

DSRL – A Quick Tutorial

This quick tutorial will show you the key features of the ISO/IEC 19757-8 Document Schema Renaming Language (DSRL) and demonstrate how DSRL (pronounced “disrule”) can be used to convert XML files in local dialects to a form that can be processed by an generalized schema.

What does DSRL do?

DSRL allows you to convert element name, attribute names, attribute values tokens, entity names and processing instruction names from locally meaningful names to the names imposed by validation schemas. It contains the following options that can be used to create a translation map:

1. For each element whose name you want to map to that in a validation schema create a rule of the form:

```
<dsrl:element-name-map target="my-element-name">
  schema-element-name </dsrl:element-name-map>
```

2. For each attribute whose name you want to map to that in a validation schema create a rule of the form:

```
<dsrl:attribute-name-map target="my-element-name[@my-attribute-name]">
  schema-attribute-name </dsrl:attribute-name-map>
```

3. For each attribute value token whose name you want to map to that in a validation schema create a rule of the form:

```
<dsrl:attribute-values-map target="adresse[@sorte]">
  my-value1 schema-value1 my-value2 schema-value2
</dsrl:attribute-values-map>
```

4. For each of entity whose name you want to map to that in an entity set a rule of the form:

```
<dsrl:entity-name-map target="adresse[@sorte]">
  xml-name      my-name1
  html-name     my-name2
  dtd-name      my-name3
</dsrl:entity-name-map>
```

5. For each of processing instruction type whose name you want to map to that in a validation schema create a rule of the form:

```
<dsrl:map-pi-target target-name="schema-PI-name"
  alternative-names="my-PI-name1 my-PI-name2"/>
```

In addition you can manage the values of attributes and the content of elements:

1. For each attribute that you want to give a default value to create a rule of the form:

```
<dsrl:default-attribute-values target="my-element-name">
  attribute-name attribute-value
</dsrl:default-attribute-values>
```

2. For each element that you want to give a default value to create a rule of the form:

```
<dsrl:default-content target="my-element-name" parent="parent-element-name"
  map-to-element-name="schema-element-name" force-default="true">
  Default content
</dsrl:default-content>
```

Other techniques provided in the DSRL standard allow you to:

1. Record your translation requirements as attributes of an element in a document instance:

```
<my-element-name dsrl:element-name-map="schema-element-name">
```

2. Record your translation requirements as application-specific information in a schema:

```
<appinfo>
  <dsrl:element-name-map target="my-element-name">
    schema-element-name </dsrl:element-name-map>
</appinfo>
```

Using XSLT to process DSRL maps

DSDL.ORG provides an XSLT 2.0 transformation, `TransformDSRLmap.xsl`, that can be used to convert DSRL maps into application-specific XSLT transformations.

The example map provided with the transformation rules, `MapFrenchToEuropeanAddress.xml`, contains examples of the use of each DSRL feature (at the expense of being somewhat unrealistic!). It reads:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="TransformDSRLmaps.xsl"?>
<dsrl:maps xmlns:dsrl="http://purl.oclc.org/dsdl/dsrl">
  <!--Mapping of element and attribute names and attribute values-->
  <dsrl:element-name-map target="adresse"
    targetNamespace="http://www.my-co.fr/adresses"
    targetSchemaLocation="EuropeanAddress.xsd">address</dsrl:element-name-map>
  <dsrl:attribute-name-map target="adresse[@sorte]">
    type</dsrl:attribute-name-map>
  <dsrl:attribute-values-map target="adresse[@sorte]">
    maison home
    bureau office
  </dsrl:attribute-values-map>
  <dsrl:attribute-name-map target="adresse[@torte]">
    type2</dsrl:attribute-name-map>
  <dsrl:attribute-values-map target="adresse[@torte]">
    maison home
    bureau office
  </dsrl:attribute-values-map>
  <dsrl:element-name-map target="numero">
    building-identifiser</dsrl:element-name-map>
  <dsrl:element-name-map target="rue">road</dsrl:element-name-map>
  <dsrl:element-name-map target="ville">locality</dsrl:element-name-map>
  <dsrl:element-name-map target="cit ">postal-town</dsrl:element-name-map>
  <dsrl:element-name-map target="d partement">county</dsrl:element-name-map>
  <dsrl:element-name-map target="code-postal">postcode</dsrl:element-name-map>
  <dsrl:element-name-map target="pays">country</dsrl:element-name-map>
  <!--Assigning default values-->
  <dsrl:entity-name-map>
    eacute e
    amp et
    amp and
    lt open-tag
    gt close-tag
  </dsrl:entity-name-map>
  <dsrl:map-pi-target target-name="PIname"
    alternative-names="PInameAsInput AlternativePIname"/>
  <dsrl:map-pi-target target-name="ProcessThis" alternative-names="MyPI"/>
  <dsrl:default-content target="cit " parent="adresse"
    map-to-element-name="postal-town" force-default="true">
    Bordeaux
  </dsrl:default-content>

