DFDL Introduction for Beginners

Lesson 6: Describing Optional and Repeating Data

In the previous lessons all the elements had to occur exactly once in the data stream; in this lesson we will learn how to model elements that occur optionally and elements that occur repeatedly.

The permitted number of occurrences of an element in the data stream is given by the element's XML Schema `xs:minOccurs` and `xs:maxOccurs` properties. Because these are XML Schema properties and not DFDL properties, either can be omitted, and if so default to "1".

An element that occurs exactly once is identified by both `xs:minOccurs` and `xs:maxOccurs` having the value "1".

Elements can be modeled to occur optionally. An optional element is identified by setting `xs:minOccurs` to "0" and setting `dfdl:occursCountKind` to indicate how to determine if the element is present.

Elements can also occur repeatedly and are referred to as arrays. An array is identified by setting `xs:maxOccurs` to greater than "1" or the special value "unbounded" (meaning there is no maximum number of occurrences) and setting `dfdl:occursCountKind` to indicate how to determine how many occurrences of the element are present. The number of occurrences in an array can be fixed or can vary. A fixed array is identified by setting both `xs:minOccurs` and `xs:maxOccurs` to the same value greater than "1" and `dfdl:occursCountKind` to `"fixed"`. A variable array is identified by setting `xs:minOccurs` and `xs:maxOccurs` to different values and `dfdl:occursCountKind` to `"parsed"` or one of the advanced options described in a later lesson. If `xs:minOccurs` is set to "0" then the element is both optional and an array.

It can be observed that the `xs:minOccurs` value provides the number of elements that are required to be in the data stream.

Modeling optional data

We extend the variable length Address example to model an optional 'country' element.

Example 1: Address with optional delimited data elements

Data stream

```
[house:118*street:Ridgewood Circle*city:Rochester*state:NY]
```

```
OR

[house:118*street:Ridgewood Circle*city:Rochester*state:N
Y*country:USA]
```

DFDL schema

```
1 <xs:schema … xmlns:dfdl="http://www.ogf.org/dfdl/dfdl-1.0/">

2  <xs:annotation>
3    <xs:appinfo source="http://www.ogf.org/dfdl/" >
4      <dfdl:format representation="text"
                    lengthKind="delimited"
                    encoding="ASCII" />
5    </xs:appinfo>
6  </xs:annotation>

7  <xs:element name="address" dfdl:lengthKind="implicit"
         dfdl:initiator="[" dfdl:terminator="]">
8    <xs:complexType>
9      <xs:sequence dfdl:sequenceKind="ordered"
                    dfdl:separator="*"
                    dfdl:separatorPosition="infix" >
10       <xs:element name="houseNumber" type="xs:string"
               dfdl:initiator="house:" />
11        <xs:element name="street" type="xs:string"
               dfdl:initiator="street:" />
12       <xs:element name="city" type="xs:string"
               dfdl:initiator="city:" />
13       <xs:element name="state" type="xs:string"
               dfdl:initiator="state:" />
14       <xs:element name="country" type="xs:string"
               dfdl:initiator="country:"
               xs:minOccurs="0" xs:maxOccurs="1"
               dfdl:occursCountKind="parsed" />
15     </xs:sequence>
16   </xs:complexType>
17 </xs:element>

18</xs:schema>
```

Infoset:

```
    address
        houseNumber(string)    '118'
        street(string)         'Ridgewood Circle'
        city(string)           'Rochester'
        state(string)          'NY'

OR
```

```
    address
        houseNumber(string)    '118'
        street(string)         'Ridgewood Circle'
        city(string)           'Rochester'
        state(string)          'NY'
        country(string)        'USA'
```

The optional ‘country’ element on line 14 is indicated by `xs:minOccurs`= 0.

The parser needs to be able to find out if the element is present in the data stream. For initiated elements like the example above it can be done easily by looking for the initiator so the `dfdl:occursCountKind` should be set to "parsed".

