DFDL Enhancements

SOA Subcommittee

Bradd Kadlecik
z/TPF Development

IBM **z/TPF**
April 11, 2016

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.

5 Minutes	**Intro**
15 Minutes	**What's new**
5 Minutes	**What's next**
5 Minutes	**Q&A**

What is DFDL?

- Data Format Description Language
- A standardized way of describing data

  - or -

- A universal, shareable, non-prescriptive description for general text (CLOB) and binary data (BLOB) formats

Why should I care about DFDL?

XML/JSON rendering

tpf_doc_buildDocument

tpf_dfdl_parseData

XML/JSON serialization

tpf_doc_parseDocument

tpf_dfdl_serializeData

2014 – Data Events

- Generate DFDL for TPFDF lrecs (ZUDFM)
- DFDL files enabled by simply loading them (common deployment)
- DFDL APIs to create XML/JSON for BLOBs

2015 – z/TPF support for MongoDB

- Enhance DFDL support for serialization
- Improve performance

2016 – REST interface

- Generate DFDL for C structures via maketpf (PJ43440)
- Generate DFDL for assembler DSECTs via maketpf (HLASM update)
- DFDL APIs to create BLOBs from XML/JSON

So what’s new?

2014 supported DFDL attributes

- choiceLength
- choiceLengthKind
- length
- lengthKind
- lengthUnits
- occursCount
- occursCountKind

2015 supported DFDL attributes

- alignmentUnits
- binaryDecimalVirtualPoint
- binaryNumberRep
- encoding
- leadingSkip
- nilKind
- nilValue
- ref
- representation
- textPadKind
- textStringPadCharacter
- textTrimKind
- trailingSkip
- truncateSpecifiedLengthString
- useNilForDefault

So what does that all mean?

Unicode strings

ブラード

Unicode strings

- UTF-8 strings can be defined by setting the DFDL encoding attribute to “utf-8” for any string type. (PJ43302)

Unicode strings

Packed decimal format

Decimal numbers

Data gaps

Optional fields

Padding/trimming

Packed decimals

“1234”

01234C

Unicode strings

Packed decimal format

Decimal numbers

Data gaps

Optional fields

Padding/trimming

Packed decimals

- Packed decimal numbers can be defined by setting the DFDL binaryNumberRep attribute to “packed” for any decimal or integer types. (PJ43719)

Unicode strings

Packed decimal format

Decimal numbers

Data gaps

Optional fields

Padding/trimming

Decimal numbers

"1.234"

1234

Decimal numbers

- Decimal numbers can be defined by using the DFDL binaryDecimalVirtualPoint attribute to specify the position of the decimal point for a decimal type. (PJ43719)
  - ex: For the value of 1234 to be represented as “1.234”, the binaryDecimalVirtualPoint attribute would have the value of 3.

