DFDL Enhancements

2022 TPF Users Group Conference
March 27-30, Dallas, TX
Web Services

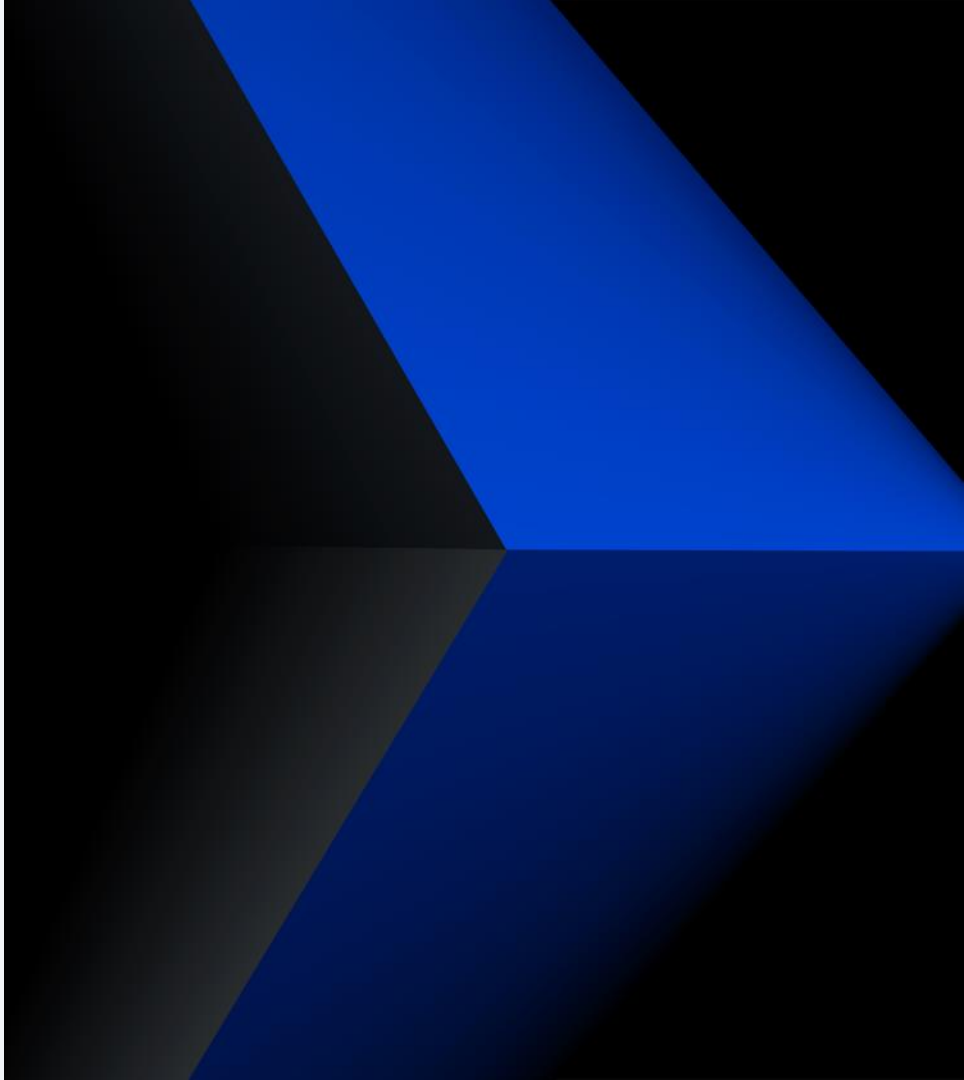
Bradd Kadlecik

IBM Z

IBM

Disclaimer

Any reference to future plans are for planning purposes only. IBM reserves the right to change those plans at its discretion. Any reliance on such a disclosure is solely at your own risk. IBM makes no commitment to provide additional information in the future.