Uninitiated text elements are often not optional or there is something in the data stream which can indicate the absence of the optional element such as a delimiter or another field. The use of another field requires `dfdl:occursCountKind` to be set to "expression" which is described in a later lesson.

Fixed length binary fields are usually not optional but can use the same techniques as text elements.

Modeling fixed arrays

This is typical of fixed length data where there are no initiators, so all elements are identified by their position in the data stream. We extend the fixed length Address example to model the ‘street’ element repeating a fixed number of times.

Example 1: Address with repeating fixed length data elements

Data stream

```
000118Ridgewood Circle    Main Street          Rochester
        NYUSA
```

DFDL schema

```
1 <xs:schema … xmlns:dfdl=“http://www.ogf.org/dfdl/dfdl-1.0/“>

2  <xs:annotation>
3    <xs:appinfo source=”http://www.ogf.org/dfdl/” >
4      <dfdl:format representation=”text”
                    lengthKind="explicit"
                    lengthUnits="bytes"
                    encoding=”ASCII” />
5    </xs:appinfo>
6  </xs:annotation>

7 <xs:element name="address" dfdl:lengthKind="implicit">
8  <xs:complexType>
9    <xs:sequence dfdl:sequenceKind="ordered">
```

```
10   <xs:element name="houseNumber" type="xs:string"
              dfdl:length="6" />
11   <xs:element name="street" type="xs:string"
              dfdl:length="20"
              xs:minOccurs=”2” xs:maxOccurs=”2”
              dfdl:occursCountKind="fixed"/>
12   <xs:element name="city" type="xs:string"
              dfdl:length="20" />
13   <xs:element name="state" type="xs:string"
              dfdl:length="2" />
14        <xs:element name="country" type="xs:string"
              dfdl:length="20" />

15      </xs:sequence>
16    </xs:complexType>
17 </xs:element>

18</xs:schema>
```

Infoset

```
    address
        houseNumber(string)   '000118'
        street(string)        'Ridgewood Circle    '
        street(string)        'Main Street         '
        city(string)          'Rochester           '
        state(string)         'NY'
        country(string)       'USA                 '
```

On line 11 element ‘street’ occurs exactly twice so `xs:minOccurs` and `xs:maxOccurs` are both set to “2”' and `dfdl:occursCountKind` is "fixed".

Modeling variable arrays

This is typical of data where there are initiators. We extend the variable length Address example to model the ‘street’ element repeating a variable number of times.

Example 2: Address with repeating delimited data elements

Data stream

```
[house:118*street:Ridgewood Circle*street:Main Street*cit
y:Rochester*state:NY*country:USA]
```

DFDL schema

```
1 <xs:schema … xmlns:dfdl=“http://www.ogf.org/dfdl/dfdl-1.0/”>

2  <xs:annotation>
3    <xs:appinfo source=”http://www.ogf.org/dfdl/” >
4      <dfdl:format representation=”text”
                    lengthKind="delimited"
                    encoding=”ASCII” />
```

```
5    </xs:appinfo>
6  </xs:annotation>

7  <xs:element name="address" dfdl:lengthKind="implicit"
         dfdl:initiator="[" dfdl:terminator="]">
8    <xs:complexType>
9      <xs:sequence dfdl:sequenceKind="ordered"
                    dfdl:separator="*"
                    dfdl:separatorPosition="infix" >
10       <xs:element name="houseNumber" type="xs:string"
                  dfdl:initiator="house:" />
11       <xs:element name="street" type="xs:string"
                  dfdl:initiator="street:"
                  xs:minOccurs="0" xs:maxOccurs="2"
                  dfdl:occursCountKind="parsed" />
12       <xs:element name="city" type="xs:string"
                  dfdl:initiator="city:" />
13       <xs:element name="state" type="xs:string"
                  dfdl:initiator="state:" />
14       <xs:element name="country" type="xs:string"
                  dfdl:initiator="country:"
                  xs:minOccurs="0" xs:maxOccurs="1"
                  dfdl:occursCountKind="parsed" />

15     </xs:sequence>
16   </xs:complexType>
17 </xs:element>

18</xs:schema>
```

