">      <xsl:value-of select="'24DDT'"/>    </xsl:when>
    <xsl:when test="$input='Ag (Zilver)'">      <xsl:value-of select="'Ag'"/>      </xsl:when>
    <xsl:when test="$input='DOC opgelost organisch koolstof'">      <xsl:value-of select="'C'"/>      </xsl:when>
    <xsl:when test="$input='TOC totaal organisch koolstof'">      <xsl:value-of select="'C'"/>      </xsl:when>
    <xsl:when test="$input='Mn Mangaan'">      <xsl:value-of select="'Mn'"/>      </xsl:when>
    <xsl:otherwise>      <xsl:value-of select="'NVT'"/>      </xsl:otherwise>
  </xsl:choose>
</xsl:template>
<xsl:template name="vmf:vmf8_inputtoresult">
  <xsl:param name="input" select="()"/>
  <xsl:choose>
    <xsl:when test="$input='mg P/l'">      <xsl:value-of select="'P'"/>      </xsl:when>
    <xsl:when test="$input='mg N/l'">      <xsl:value-of select="'N'"/>      </xsl:when>
    <xsl:otherwise>      <xsl:value-of select="'NVT'"/>      </xsl:otherwise>
  </xsl:choose>
</xsl:template>
<xsl:template name="vmf:vmf9_inputtoresult">
  <xsl:param name="input" select="()"/>
  <xsl:choose>
    <xsl:when test="$input='T'">      <xsl:value-of select="'T'"/>    </xsl:when>
    <xsl:when test="$input='ZICHT'">      <xsl:value-of select="'ZICHT'"/>      </xsl:when>
    <xsl:when test="$input='HH'">      <xsl:value-of select="'HH'"/>      </xsl:when>
    <xsl:when test="$input='SALNTT'">      <xsl:value-of select="'SALNTT'"/>      </xsl:when>
    <xsl:when test="$input='DIEPTE'">      <xsl:value-of select="'DIEPTE'"/>      </xsl:when>
    <xsl:when test="$input='GELDHD'">      <xsl:value-of select="'GELDHD'"/>      </xsl:when>
    <xsl:when test="$input='pH'">      <xsl:value-of select="'pH'"/>      </xsl:when>
    <xsl:when test="$input='alkltt'">      <xsl:value-of select="'ALKLTT'"/>      </xsl:when>
    <xsl:when test="$input='Folflnaklntt'">      <xsl:value-of select="'ALKLTT'"/>      </xsl:when>
    <xsl:otherwise>      <xsl:value-of select="'NVT'"/>      </xsl:otherwise>
  </xsl:choose>
</xsl:template>
<xsl:template name="vmf:vmf10_inputtoresult">
  <xsl:param name="input" select="()"/>
  <xsl:choose>
    <xsl:when test="$input='%'">      <xsl:value-of select="'VOLMFTE'"/>    </xsl:when>
    <xsl:when test="$input='g'">      <xsl:value-of select="'MASSA'"/>    </xsl:when>
    <xsl:when test="$input='g/dl'">      <xsl:value-of select="'CONCTTE'"/>    </xsl:when>
    <xsl:when test="$input='g/kg'">      <xsl:value-of select="'MASSFTE'"/>    </xsl:when>
    <xsl:when test="$input='g/l'">      <xsl:value-of select="'CONCTTE'"/>    </xsl:when>
    <xsl:when test="$input='g/ml'">      <xsl:value-of select="'CONCTTE'"/>    </xsl:when>
    <xsl:when test="$input='kg'">      <xsl:value-of select="'MASSA'"/>    </xsl:when>
    <xsl:when test="$input='l'">      <xsl:value-of select="'VOLME'"/>    </xsl:when>
    <xsl:when test="$input='m3'">      <xsl:value-of select="'VOLME'"/>    </xsl:when>
    <xsl:when test="$input='m3/d'">      <xsl:value-of select="'Q'"/>    </xsl:when>
    <xsl:when test="$input='m3/h'">      <xsl:value-of select="'Q'"/>    </xsl:when>
    <xsl:when test="$input='m3/s'">      <xsl:value-of select="'Q'"/>    </xsl:when>
    <xsl:when test="$input='meq/l'">      <xsl:value-of select="'CONCTTE'"/>    </xsl:when>
    <xsl:when test="$input='mg/kg'">      <xsl:value-of select="'MASSFTE'"/>    </xsl:when>
    <xsl:when test="$input='mg/l'">      <xsl:value-of select="'CONCTTE'"/>    </xsl:when>
    <xsl:when test="$input='mg/m3'">      <xsl:value-of select="'CONCTTE'"/>    </xsl:when>
    <xsl:when test="$input='ml'">      <xsl:value-of select="'VOLME'"/>    </xsl:when>
    <xsl:when test="$input='ml/l'">      <xsl:value-of select="'VOLMFTE'"/>    </xsl:when>
    <xsl:when test="$input='mmol/l'">      <xsl:value-of select="'CONCTTE'"/>    </xsl:when>
    <xsl:when test="$input='mol/l'">      <xsl:value-of select="'CONCTTE'"/>    </xsl:when>
    <xsl:when test="$input='mol/m3'">      <xsl:value-of select="'CONCTTE'"/>    </xsl:when>
    <xsl:when test="$input='mol/mol'">      <xsl:value-of select="'MASSFTE'"/>    </xsl:when>
    <xsl:when test="$input='n'">      <xsl:value-of select="'AANTL'"/>    </xsl:when>
    <xsl:when test="$input='n/dl'">      <xsl:value-of select="'AANTPVLME'"/>    </xsl:when>
    <xsl:when test="$input='n/km2'">      <xsl:value-of select="'AANTPOPVTE'"/>    </xsl:when>
    <xsl:when test="$input='n/l'">      <xsl:value-of select="'AANTPVLME'"/>    </xsl:when>
    <xsl:when test="$input='n/m2'">      <xsl:value-of select="'AANTPOPVTE'"/>    </xsl:when>
    <xsl:when test="$input='n/ml'">      <xsl:value-of select="'AANTPVLME'"/>    </xsl:when>
    <xsl:when test="$input='ng/kg'">      <xsl:value-of select="'MASSFTE'"/>    </xsl:when>
    <xsl:when test="$input='ng/l'">      <xsl:value-of select="'CONCTTE'"/>    </xsl:when>
    <xsl:when test="$input='ppm'">      <xsl:value-of select="'MASFFTE'"/>    </xsl:when>
    <xsl:when test="$input='ton'">      <xsl:value-of select="'MASSA'"/>    </xsl:when>
    <xsl:when test="$input='ug/kg'">      <xsl:value-of select="'MASFFTE'"/>    </xsl:when>
    <xsl:when test="$input='ug/l'">      <xsl:value-of select="'CONCTTE'"/>    </xsl:when>
    <xsl:when test="$input='ug/mg'">      <xsl:value-of select="'MASFFTE'"/>    </xsl:when>
  </xsl:choose>
</xsl:template>
<xsl:template name="vmf:vmf11_inputtoresult">
  <xsl:param name="input" select="()"/>
```

```
  <xsl:choose>
    <xsl:when test="$input='ZS'">      <xsl:value-of select="'ZS'"/>     </xsl:when>
    <xsl:when test="$input='GR'">      <xsl:value-of select="'GR'"/>     </xsl:when>
    <xsl:when test="$input='MCCYSTe'">      <xsl:value-of select="'MICCTNS'"/>     </xsl:when>
    <xsl:otherwise>       <xsl:value-of select="'NVT'"/>     </xsl:otherwise>
  </xsl:choose>
 </xsl:template>
 <xsl:template name="vmf:vmf12_inputtoresult">
  <xsl:param name="input" select="()"/>
  <xsl:choose>
    <xsl:when test="$input='metzCl'">      <xsl:value-of select="'mzCl'"/>     </xsl:when>
    <xsl:when test="$input='ptonC1y'">      <xsl:value-of select="'C1yprton'"/>     </xsl:when>
    <xsl:when test="$input='terbtne'">      <xsl:value-of select="'terbtn'"/>     </xsl:when>
    <xsl:when test="$input='DOC'">      <xsl:value-of select="'Corg'"/>     </xsl:when>
    <xsl:when test="$input='N'">      <xsl:value-of select="'Ntot'"/>     </xsl:when>
    <xsl:when test="$input='POC'">      <xsl:value-of select="'Porg'"/>     </xsl:when>
    <xsl:when test="$input='C2ypton'">      <xsl:value-of select="'C2yprton'"/>     </xsl:when>
    <xsl:when test="$input='spton2'">      <xsl:value-of select="'sprton2'"/>     </xsl:when>
    <xsl:when test="$input='cbedzm'">      <xsl:value-of select="'carbdzm'"/>     </xsl:when>
    <xsl:when test="$input='sorgSn'">      <xsl:value-of select="'bulk:sorgSn'"/>     </xsl:when>
    <xsl:when test="$input='mtmtn'">      <xsl:value-of select="'mmtn'"/>     </xsl:when>
    <xsl:when test="$input='etofcbsO'">      <xsl:value-of select="'etofcbSO'"/>     </xsl:when>
    <xsl:when test="$input='alDcbsfn'">      <xsl:value-of select="'alDcsfn'"/>     </xsl:when>
    <xsl:when test="$input='cbfrn'">      <xsl:value-of select="'carbrn'"/>     </xsl:when>
    <xsl:when test="$input='cbrl'">      <xsl:value-of select="'carbrl'"/>     </xsl:when>
    <xsl:when test="$input='Clxrn'">      <xsl:value-of select="'bulk:Clxrn'"/>     </xsl:when>
    <xsl:when test="$input='metocbsO'">      <xsl:value-of select="'metocbSO'"/>     </xsl:when>
    <xsl:when test="$input='pirmfC1y'">      <xsl:value-of select="'C1yprmfs'"/>     </xsl:when>
    <xsl:when test="$input='doDne'">      <xsl:value-of select="'dodne'"/>     </xsl:when>
    <xsl:when test="$input='cbeAd'">      <xsl:value-of select="'carbtAd'"/>     </xsl:when>
    <xsl:when test="$input='sulctone'">      <xsl:value-of select="'sulcton'"/>     </xsl:when>
    <xsl:when test="$input='cbOxn'">      <xsl:value-of select="'carbOxn'"/>     </xsl:when>
    <xsl:when test="$input='P'">      <xsl:value-of select="'Ptot'"/>     </xsl:when>
    <xsl:when test="$input='Dcrn'">      <xsl:value-of select="'26DCl4NO2An'"/>     </xsl:when>
    <xsl:when test="$input='dffncn'">      <xsl:value-of select="'Dffncn'"/>     </xsl:when>
    <xsl:when test="$input='mecpp'">      <xsl:value-of select="'bulk:mecpp'"/>     </xsl:when>
    <xsl:otherwise>       <xsl:value-of select="'NVT'"/>     </xsl:otherwise>
  </xsl:choose>
 </xsl:template>
 <xsl:template name="vmf:vmf13_inputtoresult">
  <xsl:param name="input" select="()"/>
  <xsl:choose>
    <xsl:when test="$input='DOC'">      <xsl:value-of select="'Cnf'"/>     </xsl:when>
    <xsl:when test="$input='POC'">      <xsl:value-of select="'Cpg'"/>     </xsl:when>
    <xsl:otherwise>       <xsl:value-of select="'NVT'"/>     </xsl:otherwise>
  </xsl:choose>
 </xsl:template>
 <xsl:output method="xml" encoding="UTF-8" indent="yes"/>
 <xsl:param name="BULK_MWA2" select="'../../../XML_base/BULK_MWA_query.xml'"/>
 <xsl:param name="BULK_WNS2" select="'../../../XML_base/BULK_WNS_query.xml'"/>
 <xsl:param name="LD_MON22" select="'../../../XML_base/LD_MON2.xml'"/>
 <xsl:param name="LD_PAR2" select="'../../../XML_base/LD_PAR.xml'"/>
 <xsl:param name="MPN_Query_Results2" select="'MPN_Query_Results.xml'"/>
 <xsl:template match="/">
   <xsl:variable name="var1_FeatureCollection" as="node()?" select="ns0:FeatureCollection"/>
   <xsl:variable name="var2_Import" as="node()?" select="fn:doc($BULK_MWA2)/Import"/>
   <xsl:variable name="var3_Import" as="node()?" select="fn:doc($LD_MON22)/Import"/>
   <xsl:variable name="var4_Import" as="node()?" select="fn:doc($LD_PAR2)/Import"/>
   <xsl:variable name="var5_Import" as="node()?" select="fn:doc($BULK_WNS2)/Import"/>
   <gml:FeatureCollection xmlns:gml="http://www.opengis.net/gml/3.2" xmlns:gco="http://www.isotc211.org/2005/gco"
xmlns:gmd="http://www.isotc211.org/2005/gmd" xmlns:gsr="http://www.isotc211.org/2005/gsr"
xmlns:gss="http://www.isotc211.org/2005/gss" xmlns:gts="http://www.isotc211.org/2005/gts"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:wdb="urn:ihw:gmlas:waterdatabase" xmlns:base="urn:x-
inspire:specification:gmlas:BaseTypes:3.2" xmlns:gn="urn:x-inspire:specification:gmlas:GeographicalNames:3.0">
     <xsl:attribute name="xsi:schemaLocation" select="'http://www.opengis.net/gml/3.2 Y:/projects/P0902-GIMA/MODULE~9/03-
SOLL/waterdatabase/waterdatabase.xsd'"/>
     <xsl:attribute name="gml:id" select="fn:substring-before(fn:concat('_query_results_', xs:string(fn:current-date())), '+')"/>
     <gml:featureMembers>
       <xsl:for-each select="$var1_FeatureCollection/gml:featureMembers/wdb:EnvironmentalMonitoringFacility">
         <xsl:variable name="var31_cur" as="node()" select="."/>
         <xsl:variable name="var6_QueryResults" as="node()?" select="fn:doc($MPN_Query_Results2)/QueryResults"/>
         <xsl:variable name="var9_result" as="xs:boolean?">
           <xsl:for-each select="$var6_QueryResults">
             <xsl:variable name="var8_result" as="xs:boolean*">
               <xsl:for-each select="MPN">
                 <xsl:variable name="var7_result" as="xs:boolean?">
```