Agenda

- Offline validation utility
- Custom formatting of document data
- Removing large document restrictions
- Conclusions
- What’s next

Agenda

- **Offline validation utility**
- Custom formatting of document data
- Removing large document restrictions
- Conclusions
- What’s next

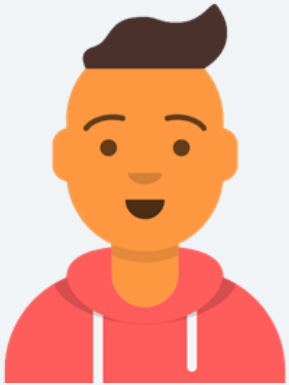
Problem Statement

Errors in DFDL schema files can be difficult to find and fix.

Users

Zach
application developer

We need to make a few updates to our new REST API. You'll be in charge of adding the new fields and updating all the artifacts.



Andrew
new hire application developer

Sounds easy enough. I'll get right on it.

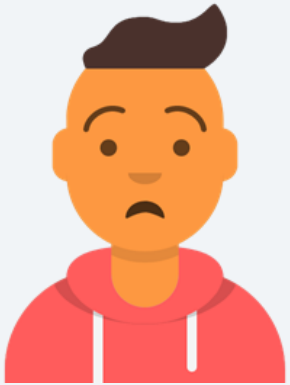
As-Is User Story

Andrew updates the C header and OpenAPI descriptor for the new fields and then tries to figure out how to modify the DFDL schema.

There’s a lot to understand about DFDL.

As-Is User Story

Andrew eventually figures out what looks like the right updates to make to the DFDL schema and tries to load it to z/TPF but gets an error.



I'll need to read about DFDL a bit more to understand this error.

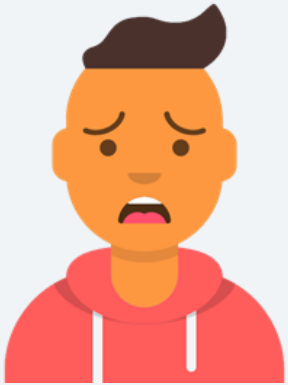
As-Is User Story

Andrew spends the rest of the day trying to figure out what to modify in the DFDL schema to make the error go away but each time he loads it he keeps getting an error.

This is really frustrating!

As-Is User Story

Andrew eventually asks Zach for help who, after a few tries himself, needs to open a case with IBM for help on the error. After a quick reply from IBM, they are able to get the DFDL successfully loaded to z/TPF.



This update is taking so much longer than I planned.

Pain Points

The DFDL schema can only be validated on z/TPF requiring a load after each change.

Error messages do not make it clear where the error is in the DFDL schema, resulting in it taking a lot more time to figure out and resolve.

It is not easy to understand which DFDL attributes are supported by z/TPF and which get ignored.

Value Statement

An application developer can debug DFDL validation errors by himself in a matter of minutes instead of days.

To-Be User Story

Andrew updates the C header and OpenAPI descriptor for the new fields and then tries to figure out how to modify the DFDL schema.

There’s a lot to understand about DFDL.

To-Be User Story

Andrew eventually figures out what looks like the right updates to make to the DFDL schema and uses the offline validation tool to quickly find and correct his errors.

The offline validation tool made it really easy to find what I needed to fix.

To-Be User Story

Andrew is able to let Zach know that he was able to finish making all the updates right on schedule.

I learned a lot about DFDL. It's not so intimidating anymore.

Technical Details – PJ46509 (Dec 2021)
z/TPF DFDL offline utility

- The z/TPF DFDL offline utility (tpfdatamap) provides DFDL schema validation that ensures the DFDL schema will be valid for loading to the z/TPF system.
- Warnings are provided for probable issues that should be fixed within the DFDL schema.
- All error and warning messages provide the line number within the DFDL schema.
- Shipped with the z/TPF product to run on Linux on Z.

Technical Details
DFDL validation warnings

1) A DFDL, XSD, TDDT attribute that is not part of the specification.
   Cause: Possible mistype of the attribute name
   System action: Attribute is ignored

2) A DFDL, XSD, TDDT attribute that is not allowed at the current location.
   Cause : Unfamiliarity of which attributes are allowed where
   System action : Attribute is ignored

3) Certain attribute combinations allowed by z/TPF but not valid per the DFDL specification.
   Cause : Unfamiliarity with the DFDL specification
   System action : Allowed for now but there is likely an accepted way to accomplish the same thing.