Infoset

```
    address
        houseNumber(string)    '118'
        street(string)         'Ridgewood Circle'
        street(string)         'Main Street'
        city(string)           'Rochester'
        state(string)          'NY'
        country(string)        'USA'
```

On line 11 the element ‘street’ occurs 0, 1 or 2 times so `xs:minOccurs` is set to “0” and `xs:maxOccurs` is set to “2”'. The element has an initiator "street:" and can be easily identified in the data stream, so `dfdl:occursCountKind` is "parsed".

With delimited text data it is common to use a different separator between the occurrences of an array compared to the separator used between the elements of the wider sequence. To specify the properties of the array as a whole, such as its separator, we model the array by wrapping the repeating element in a complex element. The example below shows the repeating

element wrapped by an element called ‘streets’, the `xs:sequence` of which specifies a different separator of “~”..

Example 4: Address with repeating delimited data elements (different separator)

Data stream

```
[house:118*street:Ridgewood Circle~street:Main Street*cit
y:Rochester*state:NY*country:USA]
```

DFDL schema

```
1 <xs:schema … xmlns:dfdl=“http://www.ogf.org/dfdl/dfdl-1.0/“>

2  <xs:annotation>
3    <xs:appinfo source=”http://www.ogf.org/dfdl/” >
4      <dfdl:format representation=”text”
                    lengthKind="delimited"
                    encoding=”ASCII” />
5    </xs:appinfo>
6  </xs:annotation>

7  <xs:element name="address" dfdl:lengthKind="implicit"
         dfdl:initiator="[" dfdl:terminator"]">
8    <xs:complexType>
9      <xs:sequence dfdl:sequenceKind="ordered"
                    dfdl:separator="*"
                    dfdl:separatorPosition="infix" >
10       <xs:element name="houseNumber" type="xs:string"
               dfdl:initiator="house:" />

11a      <xs:element name="streets"
             dfdl:lengthKind="implicit">
11b        <xs:complexType>
11c          <xs:sequence dfdl:sequenceKind="ordered"
                 dfdl:separator="~"
                 dfdl:separatorPosition="infix" >
11d            <xs:element name="street" type="xs:string"
                 dfdl:initiator="street:"
                 xs:minOccurs=”0” xs:maxOccurs=”2”
                 dfdl:OccursCountKind="parsed" />
11e          </xs:sequence>
11f        </xs:complexType>
11g      </xs:element>

12       <xs:element name="city" type="xs:string"
               dfdl:initiator="city:" />
13       <xs:element name="state" type="xs:string"
               dfdl:initiator="state:" />
14       <xs:element name="country" type="xs:string"
               xs:minOccurs=0 xs:maxOccurs=1
               dfdl:OccursCountKind="parsed"
               dfdl:initiator="country:" />
```

```
15      </xs:sequence>
16    </xs:complexType>
17 </xs:element>

18</xs:schema>
```

Infoset

```
    address
        houseNumber(string)    '118'
        streets
            street(string)     'Ridgewood Circle'
            street(string)     'Main Street'
        city(string)           'Rochester'
        state(string)          'NY'
        country(string)        'USA'
```

.

Summary

In this lesson we have looked at how you can define optional and repeating elements. We have only shown simple elements but the same principles apply to complex elements, allowing the modeling of optional and repeating structures. We have seen that there is more than one way in DFDL to determine the number of occurrences of an element in the data stream, controlled by the `dfdl:occursCountKind` property

In later lessons we will look at how `dfdl:occursCountKind` can be used to specify more advanced ways of determining the number of occurrences, including using count elements within the data stream itself or the use of special values in the data to indicate the end of the array