```
        <xsl:for-each select="WDB_Ident">
          <xsl:sequence select="(fn:string(.) = fn:string($var31_cur/@gml:id))"/>
        </xsl:for-each>
      </xsl:variable>
      <xsl:sequence select="fn:exists($var7_result[.])"/>
    </xsl:for-each>
  </xsl:variable>
  <xsl:sequence select="fn:exists($var8_result[.])"/>
 </xsl:for-each>
</xsl:variable>
<xsl:if test="fn:exists($var9_result[.])">
 <xsl:variable name="var10_resultof_cast" as="xs:string" select="fn:string(@gml:id)"/>
 <wdb:EnvironmentalMonitoringFacility>
  <xsl:attribute name="gml:id" select="$var10_resultof_cast"/>
  <xsl:for-each select="wdb:inspireId">
   <wdb:inspireId> <xsl:sequence select="(./@node(), ./node())"/></wdb:inspireId>
  </xsl:for-each>
  <xsl:for-each select="wdb:name">
   <wdb:name>              <xsl:sequence select="fn:string(.)"/>             </wdb:name>
  </xsl:for-each>
  <xsl:for-each select="wdb:additionalDescription">
   <wdb:additionalDescription>              <xsl:sequence select="fn:string(.)"/>             </wdb:additionalDescription>
  </xsl:for-each>
  <xsl:for-each select="wdb:mediaMonitored">
   <wdb:mediaMonitored>              <xsl:sequence select="(./@node(), ./node())"/>             </wdb:mediaMonitored>
  </xsl:for-each>
  <xsl:for-each select="wdb:geometry">
   <wdb:geometry>              <xsl:sequence select="(./@node(), ./node())"/>             </wdb:geometry>
  </xsl:for-each>
  <xsl:for-each select="wdb:legalBackground">
   <wdb:legalBackground>              <xsl:sequence select="(./@node(), ./node())"/>             </wdb:legalBackground>
  </xsl:for-each>
  <xsl:for-each select="wdb:responsibleParty">
   <wdb:responsibleParty>              <xsl:sequence select="(./@node(), ./node())"/>             </wdb:responsibleParty>
  </xsl:for-each>
  <xsl:for-each select="wdb:onlineResource">
   <wdb:onlineResource>              <xsl:sequence select="(./@node(), ./node())"/>             </wdb:onlineResource>
  </xsl:for-each>
  <xsl:for-each select="wdb:purpose">
   <wdb:purpose>              <xsl:sequence select="(./@node(), ./node())"/>             </wdb:purpose>
  </xsl:for-each>
  <xsl:for-each select="wdb:observingCapability">
   <wdb:observingCapability>              <xsl:sequence select="(./@node(), ./node())"/>             </wdb:observingCapability>
  </xsl:for-each>
  <xsl:for-each select="wdb:reportedTo">
   <wdb:reportedTo>              <xsl:sequence select="(./@node(), ./node())"/>             </wdb:reportedTo>
  </xsl:for-each>
  <xsl:for-each select="$var2_Import/Row">
   <xsl:variable name="var19_cur" as="node()" select="."/>
   <xsl:variable name="var18_result" as="xs:boolean?">
    <xsl:for-each select="$var6_QueryResults">
     <xsl:variable name="var17_result" as="xs:boolean*">
      <xsl:for-each select="MPN">
       <xsl:variable name="var16_cur" as="node()" select="."/>
       <xsl:variable name="var15_result" as="xs:boolean?">
        <xsl:for-each select="WDB_Ident">
         <xsl:variable name="var14_cur" as="node()" select="."/>
         <xsl:variable name="var13_result" as="xs:boolean?">
          <xsl:for-each select="$var16_cur/BULK_Ident">
           <xsl:variable name="var12_cur" as="node()" select="."/>
           <xsl:variable name="var11_result" as="xs:boolean?">
            <xsl:for-each select="$var19_cur/MWADTME">
             <xsl:sequence select="((fn:string($var14_cur) = $var10_resultof_cast) and (fn:string($var12_cur) =
                 fn:string(.)))"/>
            </xsl:for-each>
           </xsl:variable>
           <xsl:sequence select="fn:exists($var11_result[.])"/>
          </xsl:for-each>
         </xsl:variable>
         <xsl:sequence select="fn:exists($var13_result[.])"/>
        </xsl:for-each>
       </xsl:variable>
       <xsl:sequence select="fn:exists($var15_result[.])"/>
      </xsl:for-each>
     </xsl:variable>
     <xsl:sequence select="fn:exists($var17_result[.])"/>
```

```
    </xsl:for-each>
  </xsl:variable>
  <xsl:if test="fn:exists($var18_result[.])">
    <wdb:relatedObservation>
      <xsl:attribute name="xlink:type" select="'simple'"/>
      <xsl:for-each select="MWA_ID">
        <xsl:attribute name="xlink:href" select="xs:string(xs:anyURI(fn:concat('#', fn:concat('BULK.MWA.',
            xs:string(xs:integer(fn:string(.)))))))"/>
      </xsl:for-each>
    </wdb:relatedObservation>
  </xsl:if>
</xsl:for-each>
<xsl:for-each select="$var3_Import/Row">
  <xsl:variable name="var28_cur" as="node()" select="."/>
  <xsl:variable name="var27_result" as="xs:boolean?">
    <xsl:for-each select="$var6_QueryResults">
      <xsl:variable name="var26_result" as="xs:boolean*">
        <xsl:for-each select="MPN">
          <xsl:variable name="var25_cur" as="node()" select="."/>
          <xsl:variable name="var24_result" as="xs:boolean?">
            <xsl:for-each select="WDB_Ident">
              <xsl:variable name="var23_cur" as="node()" select="."/>
              <xsl:variable name="var22_result" as="xs:boolean?">
                <xsl:for-each select="$var25_cur/LD_Ident">
                  <xsl:variable name="var21_cur" as="node()" select="."/>
                  <xsl:variable name="var20_result" as="xs:boolean?">
                    <xsl:for-each select="$var28_cur/Lab">
                      <xsl:sequence select="((fn:string($var23_cur) = $var10_resultof_cast) and (fn:string($var21_cur)
                          = fn:string(.)))"/>
                    </xsl:for-each>
                  </xsl:variable>
                  <xsl:sequence select="fn:exists($var20_result[.])"/>
                </xsl:for-each>
              </xsl:variable>
              <xsl:sequence select="fn:exists($var22_result[.])"/>
            </xsl:for-each>
          </xsl:variable>
          <xsl:sequence select="fn:exists($var24_result[.])"/>
        </xsl:for-each>
      </xsl:variable>
      <xsl:sequence select="fn:exists($var26_result[.])"/>
    </xsl:for-each>
  </xsl:variable>
  <xsl:if test="fn:exists($var27_result[.])">
    <wdb:relatedObservation>
      <xsl:attribute name="xlink:type" select="'simple'"/>
      <xsl:for-each select="Recnum">
        <xsl:attribute name="xlink:href" select="xs:string(xs:anyURI(fn:concat('#', fn:concat('LD.MON.',
            xs:string(xs:integer(fn:string(.)))))))"/>
      </xsl:for-each>
    </wdb:relatedObservation>
  </xsl:if>
</xsl:for-each>
<xsl:for-each select="wdb:sampledFeature">
  <wdb:sampledFeature>  <xsl:sequence select="(./@node(), ./node())"/>  </wdb:sampledFeature>
</xsl:for-each>
<xsl:for-each select="wdb:positionalAccuracy">
  <xsl:variable name="var30_cur" as="node()" select="."/>
  <xsl:variable name="var29_result" as="xs:boolean*">
    <xsl:for-each select="@xsi:type">
      <xsl:sequence select="(fn:resolve-QName(fn:string(.), $var30_cur) =
          xs:QName('gmd:DQ_RelativeInternalPositionalAccuracy_Type'))"/>
    </xsl:for-each>
  </xsl:variable>
  <xsl:if test="fn:exists($var29_result[.])">
    <wdb:positionalAccuracy>
      <xsl:attribute name="xsi:type" select="xs:QName('gmd:DQ_RelativeInternalPositionalAccuracy_Type')"/>
    </wdb:positionalAccuracy>
  </xsl:if>
</xsl:for-each>
<xsl:for-each select="wdb:hostedProcedure">
  <wdb:hostedProcedure>  <xsl:sequence select="(./@node(), ./node())"/>  </wdb:hostedProcedure>
</xsl:for-each>
<xsl:for-each select="wdb:measurementRegime">
  <wdb:measurementRegime>
    <xsl:choose>
```