Example

> tpfdatamap verify packed.drvr.dfdl.xsd

DATAMAP0002E packed.drvr.dfdl.xsd:23

lengthKind error - value of emplicit is unsupported

DATAMAP0012W packed.drvr.dfdl.xsd:39

Unable to resolve namespace prefix for dfd1:length

DATAMAP0012W packed.drvr.dfdl.xsd:48

message is not implemented in z/TPF for the discriminator annotation

DATAMAP0001I Processing completed

Example

DATAMAP0002E packed.drvr.dfdl.xsd:23

lengthKind error - value of emplicit is unsupported

packed.drvr.dfdl.xsd, line 23:

<xs:element dfdl:lengthKind="emplicit" name="version_choice">

<xs:element dfdl:lengthKind="implicit" name="version_choice">

Example

DATAMAP0012W packed.drvr.dfdl.xsd:39

Unable to resolve namespace prefix for dfd1:length

packed.drvr.dfdl.xsd, line 39

<xs:element default="0" dfdl:binaryNumberRep="packed" dfd1:length="12" dfdl:lengthKind="explicit" dfdl:lengthUnits="bytes" name="packed5" type="xs:nonNegativeInteger"/>

<xs:element default="0" dfdl:binaryNumberRep="packed" dfdl:length="12" dfdl:lengthKind="explicit" dfdl:lengthUnits="bytes" name="packed5" type="xs:nonNegativeInteger"/>

Example

DATAMAP0012W packed.drvr.dfdl.xsd:48

message is not implemented in z/TPF for the discriminator annotation

packed.drvr.dfdl.xsd, line 48

<dfdl:discriminator message="">{../../version eq 1}</dfdl:discriminator>

<dfdl:discriminator>{../../version eq 1}</dfdl:discriminator>

Agenda

- Offline validation utility
- **Custom formatting of document data**
- Removing large document restrictions
- Conclusions
- What’s next

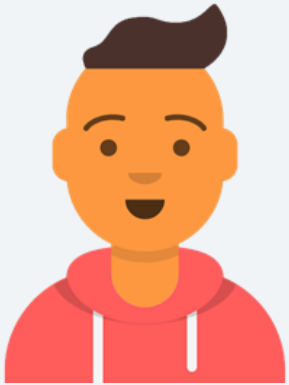
Problem Statement

DFDL doesn’t provide a way to format how the data appears in a document.

Users

Zach
application developer

We need you to change how one of the new fields you added appears in the JSON response.



Andrew
new hire application developer

Ok, I'll look into it.

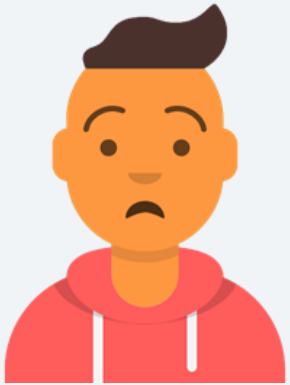
As-Is User Story

Andrew discovers the only way to customize the field value in the document is to change it to a string and format it prior to sending the response through DFDL.

Doesn’t seem too bad.

As-Is User Story

During further investigation he realizes that he can’t change the field used by the application and needs to copy the whole response structure to a new structure with the 1 field being a string.



So much work to fix just one field.

Pain Points

Changing the format may require copying all the data over to a new structure.

Customizing floating point values is complicated requiring the coding of a DFDL user exit function and adding document only fields for the transformation.

Value Statement

An application developer can easily customize how they want values to appear in documents created by DFDL.

To-Be User Story

Andrew discovers that he can use z/TPF unique attributes to describe how the new field should look. He is able to quickly report back to Zach to show how the field now looks in the response.

Nice and simple.

