DFDL Enhancements

2023 TPF Users Group Conference
April 24-26, Dallas, TX
Web Services Subcommittee

—

Bradd Kadlecik

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

- Testing DFDL schemas on Linux on IBM Z
- Creating DFDL applications on Linux on IBM Z
- Using DFDL to validate data
- Conclusions
- What’s next

Agenda

- **Testing DFDL schemas on Linux on IBM Z**
- Creating DFDL applications on Linux on IBM Z
- Using DFDL to validate data
- Conclusions
- What’s next

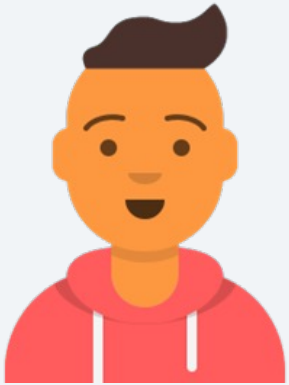
Problem Statement

Testing DFDL requires loading the DFDL files to z/TPF.

Users

Zach
Application developer

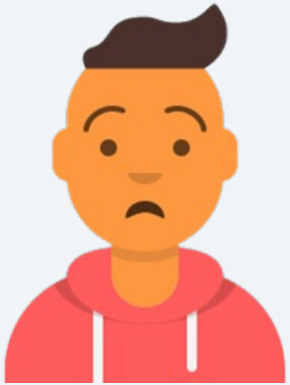
We'll be creating a new service to send data from z/TPF to other servers using DFDL.



Andrew
New hire application developer

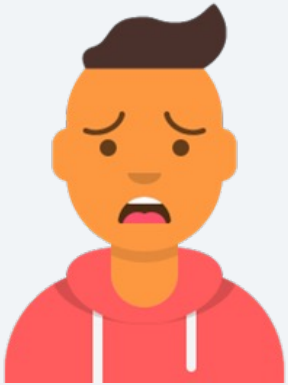
Great, what's DFDL?

As-Is User Story



Before Andrew can experiment with DFDL to understand it better, he needs to understand how to use the DFDL APIs, generate a DFDL schema, then create a program and data to load to z/TPF.

As-Is User Story



After creating everything, he runs the program to see what happens but has to modify and load the DFDL schema again each time he wants to see how things change.

Pain Points

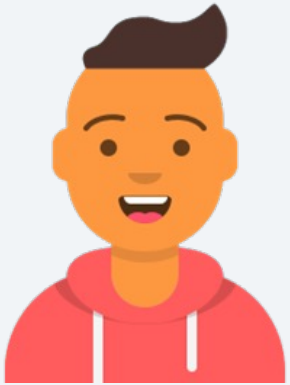
Code must be written before one can test a DFDL schema.

The DFDL parse and serialize operations can only be validated on z/TPF, which requires a load after each change.

Value Statement

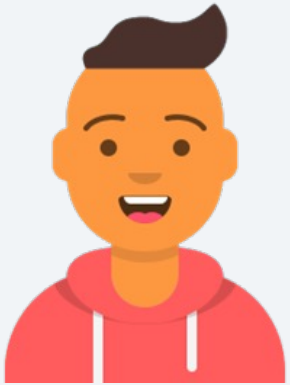
An application developer can learn about, develop, and debug DFDL in less time and with less difficulty.

To-Be User Story



After generating a DFDL schema, Andrew creates a data file that is used to easily create various document formats to understand better how DFDL works.

To-Be User Story



Andrew uses the DFDL created JSON file to recreate the data file using a DFDL serialize action and can now easily update either the JSON or DFDL schema for any modifications he wants to try out. He can also immediately see when his modification injects a runtime error or data mismatch.

Technical Details – PJ46801 (Nov 2022)

DFDL offline parser support

- The z/TPF DFDL offline utility (***tpfdatamap***) on Linux on IBM Z provides the same parse and serialize capabilities as z/TPF.
- The ***tpfdatamap*** utility is run from the same directory as ***maketpf***.
- The document format is determined by the following file extensions: .bson (BSON), .csv (CSV), .json (JSON), .properties (Java properties), .xml (XML).

Technical Details – PJ46801 (Nov 2022)

tpfdatamap syntax

tpfdatamap [parse|serialize] –r <root element> -i <input filename> -o <output filename> <DFDL schema filename>

- The filenames can be relative or absolute paths.
- The binary file can be any file extension.
- Pointers are automatically converted to and from offsets.
- See *man tpfdatamap* for more information.

Technical Details – PJ46801 (Nov 2022)
TPF_DFDL_PATH

- Additional DFDL schemas that are imported or included are located through the ***TPF_DFDL_PATH*** environment variable.
- The $TPF_ROOT/base/tpf-fdes directory is automatically added to find tpfbase.lib.dfdl.xsd.
- The directory of the specified DFDL schema file is also automatically added.