```
        <xsl:when test="(fn:translate(fn:string(@xsi:nil), 'true ', '1') = '1')">
          <xsl:attribute name="xsi:nil" select="'true'"/>
        </xsl:when>
        <xsl:otherwise>          <xsl:sequence select="fn:string(.)"/>          </xsl:otherwise>
      </xsl:choose>
    </wdb:measurementRegime>
  </xsl:for-each>
  <xsl:for-each select="wdb:mobile">
    <wdb:mobile>
      <xsl:choose>
        <xsl:when test="(fn:translate(fn:string(@xsi:nil), 'true ', '1') = '1')">
          <xsl:attribute name="xsi:nil" select="'true'"/>
        </xsl:when>
        <xsl:otherwise>          <xsl:sequence select="xs:string(xs:boolean(fn:string(.)))"/>          </xsl:otherwise>
      </xsl:choose>
    </wdb:mobile>
  </xsl:for-each>
  <xsl:for-each select="wdb:resultAcquisitionSource">
    <wdb:resultAcquisitionSource>
      <xsl:choose>
        <xsl:when test="(fn:translate(fn:string(@xsi:nil), 'true ', '1') = '1')">
          <xsl:attribute name="xsi:nil" select="'true'"/>
        </xsl:when>
        <xsl:otherwise>          <xsl:sequence select="fn:string(.)"/>          </xsl:otherwise>
      </xsl:choose>
    </wdb:resultAcquisitionSource>
  </xsl:for-each>
  <xsl:for-each select="wdb:speciallisedEMFType">
    <wdb:speciallisedEMFType>          <xsl:sequence select="(./@node(), ./node())"/>  </wdb:speciallisedEMFType>
  </xsl:for-each>
  <xsl:for-each select="wdb:operationalActivityPeriod">
    <wdb:operationalActivityPeriod>          <xsl:sequence select="(./@node(), ./node())"/></wdb:operationalActivityPeriod>
  </xsl:for-each>
 </wdb:EnvironmentalMonitoringFacility>
</xsl:if>
</xsl:for-each>
<xsl:for-each select="$var3_Import/Row">
 <xsl:variable name="var57_cur" as="node()" select="."/>
 <xsl:variable name="var32_resultof_create_attribute" as="node()">
  <xsl:attribute name="xlink:type" select="'simple'"/>
 </xsl:variable>
 <xsl:variable name="var33_Datum" as="node()?" select="Datum"/>
 <xsl:variable name="var34_Recnum" as="node()?" select="Recnum"/>
 <xsl:variable name="var35_Parnr" as="node()?" select="Parnr"/>
 <wdb:OM_Observation>
  <xsl:for-each select="$var34_Recnum">
   <xsl:attribute name="gml:id" select="fn:concat('LD.MON.', xs:string(xs:integer(fn:string(.))))"/>
  </xsl:for-each>
  <wdb:phenomenonTime>
   <xsl:attribute name="xsi:type" select="xs:QName('gml:TimeInstantType')"/>
   <xsl:for-each select="$var34_Recnum">
    <xsl:attribute name="gml:id" select="fn:concat(fn:concat('LD.MON.', xs:string(xs:integer(fn:string(.)))), 'pt')"/>
   </xsl:for-each>
   <xsl:for-each select="$var33_Datum">
    <xsl:variable name="var42_cur" as="node()" select="."/>
    <xsl:for-each select="$var57_cur/Tijd">
     <xsl:variable name="var36_resultof_cast" as="xs:string" select="fn:string($var42_cur)"/>
     <xsl:variable name="var37_resultof_substring_after" as="xs:string" select="fn:substring-after($var36_resultof_cast, '-')"/>
     <xsl:variable name="var38_resultof_substring_before" as="xs:string"
                   select="fn:substring-before($var36_resultof_cast, '-')"/>
     <xsl:variable name="var39_resultof_substring_before" as="xs:string"
                   select="fn:substring-before($var37_resultof_substring_after, '-')"/>
     <xsl:variable name="var40_result" as="xs:string">
      <xsl:choose>
       <xsl:when test="(fn:string-length($var39_resultof_substring_before) = xs:decimal('1'))">
        <xsl:sequence select="fn:concat('0', $var39_resultof_substring_before)"/>
       </xsl:when>
       <xsl:otherwise>          <xsl:sequence select="$var39_resultof_substring_before"/>          </xsl:otherwise>
      </xsl:choose>
     </xsl:variable>
     <xsl:variable name="var41_result" as="xs:string">
      <xsl:choose>
       <xsl:when test="(fn:string-length($var38_resultof_substring_before) = xs:decimal('1'))">
        <xsl:sequence select="fn:concat('0', $var38_resultof_substring_before)"/>
       </xsl:when>
       <xsl:otherwise>          <xsl:sequence select="$var38_resultof_substring_before"/>          </xsl:otherwise>
```

```
      </xsl:choose>
    </xsl:variable>
    <gml:timePosition>
      <xsl:sequence select="fn:concat(fn:concat(fn:concat(fn:concat(fn:concat(fn:concat(fn:substring-before(
                            fn:substring-after($var37_resultof_substring_after, '-'), ' '), '-'), $var40_result), '-'), $var41_result), 'T'),
                            fn:substring-after(fn:string(.), ' '))"/>
    </gml:timePosition>
   </xsl:for-each>
  </xsl:for-each>
</wdb:phenomenonTime>
<wdb:resultTime>
  <xsl:for-each select="$var34_Recnum">
   <xsl:attribute name="gml:id" select="fn:concat(fn:concat('LD.MON.', xs:string(xs:integer(fn:string(.)))), '.ot')"/>
  </xsl:for-each>
  <xsl:for-each select="$var33_Datum">
   <xsl:variable name="var49_cur" as="node()" select="."/>
   <xsl:for-each select="$var57_cur/Tijd">
    <xsl:variable name="var43_resultof_cast" as="xs:string" select="fn:string($var49_cur)"/>
    <xsl:variable name="var44_resultof_substring_after" as="xs:string" select="fn:substring-after($var43_resultof_cast, '-')"/>
    <xsl:variable name="var45_resultof_substring_before" as="xs:string"
                  select="fn:substring-before($var43_resultof_cast, '-')"/>
    <xsl:variable name="var46_resultof_substring_before" as="xs:string"
                  select="fn:substring-before($var44_resultof_substring_after, '-')"/>
    <xsl:variable name="var47_result" as="xs:string">
     <xsl:choose>
      <xsl:when test="(fn:string-length($var46_resultof_substring_before) = xs:decimal('1'))">
       <xsl:sequence select="fn:concat('0', $var46_resultof_substring_before)"/>
      </xsl:when>
      <xsl:otherwise>                <xsl:sequence select="$var46_resultof_substring_before"/>                </xsl:otherwise>
     </xsl:choose>
    </xsl:variable>
    <xsl:variable name="var48_result" as="xs:string">
     <xsl:choose>
      <xsl:when test="(fn:string-length($var45_resultof_substring_before) = xs:decimal('1'))">
       <xsl:sequence select="fn:concat('0', $var45_resultof_substring_before)"/>
      </xsl:when>
      <xsl:otherwise>                <xsl:sequence select="$var45_resultof_substring_before"/>                </xsl:otherwise>
     </xsl:choose>
    </xsl:variable>
    <gml:timePosition>
     <xsl:sequence select="fn:concat(fn:concat(fn:concat(fn:concat(fn:concat(fn:concat(fn:substring-before(
                           fn:substring-after($var44_resultof_substring_after, '-'), ' '), '-'), $var47_result), '-'), $var48_result), 'T'),
                           fn:substring-after(fn:string(.), ' '))"/>
    </gml:timePosition>
   </xsl:for-each>
  </xsl:for-each>
</wdb:resultTime>
<wdb:procedure>
  <xsl:sequence select="$var32_resultof_create_attribute"/>
  <xsl:attribute name="xlink:href" select="xs:string(xs:anyURI('#unknown'))"/>
</wdb:procedure>
<wdb:result>
  <xsl:attribute name="xsi:type" select="xs:QName('wdb:AnalyticalResult')"/>
  <xsl:for-each select="$var34_Recnum">
   <xsl:attribute name="gml:id" select="fn:concat(fn:concat('LD.MON.', xs:string(xs:integer(fn:string(.)))), 'om')"/>
  </xsl:for-each>
  <xsl:for-each select="Lab">
   <gml:description>              <xsl:sequence select="fn:string(.)"/>              </gml:description>
  </xsl:for-each>
  <xsl:for-each select="Waarde">
   <wdb:numericValue>
    <xsl:for-each select="$var35_Parnr">
     <xsl:variable name="var53_cur" as="node()" select="."/>
     <xsl:for-each select="$var4_Import/Row">
      <xsl:variable name="var52_cur" as="node()" select="."/>
      <xsl:for-each select="Parnr[(xs:integer(fn:string($var53_cur)) = xs:integer(fn:string(.))) and
                            fn:exists($var52_cur/EenheidOw))]">
       <xsl:attribute name="uom">
        <xsl:if test="(xs:integer(fn:string($var53_cur)) = xs:integer(fn:string(.)))">
         <xsl:for-each select="$var52_cur/EenheidOw">
          <xsl:variable name="var50_resultof_vmf__inputtoresult" as="xs:string">
           <xsl:call-template name="vmf:vmf1_inputtoresult">
            <xsl:with-param name="input" select="fn:string(.)" as="xs:string"/>
           </xsl:call-template>
          </xsl:variable>
          <xsl:variable name="var51_result" as="xs:string">
```

```
                <xsl:choose>
                  <xsl:when test="('NVT' = $var50_resultof_vmf__inputtoresult)">
                    <xsl:sequence select="fn:string(.)"/>
                  </xsl:when>
                  <xsl:otherwise>
                    <xsl:call-template name="vmf:vmf1_inputtoresult">
                      <xsl:with-param name="input" select="fn:string(.)" as="xs:string"/>
                    </xsl:call-template>
                  </xsl:otherwise>
                </xsl:choose>
              </xsl:variable>
              <xsl:sequence select="fn:concat('urn:aquo:eenheid:', $var51_result)"/>
            </xsl:for-each>
          </xsl:if>
        </xsl:attribute>
      </xsl:for-each>
    </xsl:for-each>
    <xsl:sequence select="xs:string(xs:double(fn:string(.)))"/>
  </wdb:numericValue>
</xsl:for-each>
<wdb:relatedObservationType>
  <xsl:sequence select="$var32_resultof_create_attribute"/>
  <xsl:for-each select="$var35_Parnr">
    <xsl:variable name="var56_cur" as="node()" select="."/>
    <xsl:for-each select="$var4_Import/Row">
      <xsl:variable name="var55_cur" as="node()" select="."/>
      <xsl:for-each select="Parnr[(xs:integer(fn:string($var56_cur)) = xs:integer(fn:string(.))) and
                                  fn:exists($var55_cur/Recnum))]">
        <xsl:variable name="var54_result" as="xs:string">
          <xsl:if test="(xs:integer(fn:string($var56_cur)) = xs:integer(fn:string(.)))">
            <xsl:for-each select="$var55_cur/Recnum">
              <xsl:sequence select="fn:concat('#', fn:concat('_LD.PAR.', xs:string(xs:integer(fn:string(.)))))"/>
            </xsl:for-each>
          </xsl:if>
        </xsl:variable>
        <xsl:attribute name="xlink:href" select="xs:string(xs:anyURI($var54_result))"/>
      </xsl:for-each>
    </xsl:for-each>
  </xsl:for-each>
</wdb:relatedObservationType>
</wdb:result>
</wdb:OM_Observation>
</xsl:for-each>
<xsl:for-each select="$var2_Import/Row">
<xsl:variable name="var80_cur" as="node()" select="."/>
<xsl:variable name="var58_resultof_create_attribute" as="node()">
  <xsl:attribute name="xlink:type" select="'simple'"/>
</xsl:variable>
<xsl:variable name="var59_WNSID" as="node()?" select="WNS_ID"/>
<xsl:variable name="var60_MWAID" as="node()?" select="MWA_ID"/>
<xsl:variable name="var61_MWADTMB" as="node()?" select="MWADTMB"/>
<wdb:OM_Observation>
  <xsl:for-each select="$var60_MWAID">
    <xsl:attribute name="gml:id" select="fn:concat('BULK.MWA.', xs:string(xs:integer(fn:string(.))))"/>
  </xsl:for-each>
  <wdb:phenomenonTime>
    <xsl:attribute name="xsi:type" select="xs:QName('gml:TimeInstantType')"/>
    <xsl:for-each select="$var60_MWAID">
      <xsl:attribute name="gml:id" select="fn:concat(fn:concat('BULK.MWA.', xs:string(xs:integer(fn:string(.)))), 'pt')"/>
    </xsl:for-each>
    <xsl:for-each select="$var61_MWADTMB">
      <xsl:variable name="var68_cur" as="node()" select="."/>
      <xsl:for-each select="$var80_cur/MWATIJDB">
        <xsl:variable name="var62_resultof_cast" as="xs:string" select="fn:string($var68_cur)"/>
        <xsl:variable name="var63_resultof_substring_after" as="xs:string" select="fn:substring-after($var62_resultof_cast, '-')"/>
        <xsl:variable name="var64_resultof_substring_before" as="xs:string"
                      select="fn:substring-before($var62_resultof_cast, '-')"/>
        <xsl:variable name="var65_resultof_substring_before" as="xs:string"
                      select="fn:substring-before($var63_resultof_substring_after, '-')"/>
        <xsl:variable name="var66_result" as="xs:string">
          <xsl:choose>
            <xsl:when test="(fn:string-length($var65_resultof_substring_before) = xs:decimal('1'))">
              <xsl:sequence select="fn:concat('0', $var65_resultof_substring_before)"/>
            </xsl:when>
            <xsl:otherwise>                <xsl:sequence select="$var65_resultof_substring_before"/>                </xsl:otherwise>
```