Technical Details – PJ46516 (Nov 2021)
DFDL support for formatting

- Values for float and double types can be customized using **tddt:floatingPointFormat**.
- Values for integer types can be customized using **tddt:integerFormat**.
- Values for dateTime types can be customized using **tddt:dateTimeFormat**.
- The z/TPF DFDL generation utility was updated to make decimal the default format for all floating point types (float and double) instead of scientific notation.
- REST provider and REST consumer support already use decimal notation by default if the OpenAPI descriptor specifies “number” as the type.

Technical Details
floating point format

```
<xs:element name="average" type="xs:float" dfdl:lengthKind="explicit"
dfdl:length="4" dfdl:lengthUnits="bytes" default="0"/>
```

```
"average":"7.500000e-01"
```

```
<xs:element name="average" type="xs:float" dfdl:lengthKind="explicit"
dfdl:length="4" dfdl:lengthUnits="bytes" tddt:floatingPointFormat="%f"
default="0"/>
```

```
"average":0.750000
```

Technical Details
integer format

```
<xs:element name="billNum" type="xs:unsignedShort" dfdl:lengthKind="explicit"
dfdl:length="2" dfdl:lengthUnits="bytes" default="0"/>
```

"billNum":100

```
<xs:element name="billNum" type="xs:unsignedShort" dfdl:lengthKind="explicit"
dfdl:length="2" dfdl:lengthUnits="bytes" tddt:integerFormat="%0.5lu"
default="0"/>
```

"billNum":00100

Technical Details
dateTime format

Additional binary definitions:

- **dfdl:binaryCalendarRep** specifies whether the binary value is in milliseconds or seconds.
- **tddt:dateTimeRep** specifies if the binary value is in TOD clock value format.
- **tddt:localTime** specifies if the binary value is using system local time or GMT.

Document format definitions:

- **tddt:dateTimeFormat** accepts formats according to the strftime function
- **tddt:includeMilliseconds** defines whether or not to include milliseconds.

Technical Details
dateTime format

A time_t definition:

```
<xs:simpleType name="time_t" dfdl:representation="binary"
dfdl:binaryCalendarRep="binarySeconds">
    <xs:restriction base="xs:dateTime"/>
</xs:simpleType>

<xs:element name="timestamp" type="time_t" dfdl:lengthKind="explicit"
dfdl:length="8" dfdl:lengthUnits="bytes" tddt:localTime="true"
tddt:includeMilliseconds="false" tddt:dateTimeFormat="%H:%M:%S on
%m/%d/%Y"/>
```

"timestamp":"18:30:20 on 12/1/2020"

Technical Details
dateTime format

A TOD clock definition:

<xsd:element name="todForm" type="xsd:dateTime" dfdl:length="8" tddt:dateTimeRep="tod" />

"todForm":"2021-11-02T19:18:05.076"

Agenda

- Offline validation utility
- Custom formatting of document data
- **Removing large document restrictions**
- Conclusions
- What’s next

Problem Statement

The tree structure used by the z/TPF parser is limited to 65,535 nodes per node type.

Pain Points

A JSON/XML document of several megabytes or larger cannot be parsed by the z/TPF parser.

The z/TPF parser is used by DFDL to serialize documents into data. Therefore, REST provider cannot read very large JSON requests and REST consumer cannot read very large JSON responses.

Value Statement

Allow z/TPF systems to process JSON/XML documents of several megabytes or larger.

Technical Details – PJ46708 (March 2022)
z/TPF parser support for greater than 64K nodes

- For JSON, element nodes contain JSON properties while content nodes contain JSON values.
- The limit for element, attribute, and content nodes is changed from 65,535 to 4,294,967,295.
- The nodes reside in 64-bit ECB heap for documents larger than 100K (PJ46135 Aug 2020).
- The tree structure created for XML documents should take up 20-50% less memory.
- The tree structure created for JSON documents will be built more efficiently by using much fewer reallocs but use a little more memory for larger documents.

Conclusion

PJ46516 (Nov 2021):

- Provides a way to easily customize document values such as floating point precision or date and time.

PJ46509 (Dec 2021):

- Provides a way to quickly validate and debug DFDL errors offline.

PJ46708 (March 2022):

- Provides the ability to parse very large JSON and XML documents.