Unicode strings

Packed decimal format

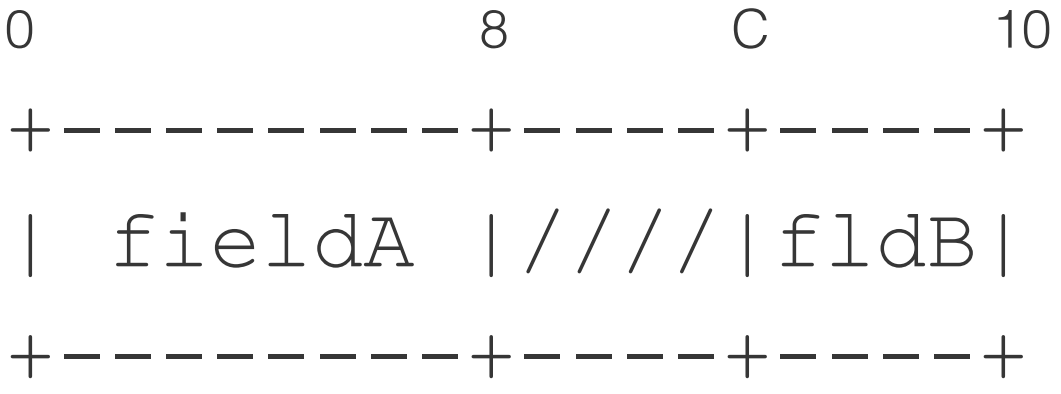
Decimal numbers

Data gaps

Optional fields

Padding/trimming

Data gaps



Data gaps

- Gaps in the data definition due to alignment and such can be represented using the DFDL leadingSkip or trailingSkip attributes. (PJ43302)
  - The DFDL alignmentUnits attribute controls whether the gap is defined in bits or bytes.

Optional fields

```
{
  a: 10,
  b: 5
}
```

a b

000A0005

```
{
  a: 10
}
```

000A0000

Optional fields

- For purposes of serialization, fields can be defined as optional by using either the XSD default or DFDL useNilForDefault attributes. (PJ43302)
  - Currently only a value of “0” is supported for default, while the nil value is set through the DFDL nilValue and nilKind attributes. (PJ43719)
  - Variable length fields cannot have default values.

