DFDL Introduction for Beginners

Lesson 6: Describing Optional and Repeating Data

Version	Author	Date	Change
1	A Powell	2011	Created
2	S Hanson	2011-03-30	Updated
3	S Hanson	2013-09-17	Expand and match latest spec

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 minOccurs and 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 minOccurs and maxOccurs having the value '1' or defaulting to '1'.

Elements can be modeled to occur optionally. An optional element is identified by setting 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 element is identified by setting 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 minOccurs and maxOccurs to the same value greater than '1'. If minOccurs is set to '0' then the element is both optional and an array.

The dfdl:occursCountKind property

This is an important property and must be set for all optional elements and array elements. It determines how the number of occurrences is identified in the data stream.

'fixed'
This should be used when the number of occurrences is always the same. The number is provided by the XSD maxOccurs property. The minOccurs property must be the same value.

'implicit'
This should be used when the number of occurrences varies and the parser will determine the number automatically within the bounds given by XSD

minOccurs and maxOccurs. For example, occurrences are determined by using an initiator or a separator.

'parsed'
This should be used when the number of occurrences varies and the parser will determine the number automatically but not using any bounds. For example, occurrences are determined by using an initiator or a separator.

'expression'
This should be used when the number of occurrences varies and the actual number is provided earlier in the data stream. The dfdl:occursCount property provides a DFDL expression which the parser evaluates to obtain the number. Expressions are covered in a later lesson.

'stopValue'
This should be used when the end of the array is signalled by an occurrence which has a special 'stop value'. The dfdl:occursStopValue property provides the list of special 'stop values' to use. The 'stop value' itself is not considered to be an occurrence and is not added to the infoset.

Notice that XSD minOccurs and maxOccurs are only used to assist the parser when dfdl:occursCountKind is 'fixed' and 'implicit'. However, if validation is switched on, the parser checks that the number of occurrences in the infoset is within the bounds specified by minOccurs and maxOccurs, whatever the setting of dfdl:occursCountKind.

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:N
Y]

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:"
                minOccurs="0" 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 XSD 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’ or ‘implicit’.

Text elements without an initiator are often not optional, or if they are then there is something in the data stream which can indicate the absence of the optional element, such as a delimiter or perhaps another element earlier in the data stream (dfdl:occursCountKind 'expression').

Fixed length binary fields are usually not optional, but if they are 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 2: 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"
              minOccurs="2" 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 XSD minOccurs and 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 3: 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:"
               minOccurs=”1” 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 1 or 2 times so XSD minOccurs is set to ‘1’ 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’ or ‘implicit’.

Notice that when an array element has an initiator, every occurrence in the data must have the initiator. What happens if only the first occurrence in the data has the initiator, so the initiator is really identifying the array as a whole? This is best modeled by wrapping the array element in a complex element which carries the initiator, and removing the initiator from the array element.

A commonly used technique is the use of a different separator between the occurrences, as it enables the occurrences of the array element to be distinguished from the following element, especially useful when there are no initiators. Again, this is best modeled by wrapping the array element in a complex element, the xs:sequence of which carries the different separator.

We extend the example to show both of these, the ‘street’ element losing its initiator and being wrapped by an element called ‘streets’ with initiator ‘streets:’ and with an xs:sequence with different separator ‘~’.

Example 4: Address with repeating delimited data elements (delimiters for the array as a whole)

Data stream

```
[house:118*streets:Ridgewood Circle~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:initiator="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"
                 minOccurs="1" 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 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, and shown some examples.

We have only shown simple elements but the same principles apply to complex elements, allowing the modeling of optional and repeating structures.