What's next
Future possibilities

- Add pack option to C headers generated by OpenAPI code generation utility.
- DFDL offline library support
  - Perform offline test parse/serialize to validate data and document formats.
  - Use z/TPF DFDL APIs in a Linux application.
- Display for REST related artifacts
  - Find all artifacts for a REST API by operationId.

We want sponsor users!

Our development cycle is driven by your feedback.

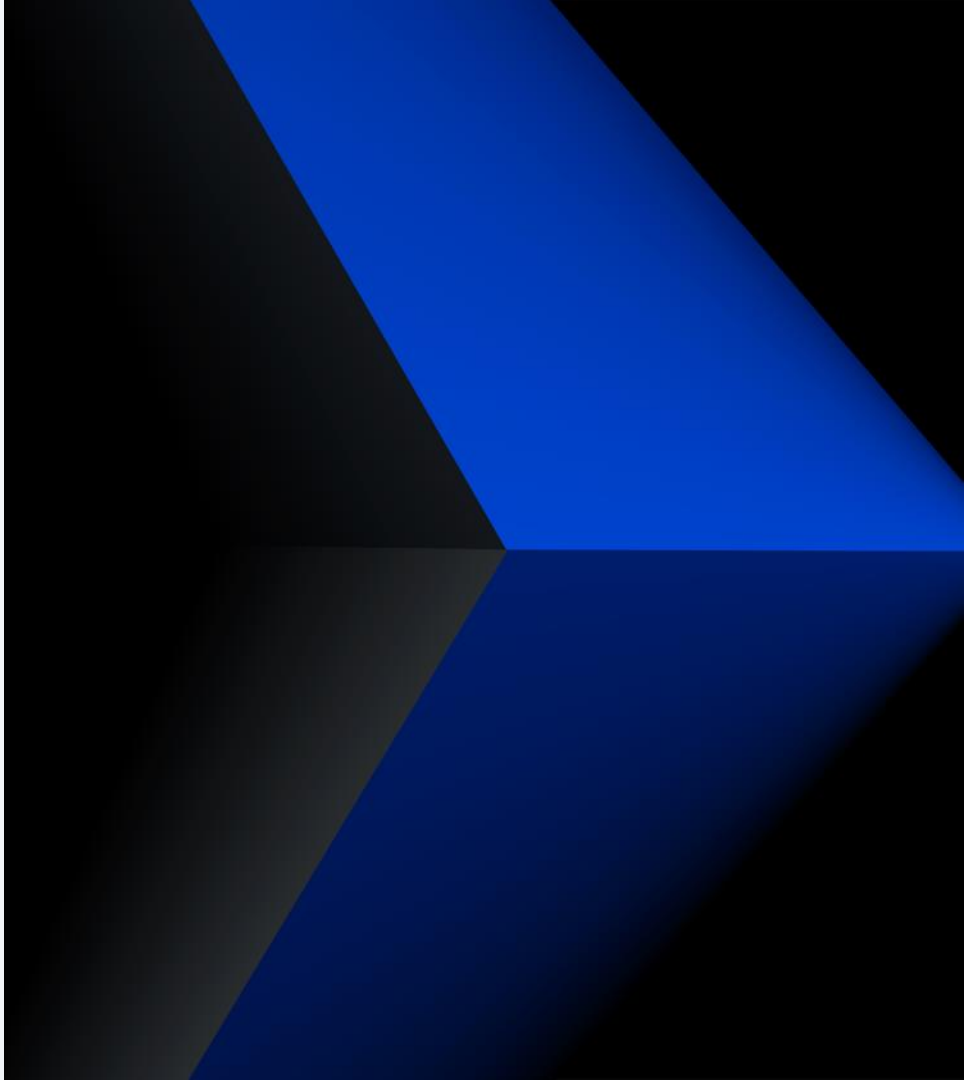
We are looking for sponsor users to assist in design and implementation, targeting the following personas:

- Operator
- Coverage programmer

We expect to begin engaging with the sponsor users in the 2nd half of 2022.

If you are interested in participating as a sponsor user, please contact:

Bradd Kadlecik (braddk@us.ibm.com)



Thank you