  <dsrl:default-content target="ville" parent="adresse"
    map-to-element-name="locality">
    <dsrl:default-attribute-values force-default="false">
      required false
    </dsrl:default-attribute-values>
    Downtown
  </dsrl:default-content>

  <dsrl:default-attribute-values target="pays" force-default="true">
    code-system iso3166
  </dsrl:default-attribute-values>
</dsrl:maps>
```

When this file is transformed by `TransformDSRLmap.xsl`, it produces an XSLT stylesheet, `DSRLtransform.xsl` in the example suite, that can be used to convert French addresses to their European equivalents. For example, the unrepresentative test file:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="DSRLtransform.xsl"?>
<!DOCTYPE doc SYSTEM "Adresse.dtd" [
  <!ENTITY e "&#233;">
]>
<?AlternativePIname PI proceeds root?>
<doc xmlns:xi="http://www.w3.org/2001/XInclude">
  <adresse sorte="maison">
    <numero>29</numero>
    <rue>Rue Bricot</rue>
    <ville>Monmartre</ville>
    <?PInameAsInput Embedded PI?>
    <cit  >Paris</cit  >
    <d  partement>  le de France</d  partement>
    <code-postal>95010</code-postal>
    <pays>France &open-tag;this&and;that&close-tag;</pays>
  </adresse>
  <adresse sorte="bureau">
    <numero>2</numero>
    <rue>Avenue Charles de Gaulle</rue>
    <?MyPI Another mapped PI?>
    <cit  >Toulon</cit  >
    <d  partement>Aud&e; &et; Dauphine</d  partement>
    <code-postal>12345</code-postal>
    <pays>France</pays>
  </adresse>
</doc>
<?AlternativePIname PI follows root?>
```

will be converted to:

```
<?xml version="1.0" encoding="UTF-8"?>
<?PIname PI proceeds root?><doc xmlns:xi="http://www.w3.org/2001/XInclude">
  <address xmlns="http://csw.co.uk/addresses"
    targetNamespace="http://csw.co.uk/addresses"
    targetSchemaLocation="EuropeanAddress.xsd"
    type="maison">
    <building-identifiant>29</building-identifiant>
    <road>Rue Bricot</road>
    <locality>Monmartre</locality>
    <?PIname Embedded PI?><postal-town>Bordeaux</postal-town>
    <county>  le de France</county>
    <postcode>95010</postcode>
    <country code-system="iso3166">France &lt;this&amp;that&gt;</country>
  </address>
  <address xmlns="http://csw.co.uk/addresses"
    targetNamespace="http://csw.co.uk/addresses"
    targetSchemaLocation="EuropeanAddress.xsd"
    type="bureau">
    <building-identifiant>2</building-identifiant>
    <road>Avenue Charles de Gaulle</road>
    <?ProcessThis Another mapped PI?><postal-town>Bordeaux</postal-town>
    <county>Aud   &amp; Dauphine</county>
    <postcode>12345</postcode>
    <country code-system="iso3166">France</country>
    <locality>Downtown</locality>
  </address>
</doc>
<?PIname PI follows root?>
```