Unicode strings

Packed decimal format

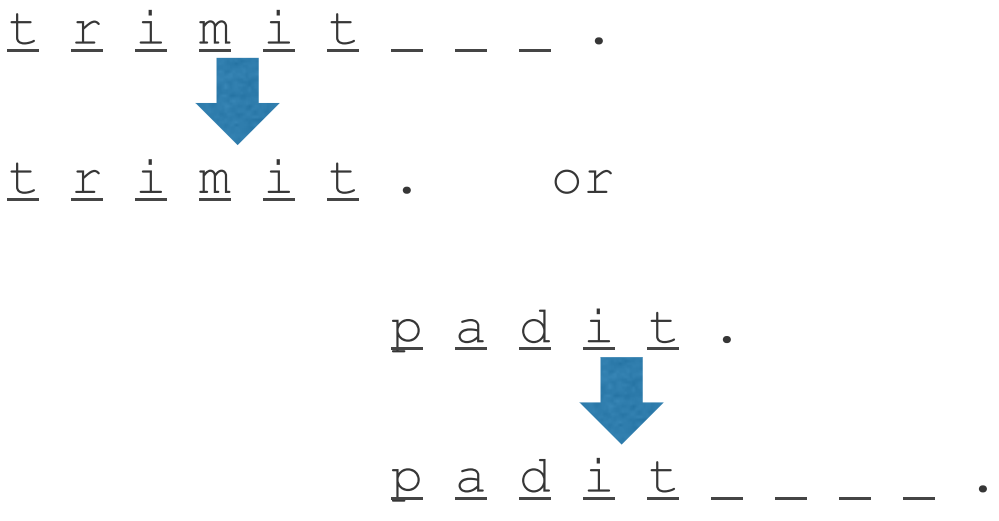
Decimal numbers

Data gaps

Optional fields

Padding/trimming

Padding/trimming



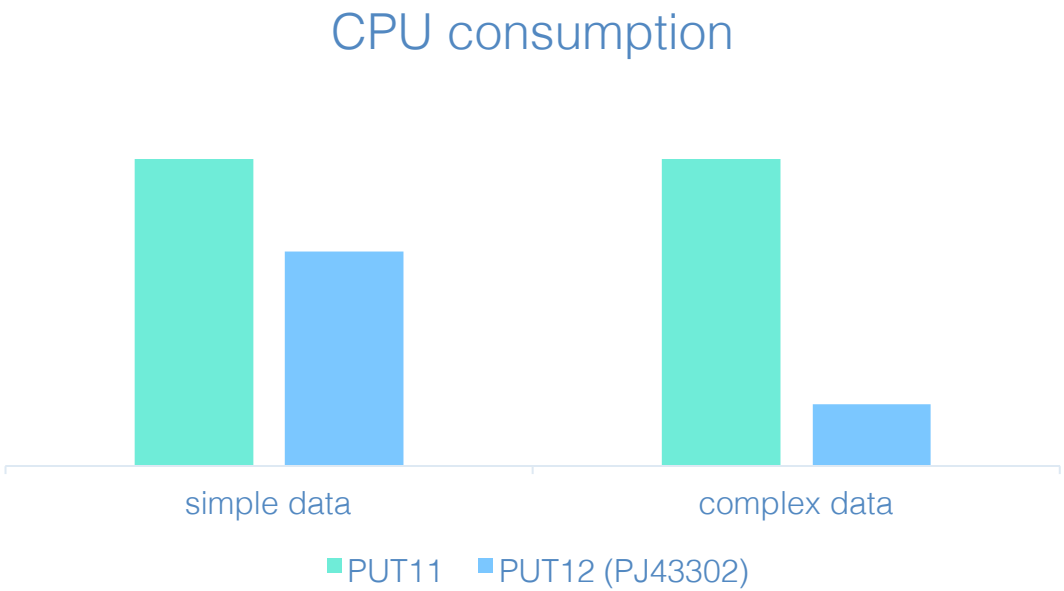
Padding/trimming

- Padding and trimming of string types can be performed using the DFDL textStringPadCharacter attribute. (PJ43302)
  - Trimming will be performed during parse by setting the DFDL textTrimKind attribute to “padChar”.
  - Conversely, padding will be performed during serialization by setting the DFDL textTrimKind attribute to “padChar”.

DFDL APIs understand XPATH

- Absolute XPATHs accepted on the following DFDL APIs:
  - tpf_dfdl_getElementInfo
  - tpf_dfdl_serializeData (future)
- Can specify the element name as:
  - “stdpgm”
  - “/stdhd/stdpgm”
- Delivered with PJ43302.

Performance improvement for XML/ JSON rendering from a BLOB



Can I generate DFDL for something other than TPFDF lrecs?

C struct to DFDL converter

C

```
typedef struct {
} mytype;
struct mystruct {
};
```

DFDL

```
<complexType
name="mytype">
<group name="mystruct">
```

tpf/c_stdhd.h

```
struct stdhd                          /* TPF's standard header          */
{                                     /*                                */
  unsigned char stdbid[2];            /* Record ID                      */
  unsigned char stdchk;               /* Record code                    */
  unsigned char stdctl;               /* Data control                   */
  unsigned char stdpgm[4];            /* Program ID                     */
  unsigned int  stdfch;               /* Forward Chain Field            */
  unsigned int  stdbch;               /* Backward Chain Field           */
}
```

stdhd.gen.dfdl.xsd

```
<xs:group name="stdhd">
   <xs:sequence>
      <xs:element name="stdbid" type="xs:string" dfdl:lengthKind="explicit" dfdl:length="2"
      dfdl:lengthUnits="bytes" nillable="true" dfdl:useNilForDefault="yes"
      dfdl:nilKind="literalCharacter" dfdl:nilValue="%NUL;" />
      <xs:element name="stdchk" type="xs:unsignedByte" dfdl:lengthKind="explicit"
      dfdl:length="1" dfdl:lengthUnits="bytes" default="0" />
      <xs:element name="stdctl" type="xs:unsignedByte" dfdl:lengthKind="explicit"
      dfdl:length="1" dfdl:lengthUnits="bytes" default="0" />
      <xs:element name="stdpgm" type="xs:string" dfdl:lengthKind="explicit" dfdl:length="4"
      dfdl:lengthUnits="bytes" nillable="true" dfdl:useNilForDefault="yes"
      dfdl:nilKind="literalCharacter" dfdl:nilValue="%NUL;" />
      <xs:element name="stdfch" type="xs:unsignedInt" dfdl:lengthKind="explicit"
      dfdl:length="4" dfdl:lengthUnits="bytes" default="0" />
      <xs:element name="stdbch" type="xs:unsignedInt" dfdl:lengthKind="explicit"
      dfdl:length="4" dfdl:lengthUnits="bytes" default="0" />
   </xs:sequence>
</xs:group>
```

maketpf integration (PJ43440)

