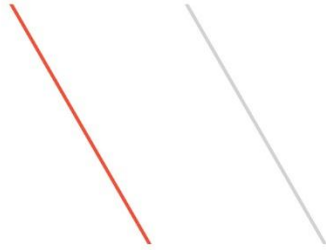


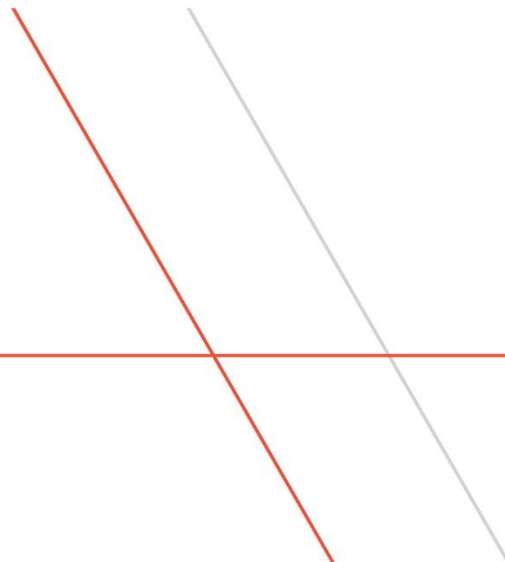
IBM z Systems



TPFUG – Defining TPF Data in DFDL

Bradd Kadlecik, TPF Development Lab

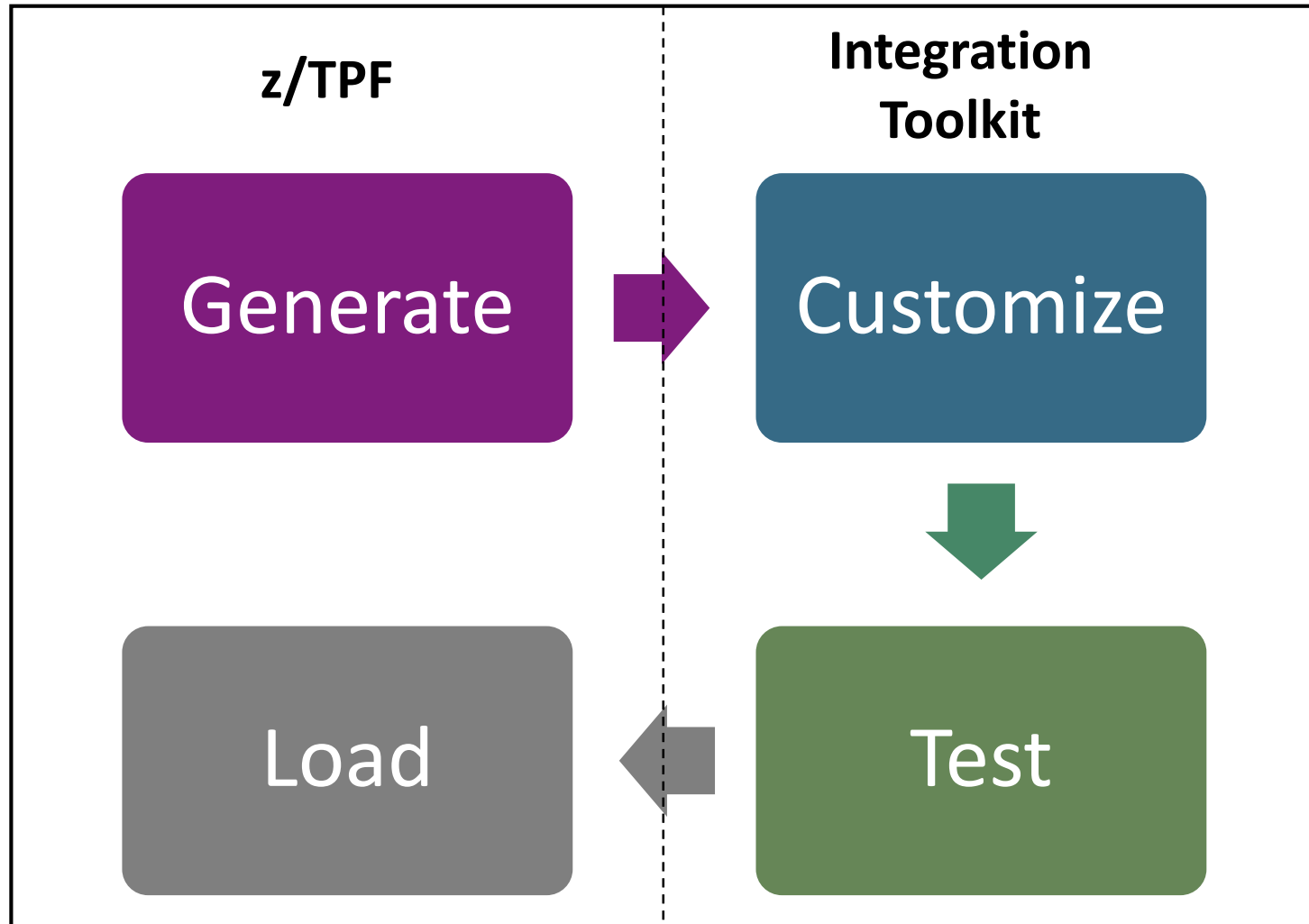
Mar 24, 2015



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.

How to define TPFDF data using DFDL



Step 1: Generating DFDL for TPFDF

ZUDFM DESCRIPTOR

- Uses MLS information to create DFDL information.
- Future: method for creating DFDL for find/file.
- Creates a DFDL file (.tpfdf.dfdl.xsd extension) in UTF-8 encoding to ftp in binary mode.

- **Example:**

```
zudfm descr file-dr26bi
CSMP0097I 10.51.41 CPU-B SS-BSS SSU-HPN IS-01
UDFM0561I 10.51.41 DR26BI FILE DESCRIPTOR BUILD STARTED
/etc/ztpfdf/descr/DR26BI.tpfdf.dfdl.xsd
UDFM0562I 10.51.41 DR26BI FILE DESCRIPTOR BUILD COMPLETE.
FILES CREATED.
```

Step 2: Customizing DFDL

- Create meaningful element names
- Verify generated data types
- Create discriminators for conditional data
- Create expressions for variable length fields
- Create expressions for variable size arrays
- Customizations when using DFDL outside z/TPF

Before you begin

- A DFDL editor is included with the Integration Toolkit, part of IBM Integration Bus for Developers (no charge license)
 - Acquiring v9:
<http://www.ibm.com/developerworks/downloads/ws/wmbd/>
 - Enrolling in v10 open beta:
<http://ibm.biz/iibopenbeta>
- Create either a general project or data design project in the Toolkit. Other project types may also work.
- Import ../base/tpf-fdes directory on linux or copy the tpfbase.lib.dfdl.xsd file (shipped with z/TPF) to your DFDL working directory.

Create meaningful element names

- The generated element names are either generic names or obscure DSECT names.
- The element names in DFDL are what will appear in XML/JSON or how it will be referenced across platforms.
- complexType names can be left alone as they will not appear anywhere

TPFDF DSECT

```
DR26ORG&CG1 EQU *      START VARIABLE DATA PER LREC
.*
*****
*          PASSENGER NUMBER LOGICAL RECORD          *
*****
DR26PNO&CG1 DS  CL8  PASSENGER NUMBER
DR26E70&CG1 EQU *      END OF LOGICAL RECORD WITH KEY = X'70'
.*
          ORG DR26ORG&CG1
*****
*          PASSENGER NAME LOGICAL RECORD          *
*****
DR26PNA&CG1 DS  CL25 PASSENGER NAME
DR26E80&CG1 EQU *      END OF LOGICAL RECORD WITH KEY = X'80'
.*
          ORG DR26ORG&CG1
*****
*          ADDRESS LOGICAL RECORD          *
*****
DR26ADR&CG1 DS  0CL1 PASSENGER ADDRESS (VARIABLE 1-20)
DR26E90&CG1 EQU *      END OF LOGICAL RECORD WITH KEY = X'90'
```


