DFDL Introduction for Beginners

Lesson 5: Modeling Alternative Structures

In lesson 2 we learnt that an `xs:sequence` is used when all the fields in the structure appear in the data stream. In this lesson we introduce `xs:choice` which is used when only one of the fields in the structure may appear (that is they are alternatives). Each alternative in the `xs:choice` is known as a 'choice branch'.

When parsing a data stream and a choice is encountered it must be possible to discover which of the possible alternatives is present. The first choice branch is used to parse the data stream. If the first branch fails to parse the data stream completely correctly then the next branch, in schema definition order, is tried and so on until a branch is found that parses the data stream correctly. This technique is sometimes called 'backtracking' or 'speculative parsing'.

Note that it is necessary to distinguish between a parsing error caused by the choice branch not being present and an error caused by the branch being present but badly formed. The former condition causes backtracking, but the latter condition does not. The `dfdl:initiatedContent` property can help here by stipulating that each branch must have an initiator and, if the initiator is found, then that choice branch is present. If `dfdl:initiatedContent` is set to "yes" and the initiator is found but a subsequent element in the choice branch fails to parse then it is a real parsing error in that choice branch and no backtracking takes place. If `dfdl:initiatedContent` is set to "no" and the initiator is found but a subsequent element in the choice branch fails to parse then the choice branch is deemed to be not present, backtracking occurs, and the next branch is tried.

Modeling alternative data structures

Building on the Address example from lesson 4, we introduce an alternative that allows either a state or a county or a province element to be present, in order to allow a US or UK or Canadian address.

Example 1: Choice of addresses

Data stream

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

OR

[house:25*street:The Hundred*city:Romsey*county:Hampshire
]
```

```
OR

[house:279*street:Lakeside Road*city:Toronto*province:Ont
ario]
```

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"
            dfdl:separatorPolicy="required">

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="jurisdiction"
               dfdl:lengthKind="implicit">
14            <xs:complexType>
15              <xs:choice
                    dfdl:choiceLengthKind="implicit"
                    dfdl:initiatedContent="yes">

16                  <xs:element name="state"
                       type="xs:string"
                       dfdl:initiator="state:" />
17                  <xs:element name="county"
                       type="xs:string"
                       dfdl:initiator="county:" />
18                  <xs:element name="province"
                       type="xs:string"
                       dfdl:initiator="province:" />
```

```
19                <xs:choice/>
20             </xs:complexType>
21          </xs:element>

22        </xs:sequence>

23    </xs:complexType>
24 </xs:element>

25 </xs:schema>
```

Infoset

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

OR

    address
        houseNumber(string)    '25'
        street(string)         'Main Street'
        city(string)           'Newtown'
        jurisdiction
           county(string)      'Hampshire'

OR

    address
        houseNumber(string)    '279'
        street(string)         'Lakeside Road'
        city(string)           'Toronto'
        jurisdiction
           province(string)    'Ontario'
```

The `xs:choice` on line 15 indicates that there can be one of a number of elements at this point in the data stream. The `dfdl:initiatedContent` = "yes" indicates that the elements' initiators should be used alone to decide which is present. The parser will look for the "state:" initiator and if found will add the state element to the infoset. Otherwise it will look for a "county:" initiator and add a county element to the infoset. Otherwise it will look for a "province:" initiator and add a province element to the infoset.

The `dfdl:choiceLengthKind` = "implicit" specified on the xs:choice on line 15 indicates that the length of the choice in the data stream is the length of the selected choice branch. A `dfdl:choiceLengthKind` = "explicit" indicates that irrespective of which choice branch is found, the length

occupied by the choice is given by `dfdl:choiceLength`. Some languages, such as COBOL, require choices (REDEFINES in the COBOL language) to be the same length and `dfdl:choiceLengthKind` "explicit" is used to model this.

Summary

In this lesson we have looked at how you can define alternative structures that might appear in the same place in the data stream. We also saw how `dfdl:initiatedContent` can be used to decide which alternative is present.

When no initiators are present in the data stream, DFDL provides an alternative construct called a ‘discriminator’ which can be used to decide which alternative is present. Discriminators are described in a later lesson.