- New “dfdl” target to generate DFDL
  - ex: maketpf qzz9 dfdl
- New TPF_DFDL_DIR maketpf environment variable to control what directory DFDL files get generated in
- Generated file names have the format of:
  <struct or typedef name>.gen.dfdl.xsd
- Do not support gcc 4.1 for C++

C data type	XML Schema type	Notes
signed char	byte	length=1
signed char[10]	byte	length=1 maxOccurs=10
char, unsigned char	unsignedByte	length=1
unsigned char[10]	string	length=10
int	int	length=4
int[2]	int	length=4 maxOccurs=2
unsigned int	unsignedInt	length=4
<any pointer type>	hexBinary	length=8

So what’s next?

XML/JSON serialization

tpf_doc_parseDocument

tpf_dfdl_serializeData

JSON -> binary

```
{
   stdhd: {
      stdbid: "BD",
      stdchk: 1,
      stdctl: 0,
      stdpgm: "ABCD",
      stdfch: 0,
      stdbch: 0
   }
}
```

```
C2C40100C1C2C3C4
0000000000000000
```

XML/JSON subset serialization

tpf_doc_parseDocument

tpf_dfdl_serializeData

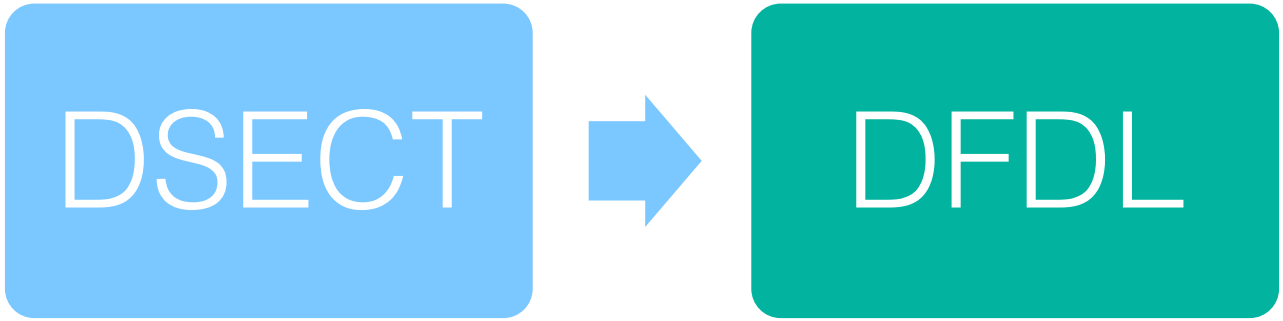
JSON -> binary

```
{
   stdpgm: "DFDL"
}
```

```
/stdhd/stdpgm
```

```
C2C40100C4C6C4D3
0000000000000000
```

DSECT to DFDL converter



DFDL from DSECTs

- HLASM will be updated soon to produce DFDL for assembler DSECTs
- maketpf will be updated for assembler segments to generate DFDL using the existing “dfdl” target.

Value of DFDL

- Provides the ability to convert z/TPF binary data to XML/JSON on or off platform and vice versa.
- Integrated with new distributed technologies like data events and z/TPF support for MongoDB.
- A powerful tool that continues to be enhanced and leveraged by z/TPF in new ways.

Thank you!

Questions or comments?

Trademarks

- IBM, the IBM logo, ibm.com and Rational are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Notes

- Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.
- All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.
- This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.
- All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.
- Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.
- Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.
- This presentation and the claims outlined in it were reviewed for compliance with US law. Adaptations of these claims for use in other geographies must be reviewed by the local country counsel for compliance with local laws.