```
      </xsl:choose>
    </xsl:variable>
    <xsl:variable name="var67_result" as="xs:string">
      <xsl:choose>
        <xsl:when test="(fn:string-length($var64_resultof_substring_before) = xs:decimal('1'))">
          <xsl:sequence select="fn:concat('0', $var64_resultof_substring_before)"/>
        </xsl:when>
        <xsl:otherwise>                <xsl:sequence select="$var64_resultof_substring_before"/>                </xsl:otherwise>
      </xsl:choose>
    </xsl:variable>
    <gml:timePosition>
      <xsl:sequence select="fn:concat(fn:concat(fn:concat(fn:concat(fn:concat(fn:concat(fn:substring-before(
                fn:substring-after($var63_resultof_substring_after, '-'), ' '), '-'), $var66_result), '-'), $var67_result), 'T'),
                fn:substring-after(fn:string(.), ' '))"/>
    </gml:timePosition>
   </xsl:for-each>
  </xsl:for-each>
</wdb:phenomenonTime>
<wdb:resultTime>
  <xsl:for-each select="$var60_MWAID">
   <xsl:attribute name="gml:id" select="fn:concat(fn:concat('BULK.MWA.', xs:string(xs:integer(fn:string(.)))), '.ot')"/>
  </xsl:for-each>
  <xsl:for-each select="$var61_MWADTMB">
   <xsl:variable name="var75_cur" as="node()" select="."/>
   <xsl:for-each select="$var80_cur/MWATIJDB">
    <xsl:variable name="var69_resultof_cast" as="xs:string" select="fn:string($var75_cur)"/>
    <xsl:variable name="var70_resultof_substring_after" as="xs:string" select="fn:substring-after($var69_resultof_cast, '-')"/>
    <xsl:variable name="var71_resultof_substring_before" as="xs:string"
                  select="fn:substring-before($var69_resultof_cast, '-')"/>
    <xsl:variable name="var72_resultof_substring_before" as="xs:string"
                  select="fn:substring-before($var70_resultof_substring_after, '-')"/>
    <xsl:variable name="var73_result" as="xs:string">
      <xsl:choose>
        <xsl:when test="(fn:string-length($var72_resultof_substring_before) = xs:decimal('1'))">
          <xsl:sequence select="fn:concat('0', $var72_resultof_substring_before)"/>
        </xsl:when>
        <xsl:otherwise>                <xsl:sequence select="$var72_resultof_substring_before"/>                </xsl:otherwise>
      </xsl:choose>
    </xsl:variable>
    <xsl:variable name="var74_result" as="xs:string">
      <xsl:choose>
        <xsl:when test="(fn:string-length($var71_resultof_substring_before) = xs:decimal('1'))">
          <xsl:sequence select="fn:concat('0', $var71_resultof_substring_before)"/>
        </xsl:when>
        <xsl:otherwise>                <xsl:sequence select="$var71_resultof_substring_before"/>                </xsl:otherwise>
      </xsl:choose>
    </xsl:variable>
    <gml:timePosition>
      <xsl:sequence select="fn:concat(fn:concat(fn:concat(fn:concat(fn:concat(fn:concat(fn:substring-before(
                fn:substring-after($var70_resultof_substring_after, '-'), ' '), '-'), $var73_result), '-'), $var74_result), 'T'),
                fn:substring-after(fn:string(.), ' '))"/>
    </gml:timePosition>
   </xsl:for-each>
  </xsl:for-each>
</wdb:resultTime>
<wdb:procedure>
  <xsl:sequence select="$var58_resultof_create_attribute"/>
  <xsl:attribute name="xlink:href" select="xs:string(xs:anyURI('#unknown'))"/>
</wdb:procedure>
<wdb:result>
  <xsl:attribute name="xsi:type" select="xs:QName('wdb:AnalyticalResult')"/>
  <xsl:for-each select="$var60_MWAID">
   <xsl:attribute name="gml:id" select="fn:concat(fn:concat('BULK.MWA.', xs:string(xs:integer(fn:string(.)))), 'om')"/>
  </xsl:for-each>
  <xsl:for-each select="MWAWRDEA">
   <gml:description>            <xsl:sequence select="fn:string(.)"/>            </gml:description>
  </xsl:for-each>
  <xsl:for-each select="MRSINOVS_ID">
   <xsl:variable name="var76_resultof_vmf__inputtoresult" as="xs:string?">
    <xsl:call-template name="vmf:vmf2_inputtoresult">
     <xsl:with-param name="input" select="xs:integer(fn:string(.))" as="xs:integer"/>
    </xsl:call-template>
   </xsl:variable>
   <xsl:if test="fn:exists($var76_resultof_vmf__inputtoresult)">
    <wdb:limitSymbol>            <xsl:sequence select="$var76_resultof_vmf__inputtoresult"/>            </wdb:limitSymbol>
   </xsl:if>
```

```
    </xsl:for-each>
    <xsl:for-each select="MWAWRDEN">
     <wdb:numericValue>
      <xsl:for-each select="$var59_WNSID">
       <xsl:variable name="var78_cur" as="node()" select="."/>
       <xsl:for-each select="$var5_Import/Row">
        <xsl:variable name="var77_cur" as="node()" select="."/>
        <xsl:for-each select="WNS_ID[((xs:integer(fn:string($var78_cur)) = xs:integer(fn:string(.))) and
                                      fn:exists($var77_cur/BV_MEP_DOMGWCOD))]">
         <xsl:attribute name="uom">
          <xsl:if test="(xs:integer(fn:string($var78_cur)) = xs:integer(fn:string(.)))">
           <xsl:for-each select="$var77_cur/BV_MEP_DOMGWCOD">
            <xsl:sequence select="fn:concat('urn:bulk:bv_mep:domgwcod:', fn:string(.))"/>
           </xsl:for-each>
          </xsl:if>
         </xsl:attribute>
        </xsl:for-each>
       </xsl:for-each>
      </xsl:for-each>
      <xsl:sequence select="xs:string(xs:double(fn:string(.)))"/>
     </wdb:numericValue>
    </xsl:for-each>
    <xsl:for-each select="MRSINKWA_ID">
     <wdb:qualityIndicator>
      <xsl:attribute name="codeSpace" select="xs:string(xs:anyURI('urn:bulk:bv_mrsinkwa:id'))"/>
      <xsl:sequence select="xs:string(xs:integer(fn:string(.)))"/>
     </wdb:qualityIndicator>
    </xsl:for-each>
    <wdb:relatedObservationType>
     <xsl:sequence select="$var58_resultof_create_attribute"/>
     <xsl:for-each select="$var59_WNSID">
      <xsl:variable name="var79_cur" as="node()" select="."/>
      <xsl:for-each select="$var5_Import/Row/WNS_ID[(xs:integer(fn:string($var79_cur)) = xs:integer(fn:string(.)))]">
       <xsl:attribute name="xlink:href" select="xs:string(xs:anyURI(fn:concat('#',
                                    fn:concat('_BK.WNS.', xs:string(xs:integer(fn:string(.)))))))"/>
      </xsl:for-each>
     </xsl:for-each>
    </wdb:relatedObservationType>
   </wdb:result>
  </wdb:OM_Observation>
</xsl:for-each>
<xsl:for-each select="$var4_Import/Row">
 <xsl:variable name="var105_cur" as="node()" select="."/>
 <xsl:variable name="var81_Naam" as="node()?" select="Naam"/>
 <wdb:ObservationType>
  <xsl:for-each select="Recnum">
   <xsl:attribute name="gml:id" select="fn:concat('_LD.PAR.', xs:string(xs:integer(fn:string(.))))"/>
  </xsl:for-each>
  <xsl:for-each select="$var81_Naam">
   <xsl:variable name="var82_resultof_vmf__inputtoresult" as="xs:string">
    <xsl:call-template name="vmf:vmf3_inputtoresult">
     <xsl:with-param name="input" select="fn:string(.)" as="xs:string"/>
    </xsl:call-template>
   </xsl:variable>
   <xsl:variable name="var83_resultof_equal" as="xs:boolean" select="('NVT' = $var82_resultof_vmf__inputtoresult)"/>
   <xsl:variable name="var91_result" as="xs:boolean">
    <xsl:choose>
     <xsl:when test="$var83_resultof_equal">
      <xsl:variable name="var87_result" as="xs:boolean?">
       <xsl:for-each select="$var105_cur/EenheidOw">
        <xsl:variable name="var84_resultof_cast" as="xs:string" select="fn:string(.)"/>
        <xsl:variable name="var85_resultof_vmf__inputtoresult" as="xs:string">
         <xsl:call-template name="vmf:vmf1_inputtoresult">
          <xsl:with-param name="input" select="$var84_resultof_cast" as="xs:string"/>
         </xsl:call-template>
        </xsl:variable>
        <xsl:variable name="var86_resultof_vmf__inputtoresult" as="xs:string?">
         <xsl:call-template name="vmf:vmf4_inputtoresult">
          <xsl:with-param name="input" as="xs:string">
           <xsl:choose>
            <xsl:when test="('NVT' = $var85_resultof_vmf__inputtoresult)">
             <xsl:sequence select="$var84_resultof_cast"/>
            </xsl:when>
            <xsl:otherwise><xsl:sequence select="$var85_resultof_vmf__inputtoresult"/></xsl:otherwise>
           </xsl:choose>
          </xsl:with-param>
```

