Issues to Address When Using DFDL to Parse IPFIX Messages

Internet Protocol Flow Information Export (IPFIX) is a flexible and extensible data format that network devices such as routers and switches use to report flow information. The IPFIX standard defines both a template record and a data record. A template record describes the format of data records. A network device periodically sends a template record to a collector telling it what data to expect and subsequently sends the data records to the collector for analysis.

If the collector does not know the format of a data record until the template record is received, then an application must dynamically generate the Data Format Description Language (DFDL) schema for the data record when it receives the template record. This process consists of the following steps:

* An application receives a template record and parses it to produce XML.
* The application analyzes the XML to generate a new DFDL schema for data records.
* When the application receives a similar data record in the future, the application parses it using the dynamically generated DFDL schema.

The following outlines a DFDL schema for parsing template and data records.

Create a single "master" DFDL schema. This master schema contains a choice with discriminators to select the proper schema code based on the input record's type and ID. The choice looks something like this:

<choice>

<element ...> <!-- parse template with ID=1 -->  
   ...<dfdl:discriminator> { (type eq 'template') and (id eq 1) } </dfdl:discriminator>...

</element>

<element ...> <!-- parse template with ID=2 -->  
   ...<dfdl:discriminator> { (type eq 'template') and (id eq 2) } </dfdl:discriminator>...

</element>

<element ...> <!-- parse data using schema generated from template with ID=1 -->  
   ...<dfdl:discriminator> { (type eq 'data') and (id eq 1) } </dfdl:discriminator>...

</element>

<element ...> <!-- parse data using schema generated from template with ID=2 -->  
   ...<dfdl:discriminator> { (type eq 'data') and (id eq 2) } </dfdl:discriminator>...

</element>

<element ...> <!-- parse new, never-before-seen template -->  
   ...<dfdl:discriminator> {(type eq 'template') and (id ne 1) and (id ne 2) } </dfdl:discriminator>...

</element>

<element ...> <!-- parse unknown data record as a hexBinary blob -->  
   ...<dfdl:discriminator> { (type eq 'data') and (id ne 1) and (id ne 2) } </dfdl:discriminator>...

</element>

</choice>

When the application encounters an unknown template with ID=*n*, it generates a new template-*n* element and a new data-record-*n* element, adds them to the choice, and recompiles the master schema to create a new master schema. The application uses this new master schema for parsing all future IPFIX messages. If an unknown data record is received, the application reparses it with an updated master schema.

An IPFIX collector might receive IPFIX messages from multiple network devices. The collector must keep track of which IPFIX messages came from which network device because each network device generates template IDs independently, so two network devices might generate IPFIX messages with the same template IDs but with different content. When it receives a data record, the collector must be careful to use the DFDL schema with the matching template ID *and* matching network device.