DFDL Introduction for Beginners

Lesson 5: Modeling Alternative Structures

Version	**Author**	**Date**	**Change**
1	A Powell	2011	Created
2	S Hanson	2011-03-30	Updated
3	S Hanson	2013-09-03	Match latest spec

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', and may be an element, a sequence or another choice.

When parsing a data stream and a choice is encountered it must be possible to discover which of the alternatives is present. By default, 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 'speculative parsing', and the act of trying the next branch after a failure is sometimes called 'backtracking'.

Note that it is necessary for the parser to be able to distinguish between a parsing error caused by the trying the wrong choice branch and a parsing error caused by trying the correct choice branch but the data being badly formed. The former condition should cause backtracking, but the latter condition should not. There are several mechanisms provided by DFDL that allow you to make a positive assertion that the correct choice branch has been found, and that subsequent processing errors will not cause backtracking.

Using initiators
Sometimes every branch of the choice starts with an initiator. If so the dfdl:initiatedContent property of the choice can be set to ''yes" to indicate that if the initiator is found, then that choice branch is deemed to be present. If a subsequent component in that choice branch fails to parse then it is a real parsing error in that choice branch and no backtracking takes place.

Using direct dispatch
Sometimes the branch to take is indicated by an earlier field in the data stream. If so then the dfdl:choiceDispatchKey property of the choice can be used to look at the indicator, and match it to the dfdl:choiceBranchKey property of one of the branches. If a match is found then that choice branch is deemed to be present. If a subsequent component in that choice branch fails to parse then it is a real parsing error in that choice branch and no backtracking takes place. The dfdl:choiceDispatchKey property is a DFDL expression. DFDL expressions are covered in lesson 12.

Using discriminators
Sometimes the parser can only know it is in the correct branch when it has parsed some of the data in the branch. For example, an identifier field exists but it does not occur at the start of the branch. If so then a discriminator annotation can be used. This contains a condition to evaluate which, if successful, means that the choice branch is deemed to be present. If a subsequent component in that choice branch fails to parse then it is a real parsing error in that choice branch and no backtracking takes place. The discriminator condition can either be a DFDL expression or a regular expression. Discriminators are covered in detail in lesson 14.

If none of the above techniques is used, and a component in a choice branch causes a parsing error, then backtracking takes place and the next branch is tried.

Modeling alternative data structures

Building on the Address example from lesson 4, we introduce an alternative which 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

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

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

```
<xs:element name="address" dfdl:lengthKind="implicit"
    dfdl:initiator="[" dfdl:terminator"]">
  <xs:complexType>
      <xs:sequence dfdl:sequenceKind="ordered"
           dfdl:separator="*"
           dfdl:separatorPosition="infix"
           dfdl:separatorPolicy="required">

         <xs:element name="houseNumber"
             type="xs:string"
             dfdl:initiator="house:" />
         <xs:element name="street" type="xs:string"
             dfdl:initiator="street:" />
         <xs:element name="city" type="xs:string"
             dfdl:initiator="city:" />

         <xs:element name="jurisdiction"
             dfdl:lengthKind="implicit">
            <xs:complexType>
              <xs:choice
                  dfdl:choiceLengthKind="implicit"
                  dfdl:initiatedContent="yes">

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

              <xs:choice/>
            </xs:complexType>
         </xs:element>

       </xs:sequence>

   </xs:complexType>
</xs:element>

</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 all the choice branches must have an initiator defined, and that the initiators on their own should be used to decide which is present. The parser will look for the "state:" initiator in the data 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[1], require all choice branches 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.

[1] The COBOL language construct for a choice is called REDEFINES