```
          </xsl:call-template>
        </xsl:variable>
        <xsl:sequence select="fn:exists($var86_resultof_vmf__inputtoresult)"/>
      </xsl:for-each>
    </xsl:variable>
    <xsl:sequence select="fn:exists($var87_result[.])"/>
  </xsl:when>
  <xsl:otherwise>  <xsl:sequence select="fn:true()"/>  </xsl:otherwise>
 </xsl:choose>
</xsl:variable>
<xsl:if test="$var91_result">
 <wdb:quanitity>
  <xsl:attribute name="codeSpace" select="xs:string(xs:anyURI('urn:aquo:grootheid:code'))"/>
  <xsl:choose>
   <xsl:when test="$var83_resultof_equal">
    <xsl:for-each select="$var105_cur/EenheidOw">
     <xsl:variable name="var88_resultof_cast" as="xs:string" select="fn:string(.)"/>
     <xsl:variable name="var89_resultof_vmf__inputtoresult" as="xs:string">
      <xsl:call-template name="vmf:vmf1_inputtoresult">
       <xsl:with-param name="input" select="$var88_resultof_cast" as="xs:string"/>
      </xsl:call-template>
     </xsl:variable>
     <xsl:variable name="var90_resultof_vmf__inputtoresult" as="xs:string?">
      <xsl:call-template name="vmf:vmf4_inputtoresult">
       <xsl:with-param name="input" as="xs:string">
        <xsl:choose>
         <xsl:when test="('NVT' = $var89_resultof_vmf__inputtoresult)">
          <xsl:sequence select="$var88_resultof_cast"/>
         </xsl:when>
         <xsl:otherwise><xsl:sequence select="$var89_resultof_vmf__inputtoresult"/></xsl:otherwise>
        </xsl:choose>
       </xsl:with-param>
      </xsl:call-template>
     </xsl:variable>
     <xsl:if test="fn:exists($var90_resultof_vmf__inputtoresult)">
      <xsl:sequence select="$var90_resultof_vmf__inputtoresult"/>
     </xsl:if>
    </xsl:for-each>
   </xsl:when>
   <xsl:otherwise>  <xsl:sequence select="$var82_resultof_vmf__inputtoresult"/>  </xsl:otherwise>
  </xsl:choose>
 </wdb:quanitity>
</xsl:if>
</xsl:for-each>
<xsl:for-each select="$var81_Naam">
 <xsl:variable name="var92_resultof_cast" as="xs:string" select="fn:string(.)"/>
 <xsl:variable name="var93_resultof_vmf__inputtoresult" as="xs:string">
  <xsl:call-template name="vmf:vmf3_inputtoresult">
   <xsl:with-param name="input" select="$var92_resultof_cast" as="xs:string"/>
  </xsl:call-template>
 </xsl:variable>
 <xsl:if test="('NVT' = $var93_resultof_vmf__inputtoresult)">
  <xsl:variable name="var94_resultof_vmf__inputtoresult" as="xs:string">
   <xsl:call-template name="vmf:vmf5_inputtoresult">
    <xsl:with-param name="input" select="$var92_resultof_cast" as="xs:string"/>
   </xsl:call-template>
  </xsl:variable>
  <xsl:variable name="var95_resultof_not_equal" as="xs:boolean" select="('NVT' != $var94_resultof_vmf__inputtoresult)"/>
  <wdb:parameter>
   <xsl:variable name="var96_result" as="xs:string">
    <xsl:choose>
     <xsl:when test="$var95_resultof_not_equal">
      <xsl:sequence select="'urn:aquo:object:code'"/>
     </xsl:when>
     <xsl:otherwise>  <xsl:sequence select="'urn:aquo:chemischeStof:code'"/>  </xsl:otherwise>
    </xsl:choose>
   </xsl:variable>
   <xsl:attribute name="codeSpace" select="xs:string(xs:anyURI($var96_result))"/>
   <xsl:choose>
    <xsl:when test="$var95_resultof_not_equal"><xsl:sequence select="$var94_resultof_vmf__inputtoresult"/></xsl:when>
    <xsl:otherwise>
     <xsl:variable name="var97_resultof_vmf__inputtoresult" as="xs:string">
      <xsl:call-template name="vmf:vmf6_inputtoresult">
       <xsl:with-param name="input" select="$var92_resultof_cast" as="xs:string"/>
      </xsl:call-template>
     </xsl:variable>
```

```
          <xsl:choose>
            <xsl:when test="('NVT' = $var97_resultof_vmf__inputtoresult)">
              <xsl:sequence select="$var92_resultof_cast"/>
            </xsl:when>
            <xsl:otherwise><xsl:sequence select="$var97_resultof_vmf__inputtoresult"/>                    </xsl:otherwise>
          </xsl:choose>
        </xsl:otherwise>
      </xsl:choose>
    </wdb:parameter>
  </xsl:if>
 </xsl:for-each>
 <xsl:for-each select="$var81_Naam">
  <xsl:variable name="var98_resultof_cast" as="xs:string" select="fn:string(.)"/>
  <xsl:variable name="var99_resultof_vmf__inputtoresult" as="xs:string">
   <xsl:call-template name="vmf:vmf7_inputtoresult">
    <xsl:with-param name="input" select="$var98_resultof_cast" as="xs:string"/>
   </xsl:call-template>
  </xsl:variable>
  <xsl:variable name="var100_resultof_not_equal" as="xs:boolean" select="('NVT' != $var99_resultof_vmf__inputtoresult)"/>
  <xsl:variable name="var104_result" as="xs:boolean">
   <xsl:choose>
    <xsl:when test="$var100_resultof_not_equal">              <xsl:sequence select="fn:true()"/>              </xsl:when>
    <xsl:otherwise>
     <xsl:variable name="var102_result" as="xs:boolean?">
      <xsl:for-each select="$var105_cur/EenheidOw">
       <xsl:variable name="var101_resultof_vmf__inputtoresult" as="xs:string">
        <xsl:call-template name="vmf:vmf8_inputtoresult">
         <xsl:with-param name="input" select="fn:string(.)" as="xs:string"/>
        </xsl:call-template>
       </xsl:variable>
       <xsl:sequence select="(('NVT' != $var101_resultof_vmf__inputtoresult) or matches($var98_resultof_cast, 'filtraat'))"/>
      </xsl:for-each>
     </xsl:variable>
     <xsl:sequence select="fn:exists($var102_result[.])"/>
    </xsl:otherwise>
   </xsl:choose>
  </xsl:variable>
  <xsl:if test="$var104_result">
   <wdb:condition>
    <xsl:attribute name="codeSpace" select="xs:string(xs:anyURI('urn:aquo:hoedanigheid'))"/>
    <xsl:choose>
     <xsl:when test="$var100_resultof_not_equal">
      <xsl:sequence select="$var99_resultof_vmf__inputtoresult"/>
     </xsl:when>
     <xsl:otherwise>
      <xsl:for-each select="$var105_cur/EenheidOw">
       <xsl:variable name="var103_resultof_vmf__inputtoresult" as="xs:string">
        <xsl:call-template name="vmf:vmf8_inputtoresult">
         <xsl:with-param name="input" select="fn:string(.)" as="xs:string"/>
        </xsl:call-template>
       </xsl:variable>
       <xsl:choose>
        <xsl:when test="('NVT' != $var103_resultof_vmf__inputtoresult)">
         <xsl:sequence select="$var103_resultof_vmf__inputtoresult"/>
        </xsl:when>
        <xsl:otherwise>
         <xsl:if test="matches($var98_resultof_cast, 'filtraat')">
          <xsl:sequence select="'nf'"/>
         </xsl:if>
        </xsl:otherwise>
       </xsl:choose>
      </xsl:for-each>
     </xsl:otherwise>
    </xsl:choose>
   </wdb:condition>
  </xsl:if>
 </xsl:for-each>
</wdb:ObservationType>
</xsl:for-each>
<xsl:for-each select="$var5_Import/Row">
 <xsl:variable name="var124_cur" as="node()" select="."/>
 <xsl:variable name="var106_BVMPSDOMGWCOD" as="node()?" select="BV_MPS_DOMGWCOD"/>
 <wdb:ObservationType>
  <xsl:for-each select="WNS_ID">
   <xsl:attribute name="gml:id" select="fn:concat('_BK.WNS.', xs:string(xs:integer(fn:string(.))))"/>
  </xsl:for-each>
```

```
<xsl:for-each select="$var106_BVMPSDOMGWCOD">
 <xsl:variable name="var107_resultof_vmf__inputtoresult" as="xs:string">
  <xsl:call-template name="vmf:vmf9_inputtoresult">
   <xsl:with-param name="input" select="fn:string(.)" as="xs:string"/>
  </xsl:call-template>
 </xsl:variable>
 <xsl:variable name="var108_resultof_equal" as="xs:boolean" select="('NVT' = $var107_resultof_vmf__inputtoresult)"/>
 <xsl:variable name="var112_result" as="xs:boolean">
  <xsl:choose>
   <xsl:when test="$var108_resultof_equal">
    <xsl:variable name="var110_result" as="xs:boolean?">
     <xsl:for-each select="$var124_cur/BV_MEP_DOMGWCOD">
      <xsl:variable name="var109_resultof_vmf___inputtoresult" as="xs:string?">
       <xsl:call-template name="vmf:vmf10_inputtoresult">
        <xsl:with-param name="input" select="fn:string(.)" as="xs:string"/>
       </xsl:call-template>
      </xsl:variable>
      <xsl:sequence select="fn:exists($var109_resultof_vmf___inputtoresult)"/>
     </xsl:for-each>
    </xsl:variable>
    <xsl:sequence select="fn:exists($var110_result[.])"/>
   </xsl:when>
   <xsl:otherwise>  <xsl:sequence select="fn:true()"/>  </xsl:otherwise>
  </xsl:choose>
 </xsl:variable>
 <xsl:if test="$var112_result">
  <wdb:quanitity>
   <xsl:attribute name="codeSpace" select="xs:string(xs:anyURI('urn:aquo:grootheid:code'))"/>
   <xsl:choose>
    <xsl:when test="$var108_resultof_equal">
     <xsl:for-each select="$var124_cur/BV_MEP_DOMGWCOD">
      <xsl:variable name="var111_resultof_vmf___inputtoresult" as="xs:string?">
       <xsl:call-template name="vmf:vmf10_inputtoresult">
        <xsl:with-param name="input" select="fn:string(.)" as="xs:string"/>
       </xsl:call-template>
      </xsl:variable>
      <xsl:if test="fn:exists($var111_resultof_vmf___inputtoresult)">
       <xsl:sequence select="$var111_resultof_vmf___inputtoresult"/>
      </xsl:if>
     </xsl:for-each>
    </xsl:when>
    <xsl:otherwise>  <xsl:sequence select="$var107_resultof_vmf__inputtoresult"/>  </xsl:otherwise>
   </xsl:choose>
  </wdb:quanitity>
 </xsl:if>
</xsl:for-each>
<xsl:for-each select="$var106_BVMPSDOMGWCOD">
 <xsl:variable name="var113_resultof_cast" as="xs:string" select="fn:string(.)"/>
 <xsl:variable name="var114_resultof_vmf__inputtoresult" as="xs:string">
  <xsl:call-template name="vmf:vmf9_inputtoresult">
   <xsl:with-param name="input" select="$var113_resultof_cast" as="xs:string"/>
  </xsl:call-template>
 </xsl:variable>
 <xsl:if test="('NVT' = $var114_resultof_vmf__inputtoresult)">
  <xsl:variable name="var115_resultof_vmf___inputtoresult" as="xs:string">
   <xsl:call-template name="vmf:vmf11_inputtoresult">
    <xsl:with-param name="input" select="$var113_resultof_cast" as="xs:string"/>
   </xsl:call-template>
  </xsl:variable>
  <xsl:variable name="var116_resultof_not_equal" as="xs:boolean"
                select="($var115_resultof_vmf___inputtoresult != 'NVT')"/>
  <wdb:parameter>
   <xsl:variable name="var117_result" as="xs:string">
    <xsl:choose>
     <xsl:when test="$var116_resultof_not_equal">
      <xsl:sequence select="'urn:aquo:object:code'"/>
     </xsl:when>
     <xsl:otherwise>  <xsl:sequence select="'urn:aquo:chemischeStof:code'"/>  </xsl:otherwise>
    </xsl:choose>
   </xsl:variable>
   <xsl:attribute name="codeSpace" select="xs:string(xs:anyURI($var117_result))"/>
   <xsl:choose>
    <xsl:when test="$var116_resultof_not_equal">
     <xsl:sequence select="$var115_resultof_vmf___inputtoresult"/>
    </xsl:when>
    <xsl:otherwise>
```