Technical Details – PJ46801 (Nov 2022)
TPF_DFDL_PATH

- ***TPF_DFDL_PATH*** is a colon separated list of DFDL schema directories.

  set TPF_DFDL_PATH="/ztpf/curdrv/dfdl:/ztpf/curdrv/xmla/dfdl/schemas"

- To make usage of tpfdatmap easier, you might want to set ***TPF_DFDL_PATH*** in a system or user profile on Linux on IBM Z for all application DFDL schema directories.

Technical Details – PJ46801 (Nov 2022)
Examples

Serialize: JSON -> binary

```
>tpfdatamap s -i stdhd.json -o stdhd.bin -r stdhd ../stdhd4.gen.dfdl.xsd
DATAMAP0001I Processing completed
```

Parse: binary -> XML

```
>tpfdatamap p -i stdhd.bin -o stdhd.xml -r stdhd ../stdhd4.gen.dfdl.xsd
DATAMAP0001I Processing completed
```

Technical Details – PJ46801 (Nov 2022)
Error Example

Serialize: JSON -> binary

```
>tpfdatamap s -i stdhd.json -o stdhd.bin -r stdhd ../stdhd4.gen.dfdl.xsd

DATAMAP0008E stdhd4.gen.dfdl.xsd
  Missing required field. Path name: /stdhd/stdchk
DATAMAP0001I Processing completed
```

Agenda

- Testing DFDL schemas on Linux on IBM Z
- **Creating DFDL applications on Linux on IBM Z**
- Using DFDL to validate data
- Conclusions
- What’s next

Value Statement

The z/TPF DFDL parser can be used to transform data in applications running on Linux on IBM Z.

DFDL offline parser support
Possible uses

- Transform z/TPF data either on z/TPF or on Linux on IBM Z.
- Convert trace, logging, or debug data to a more displayable format such as CSV.
- Easily create JSON or XML payloads to send to other systems for data residing on Linux on IBM Z.

Technical Details – PJ46801 (Nov 2022)
DFDL offline parser support

- All z/TPF DFDL APIs can be used on Linux on IBM Z except for tpf_dfdl_serializeData (use tpf_dfdl_serializeDoc) and tpf_dfdl_parseData (use tpf_dfdl_buildDoc).
- Linux applications can link to the tpfdfdl.so offline DFDL library. (see tpfdatamap.mak for an example)
- The TPF_DFDL_PATH environment variable must be set to include all referenced DFDL schemas before calling tpf_dfdl_initialize_handle.

Technical Details – PJ46801 (Nov 2022)
Offline DFDL APIs

- The *tpf_dfdl_serializeDoc* function has a **TPF_DFDL_UNORDERED** option to remove the restriction of having the document order match the DFDL schema order (useable either online or offline).
- The *tpf_dfdl_initialize_handle* function creates the DFDL metadata (handled by common deployment on z/TPF). Subsequent calls in the same process for the same DFDL schema will reference the same metadata.

Agenda

- Testing DFDL schemas on Linux on IBM Z
- Creating DFDL applications on Linux on IBM Z
- **Using DFDL to validate data**
- Conclusions
- What’s next

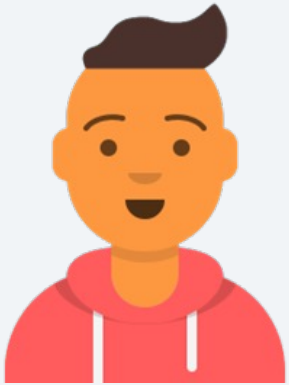
Problem Statement

Code must be written to restrict what data values are allowed after each DFDL serialize operation.

Users

Zach
Application developer

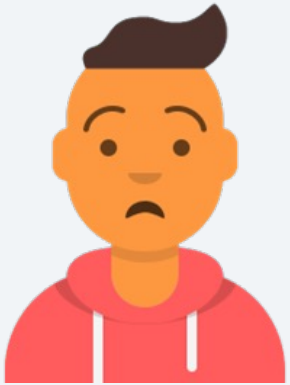
We’re going to create a REST service to update one of our databases on z/TPF.



Andrew
New hire application developer

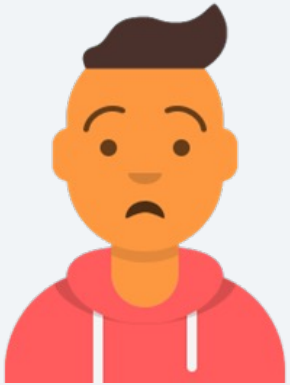
Ok

As-Is User Story



Andrew learns that data for many fields needs to be validated before the database gets updated. It takes time to investigate and learn what the allowed values are, sometimes checking other programs for what validation they do before updating the database.

As-Is User Story



He slowly puts all the validation code together with scenarios to handle what to do for each validation error. The test effort will need to test the validation for each field as there’s no common code.

Pain Points

There’s no data description that includes a description of what data values are accepted.