ZUDFM DESCRIPTOR output

The screenshot displays the IBM Integration Toolkit interface. The title bar reads "DFDL Test - DFDL/DR26BI.tpdf.dfdl.xsd - IBM Integration Toolkit - C:\Users...". The menu bar includes "File", "Edit", "Navigate", "Search", "Project", "Run", "Window", and "Help". The toolbar contains various icons for file operations and development. A "Quick Access" search box is visible. The main workspace shows a tree view of the DFDL descriptor for "DR26BI.tpdf.dfdl.xsd". The tree view includes a "Test Parse Model" and a "Test Serialize Model". The "Test Serialize Model" is expanded to show a "choice" element containing several "lrec" elements: "lrec70", "lrec80", "lrec90", and "lrecA0". The "lrec90" element is highlighted in green, and its "string" type is highlighted in yellow. The "lrecA0" element is highlighted in blue. The "string" type is highlighted in yellow. The "lrec90" element is highlighted in green, and its "string" type is highlighted in yellow. The "lrecA0" element is highlighted in blue. The "string" type is highlighted in yellow.

Element Name	Type	Count 1	Count 2
choice		1	1
lrec70	LREC70	1	1
lrec80	LREC80	1	1
lrec90	LREC90	1	1
sequence		1	1
DR26ADR	string	1	1
lrecA0	LRECA0	1	1

Meaningful Names

DFDL Test - DFDL/PNR.tpdf.dfdl.xsd - IBM Integration Toolkit - C:\Users\IB...

File Edit Navigate Search Project Run Window Help

Quick Access Integration Development DFDL Test

PNR.tpdf.dfdl.xsd

Test Parse Model Test Serialize Model Show properties Show all sections Focus on selected

Element Name	Record Name	Count 1	Count 2
choice		1	1
PassengerNumberRecord	LREC70	1	1
PassengerNameRecord	LREC80	1	1
AddressRecord	LREC90	1	1
sequence		1	1
Address	string	1	1
FlightHistoryRecord	LRECA0	1	1

Verify generated data types

- The generated data types are best guess mappings from the DSECT types.
- Character data should be strings while non-numeric, non-character data should be hexBinary. Strings undergo character encoding changes when XML/JSON is created.
- Numeric data types can be either signed or unsigned but the assembler DSECT doesn't contain information on which should be used.
- A byte consisting of various test bits could be changed to a bit-wise representation, 8 boolean bit fields with a true/false assignment.

Verify data usage

The screenshot shows the IBM Integration Toolkit interface for a DFDL Test. The main window displays the test results for the file *PNR.tpdf.dfdl.xsd*. The results are presented in a table with columns for element name, data type, and usage counts. The 'ServiceCode' element is highlighted in green, indicating it is the current focus.

Element Name	Data Type	Usage Count 1	Usage Count 2
ServiceRecord	LRECC0	1	1
sequence		1	1
ServiceCode	hexBinary	1	1
choice		1	1
sequence		1	1
sequence		1	1
sequence		1	1

Change to numeric

The screenshot shows the IBM Integration Toolkit interface. The title bar reads "DFDL Test - DFDL/PNR.tpdf.dfdl.xsd - IBM Integration Toolkit - C:\Users\IBM_A...". The menu bar includes "File", "Edit", "Navigate", "Search", "Project", "Run", "Window", and "Help". The toolbar contains various icons for file operations and development. A "Quick Access" search bar is present. The main workspace shows a tree view of the test configuration for the file "*PNR.tpdf.dfdl.xsd".

At the top of the workspace, there are several tabs: "Test Parse Model", "Test Serialize Model", "Show properties", "Show all sections", and "Focus on selected". The "Test Parse Model" tab is active, displaying a tree view of the test configuration. The "ServiceCode" element is selected and highlighted in green. The "ServiceRecord" element is expanded, showing a "sequence" child. The "ServiceCode" element is expanded, showing a "choice" child, which is further expanded to show four "sequence" children.