```
          <xsl:variable name="var118_resultof_vmf___inputtoresult" as="xs:string">
            <xsl:call-template name="vmf:vmf12_inputtoresult">
              <xsl:with-param name="input" select="$var113_resultof_cast" as="xs:string"/>
            </xsl:call-template>
          </xsl:variable>
          <xsl:choose>
            <xsl:when test="('NVT' = $var118_resultof_vmf___inputtoresult)">
              <xsl:sequence select="$var113_resultof_cast"/>
            </xsl:when>
            <xsl:otherwise><xsl:sequence select="$var118_resultof_vmf___inputtoresult"/></xsl:otherwise>
          </xsl:choose>
        </xsl:otherwise>
      </xsl:choose>
    </wdb:parameter>
  </xsl:if>
 </xsl:for-each>
 <xsl:for-each select="$var106_BVMPSDOMGWCOD">
  <xsl:variable name="var119_resultof_vmf___inputtoresult" as="xs:string">
    <xsl:call-template name="vmf:vmf13_inputtoresult">
      <xsl:with-param name="input" select="fn:string(.)" as="xs:string"/>
    </xsl:call-template>
  </xsl:variable>
  <xsl:variable name="var120_resultof_equal" as="xs:boolean" select="('NVT' = $var119_resultof_vmf___inputtoresult)"/>
  <xsl:variable name="var123_result" as="xs:boolean">
    <xsl:choose>
      <xsl:when test="$var120_resultof_equal">
        <xsl:variable name="var121_result" as="xs:boolean?">
          <xsl:for-each select="$var124_cur/BV_HOE_DOMGWCOD">
            <xsl:sequence select="fn:not(('NVT' = fn:string(.)))"/>
          </xsl:for-each>
        </xsl:variable>
        <xsl:sequence select="fn:exists($var121_result[.])"/>
      </xsl:when>
      <xsl:otherwise>          <xsl:sequence select="fn:true()"/>          </xsl:otherwise>
    </xsl:choose>
  </xsl:variable>
  <xsl:if test="$var123_result">
    <wdb:condition>
      <xsl:attribute name="codeSpace" select="xs:string(xs:anyURI('urn:bulk:bv_hoe:domgwcod'))"/>
      <xsl:choose>
        <xsl:when test="$var120_resultof_equal">
          <xsl:for-each select="$var124_cur/BV_HOE_DOMGWCOD">
            <xsl:variable name="var122_resultof_cast" as="xs:string" select="fn:string(.)"/>
            <xsl:if test="fn:not(('NVT' = $var122_resultof_cast))">
              <xsl:sequence select="$var122_resultof_cast"/>
            </xsl:if>
          </xsl:for-each>
        </xsl:when>
        <xsl:otherwise><xsl:sequence select="$var119_resultof_vmf___inputtoresult"/>          </xsl:otherwise>
      </xsl:choose>
    </wdb:condition>
  </xsl:if>
 </xsl:for-each>
</wdb:ObservationType>
</xsl:for-each>
<xsl:for-each select="$var1_FeatureCollection/gml:featureMembers/wdb:ObservingCapability">
 <xsl:variable name="var125_resultof_create_attribute" as="node()">
  <xsl:attribute name="xsi:nil" select="'true'"/>
 </xsl:variable>
 <xsl:variable name="var126_procedure" as="node()?" select="wdb:procedure"/>
 <wdb:ObservingCapability>
  <xsl:attribute name="gml:id" select="fn:string(@gml:id)"/>
  <xsl:for-each select="gml:metaDataProperty">
   <gml:metaDataProperty>          <xsl:sequence select="(./@node(), ./node())"/>          </gml:metaDataProperty>
  </xsl:for-each>
  <xsl:for-each select="gml:description">
   <gml:description>          <xsl:sequence select="(./@node(), ./node())"/>          </gml:description>
  </xsl:for-each>
  <xsl:for-each select="gml:descriptionReference">
   <gml:descriptionReference>          <xsl:sequence select="(./@node(), ./node())"/>          </gml:descriptionReference>
  </xsl:for-each>
  <xsl:for-each select="gml:identifier">
   <gml:identifier>          <xsl:sequence select="(./@node(), ./node())"/>          </gml:identifier>
  </xsl:for-each>
  <xsl:for-each select="gml:name">
   <gml:name>          <xsl:sequence select="(./@node(), ./node())"/>          </gml:name>
```

```
        </xsl:for-each>
        <xsl:for-each select="gml:boundedBy">
          <gml:boundedBy>            <xsl:sequence select="(./@node(), ./node())"/>          </gml:boundedBy>
        </xsl:for-each>
        <xsl:for-each select="gml:location">
          <gml:location>            <xsl:sequence select="(./@node(), ./node())"/>          </gml:location>
        </xsl:for-each>
        <xsl:for-each select="gml:priorityLocation">
          <gml:priorityLocation>            <xsl:sequence select="(./@node(), ./node())"/>          </gml:priorityLocation>
        </xsl:for-each>
        <xsl:for-each select="wdb:observingTime">
          <xsl:variable name="var129_cur" as="node()" select="."/>
          <xsl:variable name="var127_result" as="xs:boolean*">
            <xsl:for-each select="@xsi:type">
              <xsl:sequence select="(fn:resolve-QName(fn:string(.), $var129_cur) = xs:QName('gml:TimeInstantType'))"/>
            </xsl:for-each>
          </xsl:variable>
          <xsl:if test="fn:exists($var127_result[.])">
            <xsl:variable name="var128_frame" as="node()?" select="@frame"/>
            <wdb:observingTime>
              <xsl:attribute name="xsi:type" select="xs:QName('gml:TimeInstantType')"/>
              <xsl:attribute name="gml:id" select="fn:string(@gml:id)"/>
              <xsl:if test="fn:exists($var128_frame)">
                <xsl:attribute name="frame" select="xs:string(xs:anyURI(fn:string($var128_frame)))"/>
              </xsl:if>
              <xsl:for-each select="gml:metaDataProperty">
                <gml:metaDataProperty>                  <xsl:sequence select="(./@node(), ./node())"/>
</gml:metaDataProperty>
              </xsl:for-each>
              <xsl:for-each select="gml:description">
                <gml:description>                  <xsl:sequence select="(./@node(), ./node())"/>                </gml:description>
              </xsl:for-each>
              <xsl:for-each select="gml:descriptionReference">
                <gml:descriptionReference> <xsl:sequence select="(./@node(), ./node())"/>                </gml:descriptionReference>
              </xsl:for-each>
              <xsl:for-each select="gml:identifier">
                <gml:identifier>                  <xsl:sequence select="(./@node(), ./node())"/>                </gml:identifier>
              </xsl:for-each>
              <xsl:for-each select="gml:name">
                <gml:name>                  <xsl:sequence select="(./@node(), ./node())"/>                </gml:name>
              </xsl:for-each>
              <xsl:for-each select="gml:relatedTime">
                <gml:relatedTime>                  <xsl:sequence select="(./@node(), ./node())"/>                </gml:relatedTime>
              </xsl:for-each>
              <xsl:for-each select="gml:timePosition">
                <gml:timePosition>                  <xsl:sequence select="(./@node(), ./node())"/>                </gml:timePosition>
              </xsl:for-each>
            </wdb:observingTime>
          </xsl:if>
        </xsl:for-each>
        <xsl:for-each select="wdb:monitoredParameter">
          <wdb:monitoredParameter>
            <xsl:for-each select="wdb:quantity">
              <wdb:quantity>                <xsl:sequence select="(./@node(), ./node())"/>              </wdb:quantity>
            </xsl:for-each>
            <xsl:for-each select="wdb:parameter">
              <wdb:parameter>                <xsl:sequence select="(./@node(), ./node())"/>              </wdb:parameter>
            </xsl:for-each>
            <xsl:for-each select="wdb:resultNature">
              <wdb:resultNature>                <xsl:sequence select="fn:string(.)"/>              </wdb:resultNature>
            </xsl:for-each>
            <xsl:for-each select="wdb:frequency">
              <wdb:frequency>                <xsl:sequence select="(./@node(), ./node())"/>              </wdb:frequency>
            </xsl:for-each>
            <xsl:for-each select="wdb:frequencyDescription">
              <wdb:frequencyDescription>
                <xsl:choose>
                  <xsl:when test="(fn:translate(fn:string(@xsi:nil), 'true ', '1') = '1')">
                    <xsl:attribute name="xsi:nil" select="'true'"/>
                  </xsl:when>
                  <xsl:otherwise>                    <xsl:sequence select="fn:string(.)"/>                  </xsl:otherwise>
                </xsl:choose>
              </wdb:frequencyDescription>
            </xsl:for-each>
            <xsl:for-each select="wdb:cycle">
              <wdb:cycle>                <xsl:sequence select="(./@node(), ./node())"/>              </wdb:cycle>
```

```
  </xsl:for-each>
  <xsl:for-each select="wdb:cycleDescription">
   <wdb:cycleDescription>
    <xsl:choose>
     <xsl:when test="(fn:translate(fn:string(@xsi:nil), 'true ', '1') = '1')">
      <xsl:attribute name="xsi:nil" select="'true'"/>
     </xsl:when>
     <xsl:otherwise>                <xsl:sequence select="fn:string(.)"/>                </xsl:otherwise>
    </xsl:choose>
   </wdb:cycleDescription>
  </xsl:for-each>
  <xsl:for-each select="wdb:reasonDeviationProgram">
   <wdb:reasonDeviationProgram>
    <xsl:choose>
     <xsl:when test="(fn:translate(fn:string(@xsi:nil), 'true ', '1') = '1')">
      <xsl:attribute name="xsi:nil" select="'true'"/>
     </xsl:when>
     <xsl:otherwise>                <xsl:sequence select="fn:string(.)"/>                </xsl:otherwise>
    </xsl:choose>
   </wdb:reasonDeviationProgram>
  </xsl:for-each>
  <xsl:for-each select="wdb:parameterUse">
   <wdb:parameterUse>              <xsl:sequence select="(./@node(), ./node())"/>          </wdb:parameterUse>
  </xsl:for-each>
 </wdb:monitoredParameter>
</xsl:for-each>
<wdb:onlineResource>          <xsl:sequence select="$var125_resultof_create_attribute"/>        </wdb:onlineResource>
<wdb:procedure>
 <xsl:for-each select="$var126_procedure">
  <xsl:variable name="var130_type" as="node()?" select="@xlink:type"/>
  <xsl:if test="fn:exists($var130_type)">
   <xsl:attribute name="xlink:type" select="fn:string($var130_type)"/>
  </xsl:if>
 </xsl:for-each>
 <xsl:for-each select="$var126_procedure">
  <xsl:variable name="var131_href" as="node()?" select="@xlink:href"/>
  <xsl:if test="fn:exists($var131_href)">
   <xsl:attribute name="xlink:href" select="xs:string(xs:anyURI(fn:string($var131_href)))"/>
  </xsl:if>
 </xsl:for-each>
 <xsl:for-each select="$var126_procedure">
  <xsl:variable name="var132_role" as="node()?" select="@xlink:role"/>
  <xsl:if test="fn:exists($var132_role)">
   <xsl:attribute name="xlink:role" select="xs:string(xs:anyURI(fn:string($var132_role)))"/>
  </xsl:if>
 </xsl:for-each>
 <xsl:for-each select="$var126_procedure">
  <xsl:variable name="var133_arcrole" as="node()?" select="@xlink:arcrole"/>
  <xsl:if test="fn:exists($var133_arcrole)">
   <xsl:attribute name="xlink:arcrole" select="xs:string(xs:anyURI(fn:string($var133_arcrole)))"/>
  </xsl:if>
 </xsl:for-each>
 <xsl:for-each select="$var126_procedure">
  <xsl:variable name="var134_title" as="node()?" select="@xlink:title"/>
  <xsl:if test="fn:exists($var134_title)"><xsl:attribute name="xlink:title" select="fn:string($var134_title)"/>          </xsl:if>
 </xsl:for-each>
 <xsl:for-each select="$var126_procedure">
  <xsl:variable name="var135_show" as="node()?" select="@xlink:show"/>
  <xsl:if test="fn:exists($var135_show)"><xsl:attribute name="xlink:show" select="fn:string($var135_show)"/> </xsl:if>
 </xsl:for-each>
 <xsl:for-each select="$var126_procedure">
  <xsl:variable name="var136_actuate" as="node()?" select="@xlink:actuate"/>
  <xsl:if test="fn:exists($var136_actuate)">
   <xsl:attribute name="xlink:actuate" select="fn:string($var136_actuate)"/>
  </xsl:if>
 </xsl:for-each>
 <xsl:for-each select="$var126_procedure">
  <xsl:variable name="var137_nilReason" as="node()?" select="@nilReason"/>
  <xsl:if test="fn:exists($var137_nilReason)">
   <xsl:attribute name="nilReason" select="fn:string($var137_nilReason)"/>
  </xsl:if>
 </xsl:for-each>
 <xsl:for-each select="$var126_procedure">
  <xsl:variable name="var138_remoteSchema" as="node()?" select="@gml:remoteSchema"/>
  <xsl:if test="fn:exists($var138_remoteSchema)">
   <xsl:attribute name="gml:remoteSchema" select="xs:string(xs:anyURI(fn:string($var138_remoteSchema)))"/>
```

