Encoded Words: A Problem for DFDL

An email message has a subject header:

```
Subject: Hello World!
```

This header can be modeled easily in DFDL:

```
<xsd:element
      dfdl:initiator="Subject:%WSP*;"
      dfdl:terminator="%NL;"
      name="Subject"
      type="UnstructuredText">
</xsd:element>

<xsd:simpleType name="UnstructuredText">
  <xsd:restriction base="xsd:string">
    <xsd:pattern
      value="[ !&quot;#$%&amp;'()*+,-./0-9:;&lt;=&gt;?@A-Z\[\\\]^_`a-z{|}~]+" />
  </xsd:restriction>
</xsd:simpleType>
```

This schema simply restricts the set of valid characters in the subject.

RFC 2047, one of the MIME specifications, details a feature known as an encoded word. An encoded word allows non-ASCII characters to be used in email headers. It specifies both a text encoding (e.g., UTF-8) and a transfer encoding (e.g., Base64 or quoted printable [QP]), which allows the SMTP server to transmit the header and informs the receiving email client how to decode it for display.

The syntax of an encoded word is this:

```
encoded-word = "=?" charset "?" encoding "?" encoded-text "?="
```

So suppose my subject is in Spanish:

```
Subject: ¡Hola Señor Juan!
```

This subject is not normally allowed, but it is if we first convert it to an encoded word. We could use QP:

```
Subject: =?UTF-8?Q?=C2=A1Hola_Se=C3=B1or_Juan!?=
```

Or we could use Base64:

```
Subject: =?UTF-8?B?wqFIb2xhIFNlw7FvciBKdWFuIQ==?=
```

Email clients (including Outlook, Thunderbird, Apple's Mail, and Gmail) know how to decode and display this subject.

To make things even more complex, clients do not have to encode the entire header; they are free to encode as much or as little as necessary. And they can include multiple encoded words in a header.

Question: Can DFDL handle this?

We need DFDL to do 3 things:

1. Detect the presence of one or more encoded words.
2. Decode the words, whether in Base64- or QP-encoded.
3. Parse the **entire header**.

I don’t think DFDL can handle this. I think it comes up short in two areas:

1. It can’t decode Base64 or QP.
2. Even if it could, it doesn’t have a mechanism for multiple passes. In other words:
    a. The first pass through the subject is to decode one or more encoded words.
    b. The second pass is to combine them together with any plain text parts to create the complete subject.
    c. The third pass is continue parsing as normal (see Hello World!) above.

Am I correct?