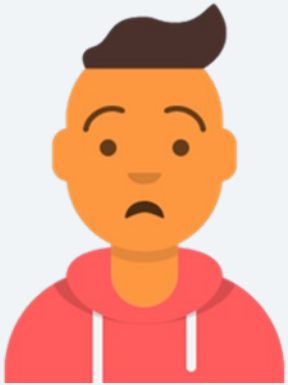
Each time a new routine is needed for receiving data on z/TPF, more validation code needs to be written.

There’s no mechanism for creating common validation routines.

Value Statement

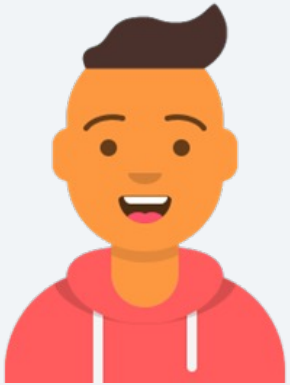
A software developer can validate that data conforms to what is required across multiple mediums without having to write code by using XML schema restrictions.

To-Be User Story



Andrew learns that data for many fields needs to be validated before the database gets updated. It takes time to investigate and learn what the allowed values are, sometimes checking other programs for what validation they do before updating the database.

To-Be User Story



Andrew updates the DFDL schema to contain all the validation logic to make it much easier on the next person. He then only needs to write code to update the database instead of worrying about all the data error scenarios.

Technical Details – PJ46951 (April 2023)
DFDL support for XML schema validation

The following XML schema facets are supported:

- enumeration – Defines a list of acceptable values (all types except boolean and calendar types).
- fractionDigits – Specifies the maximum number of decimal places allowed (decimal).
- minExclusive, minInclusive – Specifies the lower bounds for numeric values (all numeric types – float, double, decimal, integer variants).
- maxExclusive, maxInclusive – Specifies the upper bounds for numeric values (all numeric types – float, double, decimal, integer variants).

Technical Details – PJ46951 (April 2023)
DFDL support for XML schema validation

- minLength – Specifies the minimum number of characters allowed (string and hexBinary).
- maxLength – Specifies the maximum number of characters allowed (string and hexBinary).
- pattern – Defines the exact sequence of characters that are acceptable (string).
- totalDigits – Specifies the number of significant digits allowed (decimal and integer variants types).

Technical Details – PJ46951 (April 2023)
DFDL support for XML schema validation - example

```
<xs:element dfdl:length="4" dfdl:lengthKind="explicit" dfdl:lengthUnits="bytes"
dfdl:nilKind="literalCharacter" dfdl:nilValue="%NUL;" dfdl:useNilForDefault="no"
name="stdpgm" nillable="true" type="xs:string"/>
```

```
<xs:element dfdl:length="4" dfdl:lengthKind="explicit" dfdl:lengthUnits="bytes"
dfdl:nilKind="literalCharacter" dfdl:nilValue="%NUL;" dfdl:useNilForDefault="no"
name="stdpgm" nillable="true" type="pgm"/>

 <xs:simpleType name="pgm">
    <xs:restriction base="xs:string">
       <xs:pattern value="[A-Z][A-Z0-9][A-Z0-9][A-Z0-9]"/>
    </xs:restriction>
</xs:simpleType>
```

Technical Details – PJ46951 (April 2023)
DFDL support for XML schema validation

- Adding XML schema facets to DFDL schemas does not automatically enforce data validation (this is optional and must be specified).
- The z/TPF DFDL offline utility (***tpfdatamap***) has data validation turned on for both parse and serialize operations.
- The following DFDL functions support a **TPF_DFDL_VERIFY** option to enable data validation using XML schema facets:

tpf_dfdl_buildDoc
tpf_dfdl_createData
tpf_dfdl_parseData
tpf_dfdl_readData
tpf_dfdl_serializeData
tpf_dfdl_serializeDoc

Technical Details – PJ46951 (April 2023)
OpenAPI validation properties

- enum
- exclusiveMaximum
- exclusiveMinimum
- maximum
- maxItems
- maxLength
- minimum
- minItems
- minLength
- pattern

Technical Details – PJ46951 (April 2023)
z/TPF service descriptor updates

You can choose to have validation defined in either DFDL or the OpenAPI.

- OASValidation – Specify whether OpenAPI validation properties should be used to validate the data (either request or reply)
- DFDLValidation – Specify whether DFDL XSD validation should be used to validate the data (either request or reply)

Conclusion

PJ46801 (Nov 2022): DFDL offline parser support

- An application developer can learn about, develop, and debug DFDL in less time and with less difficulty.
- The z/TPF DFDL parser can be used to transform data in applications running on Linux on IBM Z.

PJ46951 (April 2023): DFDL support for XSD validation

- A software developer can validate data conforms to what is required across multiple mediums without having to write code by using XML schema restrictions.

What’s next
Future possibilities

- Improve performance of large choice branches through choiceDispatchKey and choiceBranchKey.
- Support serialization of CSV documents.
- Support a display operation for tpfdatamap to display offsets of DFDL elements.

Thank you