Element	Type	Count	Count
ServiceRecord	LRECC0	1	1
sequence		1	1
ServiceCode	byte	1	1
choice		1	1
sequence		1	1
sequence		1	1
sequence		1	1

Create discriminators for conditional data

- Conditional data is data that is not always present as is often handled in an assembler DSECT through an ORG or in a C structure through a union. These types of varying data formats are handled in DFDL through what is known as a “choice branch”.
- DFDL discriminators define a test to be used when resolving a point of uncertainty such as choice branches or optional elements.
- In the following example, the ServiceCode will be used to determine which data layout follows: the special meal plan or the unaccompanied minor information.
- The expression used is supported as of PJ42994.

TPFDF DSECT

ORG DR26ORG&CG1

* Special Services Record - LREC C0 *

DR26SVC&CG1 DS X service code

* 1 - special meal

* 2 - children travel alone

* 3 - airport assistance

DR26SVO&CG1 EQU *

* special meal

DR26MC1&CG1 DS X special meal code for first meal

DR26MC2&CG1 DS X special meal code for the second meal

DR26MC3&CG1 DS X special meal code for the third meal

DR26MNT&CG1 DS 0CL1 additional notes

*

ORG DR26SVO&CG1

* children travel alone

DR26AGE&CG1 DS X child's age, in hex

DR26GEN&CG1 DS C gender; Male or Female

DR26CUS&CG1 DS 0CL1 custodian contact info

choice branch

The screenshot displays the IBM Integration Toolkit interface for editing a DFDL test model. The main window shows a tree view of the test model for the file *PNR.tpdf.dfdl.xsd. The tree structure is as follows:

- ServiceCode (byte)
- choice
 - sequence (highlighted)
 - meal1 (byte)
 - meal2 (byte)
 - meal3 (byte)
 - mealServiceNotes (string)
- sequence
- sequence

On the right side of the interface, there is a table for defining test conditions. The table has three columns: Test Kind, Test Condition, and Me. A link labeled "Add discriminator" is visible in the Test Condition column.

Test Kind	Test Condition	Me
	Add discriminator	

Meal plan if ServiceCode = 1

The screenshot displays the IBM Integration Toolkit interface for a DFDL test. The main window shows the test configuration for the file *PNR.tpdf.dfdl.xsd. The configuration is structured as follows:

- ServiceCode (byte)
- choice
 - sequence
 - meal1 (byte)
 - meal2 (byte)
 - meal3 (byte)
 - mealServiceNotes (string)
 - sequence

On the right side, the 'Asserts and Discriminators' panel is visible, showing a test condition:

Test Kind	Test Condition
expression	{./ServiceCode eq 1}

Unaccompanied minor info if ServiceCode = 2

The screenshot displays the IBM Integration Toolkit interface for a DFDL test. The main window shows a tree view of the test model for the file *PNR.tpdf.dfdl.xsd. The 'ServiceCode' element is selected, and its value is set to '2'. The test condition is defined as `{./ServiceCode eq 2}`.

Test Parse Model Test Serialize Model Hide properties Show all sections Focus on selected Show quick outline 1

Test Kind	Test Condition
expression	{./ServiceCode eq 2}

ServiceCode byte
choice
sequence
sequence
passengerAge byte
passengerGender string
custodianContact string
sequence

Create expressions for variable length fields

- A variable length field is typically some string where the length of the string may vary.
- An expression is used in DFDL to allow the DFDL parser to be able to calculate the length of a variable length field.
- The DFDL “length” attribute can either contain a number for a fixed length field or an expression for a variable length field.

Fixed length string

The screenshot shows the IBM Integration Toolkit interface for a DFDL test. The main window displays a tree view of the test model for the file *PNR.tpdf.dfdl.xsd. The 'Destination' element is selected, and its properties are shown in the right-hand pane.

Test Model Tree:

- AddressRecord (LREC90)
- FlightHistoryRecord (LRECA0)
 - sequence
 - NumFlights (byte)
 - FlightInfo
 - sequence
 - Flight
 - Origin (string)
 - Destination (string)**
- FactsRecord (LRECB0)
- ServiceRecord (LRECC0)

Destination (Element) Properties:

Property	Value
Length Kind	explicit
Length	3
Length Units	bytes

Below the properties table, there is a section for 'Sample Test Data'.