```
    </xsl:if>
   </xsl:for-each>
   <xsl:sequence select="$var125_resultof_create_attribute"/>
  </wdb:procedure>
  <xsl:for-each select="wdb:featureOfInterest">
   <xsl:variable name="var139_title" as="node()?" select="@xlink:title"/>
   <xsl:variable name="var140_type" as="node()?" select="@xlink:type"/>
   <xsl:variable name="var141_href" as="node()?" select="@xlink:href"/>
   <xsl:variable name="var142_role" as="node()?" select="@xlink:role"/>
   <xsl:variable name="var143_arcrole" as="node()?" select="@xlink:arcrole"/>
   <xsl:variable name="var144_show" as="node()?" select="@xlink:show"/>
   <xsl:variable name="var145_remoteSchema" as="node()?" select="@gml:remoteSchema"/>
   <xsl:variable name="var146_nilReason" as="node()?" select="@nilReason"/>
   <xsl:variable name="var147_actuate" as="node()?" select="@xlink:actuate"/>
   <wdb:featureOfInterest>
    <xsl:if test="fn:exists($var140_type)">
     <xsl:attribute name="xlink:type" select="fn:string($var140_type)"/>
    </xsl:if>
    <xsl:if test="fn:exists($var141_href)">
     <xsl:attribute name="xlink:href" select="xs:string(xs:anyURI(fn:string($var141_href)))"/>
    </xsl:if>
    <xsl:if test="fn:exists($var142_role)">
     <xsl:attribute name="xlink:role" select="xs:string(xs:anyURI(fn:string($var142_role)))"/>
    </xsl:if>
    <xsl:if test="fn:exists($var143_arcrole)">
     <xsl:attribute name="xlink:arcrole" select="xs:string(xs:anyURI(fn:string($var143_arcrole)))"/>
    </xsl:if>
    <xsl:if test="fn:exists($var139_title)">
     <xsl:attribute name="xlink:title" select="fn:string($var139_title)"/>
    </xsl:if>
    <xsl:if test="fn:exists($var144_show)">
     <xsl:attribute name="xlink:show" select="fn:string($var144_show)"/>
    </xsl:if>
    <xsl:if test="fn:exists($var147_actuate)">
     <xsl:attribute name="xlink:actuate" select="fn:string($var147_actuate)"/>
    </xsl:if>
    <xsl:if test="fn:exists($var146_nilReason)">
     <xsl:attribute name="nilReason" select="fn:string($var146_nilReason)"/>
    </xsl:if>
    <xsl:if test="fn:exists($var145_remoteSchema)">
     <xsl:attribute name="gml:remoteSchema" select="xs:string(xs:anyURI(fn:string($var145_remoteSchema)))"/>
    </xsl:if>
    <xsl:for-each select="wdb:PhysicalMonitoringObject">
     <wdb:PhysicalMonitoringObject><xsl:sequence select="(./@node(), ./node())"/> </wdb:PhysicalMonitoringObject>
    </xsl:for-each>
    <xsl:for-each select="wdb:WFDGroundWaterBody">
     <wdb:WFDGroundWaterBody><xsl:sequence select="(./@node(), ./node())"/></wdb:WFDGroundWaterBody>
    </xsl:for-each>
    <xsl:for-each select="wdb:WFDSurfaceWaterBody">
     <wdb:WFDSurfaceWaterBody><xsl:sequence select="(./@node(), ./node())"/></wdb:WFDSurfaceWaterBody>
    </xsl:for-each>
   </wdb:featureOfInterest>
  </xsl:for-each>
 </wdb:ObservingCapability>
</xsl:for-each>
<xsl:for-each select="$var1_FeatureCollection/gml:featureMembers/wdb:WFD_SW_MonitoringStation">
 <xsl:variable name="var179_cur" as="node()" select="."/>
 <xsl:variable name="var148_resultof_cast" as="xs:string" select="fn:string(@gml:id)"/>
 <wdb:WFD_SW_MonitoringStation>
  <xsl:attribute name="gml:id" select="$var148_resultof_cast"/>
  <xsl:for-each select="gml:metaDataProperty">
   <gml:metaDataProperty>  <xsl:sequence select="(./@node(), ./node())"/>  </gml:metaDataProperty>
  </xsl:for-each>
  <xsl:for-each select="gml:description">
   <gml:description>  <xsl:sequence select="(./@node(), ./node())"/>  </gml:description>
  </xsl:for-each>
  <xsl:for-each select="gml:descriptionReference">
   <gml:descriptionReference>  <xsl:sequence select="(./@node(), ./node())"/>  </gml:descriptionReference>
  </xsl:for-each>
  <xsl:for-each select="gml:identifier">
   <gml:identifier>  <xsl:sequence select="(./@node(), ./node())"/>  </gml:identifier>
  </xsl:for-each>
  <xsl:for-each select="gml:name">
   <gml:name>  <xsl:sequence select="(./@node(), ./node())"/>  </gml:name>
  </xsl:for-each>
  <xsl:for-each select="gml:boundedBy">
```

```
 <gml:boundedBy>            <xsl:sequence select="(./@node(), ./node())"/>          </gml:boundedBy>
</xsl:for-each>
<xsl:for-each select="gml:location">
 <gml:location>            <xsl:sequence select="(./@node(), ./node())"/>          </gml:location>
</xsl:for-each>
<xsl:for-each select="gml:priorityLocation">
 <gml:priorityLocation>            <xsl:sequence select="(./@node(), ./node())"/>          </gml:priorityLocation>
</xsl:for-each>
<xsl:for-each select="wdb:inspireId">
 <wdb:inspireId>            <xsl:sequence select="(./@node(), ./node())"/>          </wdb:inspireId>
</xsl:for-each>
<xsl:for-each select="wdb:name">
 <wdb:name>            <xsl:sequence select="fn:string(.)"/>            </wdb:name>
</xsl:for-each>
<xsl:for-each select="wdb:additionalDescription">
 <wdb:additionalDescription>            <xsl:sequence select="fn:string(.)"/>            </wdb:additionalDescription>
</xsl:for-each>
<xsl:for-each select="wdb:mediaMonitored">
 <wdb:mediaMonitored>            <xsl:sequence select="(./@node(), ./node())"/>          </wdb:mediaMonitored>
</xsl:for-each>
<xsl:for-each select="wdb:geometry">
 <wdb:geometry>            <xsl:sequence select="(./@node(), ./node())"/>          </wdb:geometry>
</xsl:for-each>
<xsl:for-each select="wdb:legalBackground">
 <wdb:legalBackground>            <xsl:sequence select="(./@node(), ./node())"/>          </wdb:legalBackground>
</xsl:for-each>
<xsl:for-each select="wdb:responsibleParty">
 <wdb:responsibleParty>            <xsl:sequence select="(./@node(), ./node())"/>          </wdb:responsibleParty>
</xsl:for-each>
<xsl:for-each select="wdb:onlineResource">
 <wdb:onlineResource>            <xsl:sequence select="(./@node(), ./node())"/>          </wdb:onlineResource>
</xsl:for-each>
<xsl:for-each select="wdb:purpose">
 <wdb:purpose>            <xsl:sequence select="(./@node(), ./node())"/>          </wdb:purpose>
</xsl:for-each>
<xsl:for-each select="wdb:observingCapability">
 <wdb:observingCapability>            <xsl:sequence select="(./@node(), ./node())"/>          </wdb:observingCapability>
</xsl:for-each>
<xsl:for-each select="wdb:narrower">
 <wdb:narrower>            <xsl:sequence select="(./@node(), ./node())"/>          </wdb:narrower>
</xsl:for-each>
<xsl:for-each select="wdb:supersededBy">
 <wdb:supersededBy>            <xsl:sequence select="(./@node(), ./node())"/>          </wdb:supersededBy>
</xsl:for-each>
<xsl:for-each select="wdb:reportedTo">
 <wdb:reportedTo>            <xsl:sequence select="(./@node(), ./node())"/>          </wdb:reportedTo>
</xsl:for-each>
<xsl:for-each select="wdb:parameter">
 <wdb:parameter>            <xsl:sequence select="(./@node(), ./node())"/>          </wdb:parameter>
</xsl:for-each>
<xsl:for-each select="wdb:lineage">
 <wdb:lineage>            <xsl:sequence select="(./@node(), ./node())"/>          </wdb:lineage>
</xsl:for-each>
<xsl:for-each select="$var2_Import/Row">
 <xsl:variable name="var165_cur" as="node()" select="."/>
 <xsl:variable name="var156_result" as="xs:boolean?">
  <xsl:for-each select="fn:doc($MPN_Query_Results2)/QueryResults">
   <xsl:variable name="var155_result" as="xs:boolean*">
    <xsl:for-each select="MPN">
     <xsl:variable name="var154_cur" as="node()" select="."/>
     <xsl:variable name="var153_result" as="xs:boolean?">
      <xsl:for-each select="WDB_Ident">
       <xsl:variable name="var152_cur" as="node()" select="."/>
       <xsl:variable name="var151_result" as="xs:boolean?">
        <xsl:for-each select="$var154_cur/BULK_Ident">
         <xsl:variable name="var150_cur" as="node()" select="."/>
         <xsl:variable name="var149_result" as="xs:boolean?">
          <xsl:for-each select="$var165_cur/MWADTME">
           <xsl:sequence select="((fn:string($var152_cur) = $var148_resultof_cast) and (
                fn:string($var150_cur) = fn:string(.)))"/>
          </xsl:for-each>
         </xsl:variable>
         <xsl:sequence select="fn:exists($var149_result[.])"/>
        </xsl:for-each>
       </xsl:variable>
       <xsl:sequence select="fn:exists($var151_result[.])"/>
```