Variable length string

The screenshot displays the IBM Integration Toolkit interface for a DFDL test. The main window shows a tree view of the test model for the file *PNR.tpdf.dfdl.xsd. The 'Facts' element is selected, and its properties are shown in the right-hand pane.

Test Model Tree:

- lrecKey: hexBinary
- choice
 - PassengerNumberRecord: LREC70
 - PassengerNameRecord: LREC80
 - AddressRecord: LREC90
 - FlightHistoryRecord: LRECA0
 - FactsRecord: LRECB0
 - sequence
 - Facts: string**
 - ServiceRecord: LRECC0

Facts (Element) Properties:

Property	Value
Length Kind	explicit
Length	{../../LRECPrefix/size -6 -3}
Length Units	bytes

The right-hand pane also shows a 'Sample Test Data' section, which is currently collapsed.

Create expressions for variable size arrays

- A variable size array is a structure or field that can occur any (or N) number of times.
- An expression is used in DFDL to allow the DFDL parser to be able to calculate the number of occurrences of an element.
- The XML schema attributes of “minOccurs” and “maxOccurs” give the lower and upper bounds of the array.
- The DFDL “occursCount” attribute contains the expression for calculating the size of the array.

FlightInfo for every NumFlights

The screenshot displays the IBM Integration Toolkit interface for testing a DFDL file. The main window shows a tree view of the test configuration for *PNR.tpdf.dfdl.xsd. The 'FlightInfo' element is selected, and its properties are displayed in the right-hand pane.

Test Configuration Tree:

- AddressRecord (LREC90)
- FlightHistoryRecord (LRECA0)
 - sequence
 - NumFlights (byte)
- FlightInfo (selected)
 - sequence
 - Flight
 - Origin (string)
 - Destination (string)
- FactsRecord (LRECB0)
- ServiceRecord (LRECC0)

FlightInfo (Element) Properties:

Property	Value
Min Occurs	1
Max Occurs	unbounded
Occurs Count Kind	expression
Occurs Count	{../NumFlights}

Additional options shown include Representation Property, Asserts and Discriminators, and Sample Test Data.

Customizations when using DFDL outside z/TPF

- When choosing to transmit binary data defined by DFDL to other platforms, a number of other customizations can be considered.
- Examples:
 - String encodings other than EBCDIC
 - Field (element) alignment
 - Trimming/padding of strings
 - Changing the floating point representation
 - Changing XML/JSON numeric or date/time format

Step 3: Testing DFDL

- The DFDL editor in the IBM Integration toolkit can perform a DFDL parse and serialize test.
- A file containing the binary data can be used as input to the DFDL parse testing to create XML.
- A file containing XML can be used as input to the DFDL serialize test to create binary data.
- Future plans include creating a method to more easily extract binary data from z/TPF to use for DFDL parse testing.

Step 4: Loading DFDL

- DFDL is loaded to z/TPF using common deployment, making the DFDL information accessible to the system (DFDL APIs, Business Events, MongoDB, etc).
 - All DFDL files must have the following file extension: **.dfdl.xsd**
 - All DFDL files must be loaded to the following location on z/TPF:
/sys/tpf_pbfiles/tpf-fdes
- DFDL files are automatically deployed by common deployment. Files are verified and active during loadset activation.

References

- DFDL tutorials created by the DFDL Working Group at the Open Grid Forum
 - http://redmine.ogf.org/dmsf/dfdl-wg?folder_id=5485
- DFDL developerWorks tutorials
 - <http://ibm.biz/startdfdl>
- DFDL specification reference
 - <http://www.ogf.org/dfdl>

Trademarks

- IBM, the IBM logo, and ibm.com 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](http://www.ibm.com/legal/copytrade.shtml)” at www.ibm.com/legal/copytrade.shtml.
- *(Include any special attribution statements as required – see Trademark guidelines on <https://w3-03.ibm.com/chq/legal/lis.nsf/lawdoc/5A84050DEC58FE31852576850074BB32?OpenDocument#Developing%20the%20Special%20Non-IBM%20Tr>)*

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.