```
          </xsl:for-each>
        </xsl:variable>
        <xsl:sequence select="fn:exists($var153_result[.])"/>
      </xsl:for-each>
    </xsl:variable>
    <xsl:sequence select="fn:exists($var155_result[.])"/>
  </xsl:for-each>
 </xsl:variable>
 <xsl:if test="fn:exists($var156_result[.])">
  <xsl:variable name="var157_relatedObservation" as="node()*" select="$var179_cur/wdb:relatedObservation"/>
  <wdb:relatedObservation>
   <xsl:attribute name="xlink:type" select="'simple'"/>
   <xsl:for-each select="MWA_ID">
    <xsl:attribute name="xlink:href" select="xs:string(xs:anyURI(fn:concat('#',
          fn:concat('BULK.MWA.', xs:string(xs:integer(fn:string(.)))))))"/>
   </xsl:for-each>
   <xsl:for-each select="$var157_relatedObservation">
    <xsl:variable name="var158_role" as="node()?" select="@xlink:role"/>
    <xsl:if test="fn:exists($var158_role)">
     <xsl:attribute name="xlink:role" select="xs:string(xs:anyURI(fn:string($var158_role)))"/>
    </xsl:if>
   </xsl:for-each>
   <xsl:for-each select="$var157_relatedObservation">
    <xsl:variable name="var159_arcrole" as="node()?" select="@xlink:arcrole"/>
    <xsl:if test="fn:exists($var159_arcrole)">
     <xsl:attribute name="xlink:arcrole" select="xs:string(xs:anyURI(fn:string($var159_arcrole)))"/>
    </xsl:if>
   </xsl:for-each>
   <xsl:for-each select="$var157_relatedObservation">
    <xsl:variable name="var160_title" as="node()?" select="@xlink:title"/>
    <xsl:if test="fn:exists($var160_title)">
     <xsl:attribute name="xlink:title" select="fn:string($var160_title)"/>
    </xsl:if>
   </xsl:for-each>
   <xsl:for-each select="$var157_relatedObservation">
    <xsl:variable name="var161_show" as="node()?" select="@xlink:show"/>
    <xsl:if test="fn:exists($var161_show)">
     <xsl:attribute name="xlink:show" select="fn:string($var161_show)"/>
    </xsl:if>
   </xsl:for-each>
   <xsl:for-each select="$var157_relatedObservation">
    <xsl:variable name="var162_actuate" as="node()?" select="@xlink:actuate"/>
    <xsl:if test="fn:exists($var162_actuate)">
     <xsl:attribute name="xlink:actuate" select="fn:string($var162_actuate)"/>
    </xsl:if>
   </xsl:for-each>
   <xsl:for-each select="$var157_relatedObservation">
    <xsl:variable name="var163_nilReason" as="node()?" select="@nilReason"/>
    <xsl:if test="fn:exists($var163_nilReason)">
     <xsl:attribute name="nilReason" select="fn:string($var163_nilReason)"/>
    </xsl:if>
   </xsl:for-each>
   <xsl:for-each select="$var157_relatedObservation">
    <xsl:variable name="var164_remoteSchema" as="node()?" select="@gml:remoteSchema"/>
    <xsl:if test="fn:exists($var164_remoteSchema)">
     <xsl:attribute name="gml:remoteSchema" select="xs:string(xs:anyURI(fn:string($var164_remoteSchema)))"/>
    </xsl:if>
   </xsl:for-each>
   <xsl:for-each select="$var157_relatedObservation/wdb:OM_Observation">
    <wdb:OM_Observation>
     <xsl:sequence select="(./@node(), ./node())"/>
    </wdb:OM_Observation>
   </xsl:for-each>
  </wdb:relatedObservation>
 </xsl:if>
</xsl:for-each>
<xsl:for-each select="$var3_Import/Row">
 <xsl:variable name="var174_cur" as="node()" select="."/>
 <xsl:variable name="var173_result" as="xs:boolean?">
  <xsl:for-each select="fn:doc($MPN_Query_Results2)/QueryResults">
   <xsl:variable name="var172_result" as="xs:boolean*">
    <xsl:for-each select="MPN">
     <xsl:variable name="var171_cur" as="node()" select="."/>
     <xsl:variable name="var170_result" as="xs:boolean?">
      <xsl:for-each select="WDB_Ident">
       <xsl:variable name="var169_cur" as="node()" select="."/>
```

```
                <xsl:variable name="var168_result" as="xs:boolean?">
                  <xsl:for-each select="$var171_cur/LD_Ident">
                    <xsl:variable name="var167_cur" as="node()" select="."/>
                    <xsl:variable name="var166_result" as="xs:boolean?">
                      <xsl:for-each select="$var174_cur/Lab">
                        <xsl:sequence select="((fn:string($var169_cur) = $var148_resultof_cast) and (
                              fn:string($var167_cur) = fn:string(.)))"/>
                      </xsl:for-each>
                    </xsl:variable>
                    <xsl:sequence select="fn:exists($var166_result[.])"/>
                  </xsl:for-each>
                </xsl:variable>
                <xsl:sequence select="fn:exists($var168_result[.])"/>
              </xsl:for-each>
            </xsl:variable>
            <xsl:sequence select="fn:exists($var170_result[.])"/>
          </xsl:for-each>
        </xsl:variable>
        <xsl:sequence select="fn:exists($var172_result[.])"/>
      </xsl:for-each>
    </xsl:variable>
    <xsl:if test="fn:exists($var173_result[.])">
      <wdb:relatedObservation>
        <xsl:attribute name="xlink:type" select="'simple'"/>
        <xsl:for-each select="Recnum">
          <xsl:attribute name="xlink:href" select="xs:string(xs:anyURI(fn:concat('#',
              fn:concat('LD.MON.', xs:string(xs:integer(fn:string(.)))))))"/>
        </xsl:for-each>
      </wdb:relatedObservation>
    </xsl:if>
  </xsl:for-each>
  <xsl:for-each select="wdb:sampledFeature">
    <wdb:sampledFeature>        <xsl:sequence select="(./@node(), ./node())"/>      </wdb:sampledFeature>
  </xsl:for-each>
  <xsl:for-each select="wdb:positionalAccuracy">
    <xsl:variable name="var178_cur" as="node()" select="."/>
    <xsl:variable name="var175_result" as="xs:boolean*">
      <xsl:for-each select="@xsi:type">
        <xsl:sequence select="(fn:resolve-QName(fn:string(.), $var178_cur) =
xs:QName('gmd:DQ_RelativeInternalPositionalAccuracy_Type'))"/>
      </xsl:for-each>
    </xsl:variable>
    <xsl:if test="fn:exists($var175_result[.])">
      <xsl:variable name="var176_uuid" as="node()?" select="@uuid"/>
      <xsl:variable name="var177_id" as="node()?" select="@id"/>
      <wdb:positionalAccuracy>
        <xsl:attribute name="xsi:type" select="xs:QName('gmd:DQ_RelativeInternalPositionalAccuracy_Type')"/>
        <xsl:if test="fn:exists($var177_id)">
          <xsl:attribute name="id" select="fn:string($var177_id)"/>
        </xsl:if>
        <xsl:if test="fn:exists($var176_uuid)">
          <xsl:attribute name="uuid" select="fn:string($var176_uuid)"/>
        </xsl:if>
        <xsl:for-each select="gmd:nameOfMeasure">
          <gmd:nameOfMeasure>          <xsl:sequence select="(./@node(), ./node())"/>          </gmd:nameOfMeasure>
        </xsl:for-each>
        <xsl:for-each select="gmd:measureIdentification">
          <gmd:measureIdentification> <xsl:sequence select="(./@node(), ./node())"/>          </gmd:measureIdentification>
        </xsl:for-each>
        <xsl:for-each select="gmd:measureDescription">
          <gmd:measureDescription><xsl:sequence select="(./@node(), ./node())"/>          </gmd:measureDescription>
        </xsl:for-each>
        <xsl:for-each select="gmd:evaluationMethodType">
          <gmd:evaluationMethodType><xsl:sequence select="(./@node(), ./node())"/>          </gmd:evaluationMethodType>
        </xsl:for-each>
        <xsl:for-each select="gmd:evaluationMethodDescription">
          <gmd:evaluationMethodDescription>
            <xsl:sequence select="(./@node(), ./node())"/>
          </gmd:evaluationMethodDescription>
        </xsl:for-each>
        <xsl:for-each select="gmd:evaluationProcedure">
          <gmd:evaluationProcedure><xsl:sequence select="(./@node(), ./node())"/>          </gmd:evaluationProcedure>
        </xsl:for-each>
        <xsl:for-each select="gmd:dateTime">
          <gmd:dateTime>          <xsl:sequence select="(./@node(), ./node())"/>          </gmd:dateTime>
        </xsl:for-each>
```

```
            <xsl:for-each select="gmd:result">
              <gmd:result>                <xsl:sequence select="(./@node(), ./node())"/>              </gmd:result>
            </xsl:for-each>
          </wdb:positionalAccuracy>
        </xsl:if>
      </xsl:for-each>
      <xsl:for-each select="wdb:hostedProcedure">
        <wdb:hostedProcedure>            <xsl:sequence select="(./@node(), ./node())"/>          </wdb:hostedProcedure>
      </xsl:for-each>
      <xsl:for-each select="wdb:instantiatedAs">
        <wdb:instantiatedAs>          <xsl:sequence select="(./@node(), ./node())"/>          </wdb:instantiatedAs>
      </xsl:for-each>
      <xsl:for-each select="wdb:representativePoint">
        <wdb:representativePoint>            <xsl:sequence select="(./@node(), ./node())"/>          </wdb:representativePoint>
      </xsl:for-each>
      <xsl:for-each select="wdb:measurementRegime">
        <wdb:measurementRegime>
          <xsl:choose>
            <xsl:when test="(fn:translate(fn:string(@xsi:nil), 'true ', '1') = '1')">
              <xsl:attribute name="xsi:nil" select="'true'"/>
            </xsl:when>
            <xsl:otherwise>              <xsl:sequence select="fn:string(.)"/>              </xsl:otherwise>
          </xsl:choose>
        </wdb:measurementRegime>
      </xsl:for-each>
      <xsl:for-each select="wdb:mobile">
        <wdb:mobile>
          <xsl:choose>
            <xsl:when test="(fn:translate(fn:string(@xsi:nil), 'true ', '1') = '1')">
              <xsl:attribute name="xsi:nil" select="'true'"/>
            </xsl:when>
            <xsl:otherwise>              <xsl:sequence select="xs:string(xs:boolean(fn:string(.)))"/>              </xsl:otherwise>
          </xsl:choose>
        </wdb:mobile>
      </xsl:for-each>
      <xsl:for-each select="wdb:resultAcquisitionSource">
        <wdb:resultAcquisitionSource>
          <xsl:choose>
            <xsl:when test="(fn:translate(fn:string(@xsi:nil), 'true ', '1') = '1')">
              <xsl:attribute name="xsi:nil" select="'true'"/>
            </xsl:when>
            <xsl:otherwise>              <xsl:sequence select="fn:string(.)"/>              </xsl:otherwise>
          </xsl:choose>
        </wdb:resultAcquisitionSource>
      </xsl:for-each>
      <xsl:for-each select="wdb:speciallisedEMFType">
        <wdb:speciallisedEMFType>            <xsl:sequence select="(./@node(), ./node())"/>          </wdb:speciallisedEMFType>
      </xsl:for-each>
      <xsl:for-each select="wdb:relatedTo">
        <wdb:relatedTo>            <xsl:sequence select="(./@node(), ./node())"/>          </wdb:relatedTo>
      </xsl:for-each>
      <xsl:for-each select="wdb:operationalActivityPeriod">
        <wdb:operationalActivityPeriod><xsl:sequence select="(./@node(), ./node())"/>          </wdb:operationalActivityPeriod>
      </xsl:for-each>
      <xsl:for-each select="wdb:subsites">
        <wdb:subsites>            <xsl:sequence select="(./@node(), ./node())"/>          </wdb:subsites>
      </xsl:for-each>
      <xsl:for-each select="wdb:numberOfPointsInSubsite">
        <wdb:numberOfPointsInSubsite>
          <xsl:choose>
            <xsl:when test="(fn:translate(fn:string(@xsi:nil), 'true ', '1') = '1')">
              <xsl:attribute name="xsi:nil" select="'true'"/>
            </xsl:when>
            <xsl:otherwise>              <xsl:sequence select="xs:string(xs:integer(fn:string(.)))"/>              </xsl:otherwise>
          </xsl:choose>
        </wdb:numberOfPointsInSubsite>
      </xsl:for-each>
      <xsl:for-each select="wdb:monitoringUse">
        <wdb:monitoringUse>            <xsl:sequence select="(./@node(), ./node())"/>          </wdb:monitoringUse>
      </xsl:for-each>
    </wdb:WFD_SW_MonitoringStation>
  </xsl:for-each>
 </gml:featureMembers>
</gml:FeatureCollection>
</xsl:template>
</xsl:stylesheet>
```

This research was sponsored by:

piLot Survey Services

Informatiehuis Water
laat data stromen