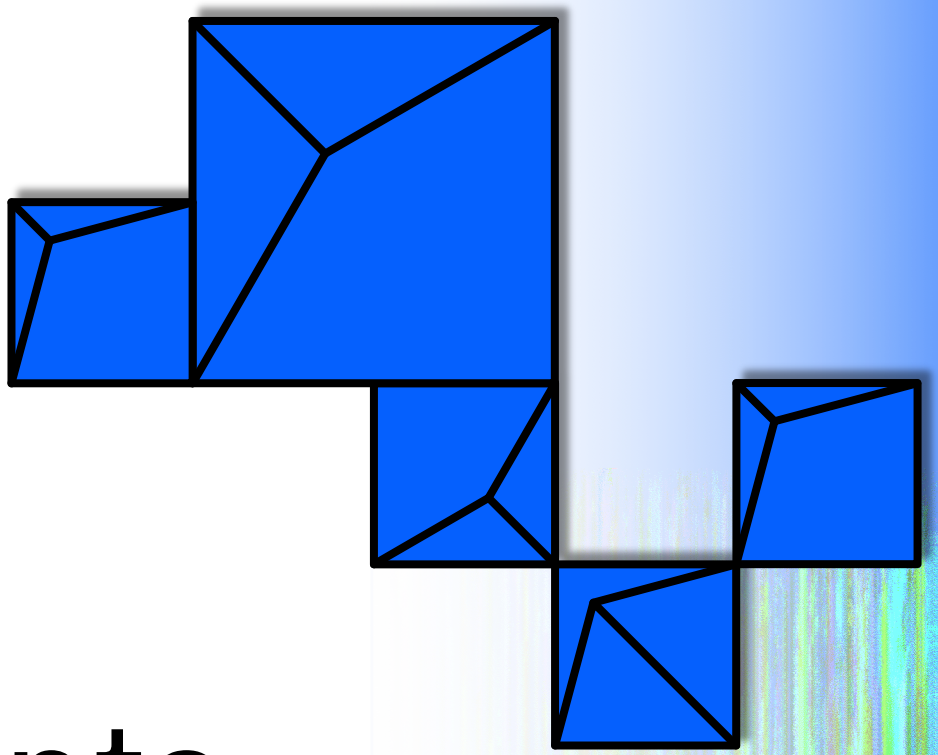


DFDL Enhancements

Bradd Kadlecik

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

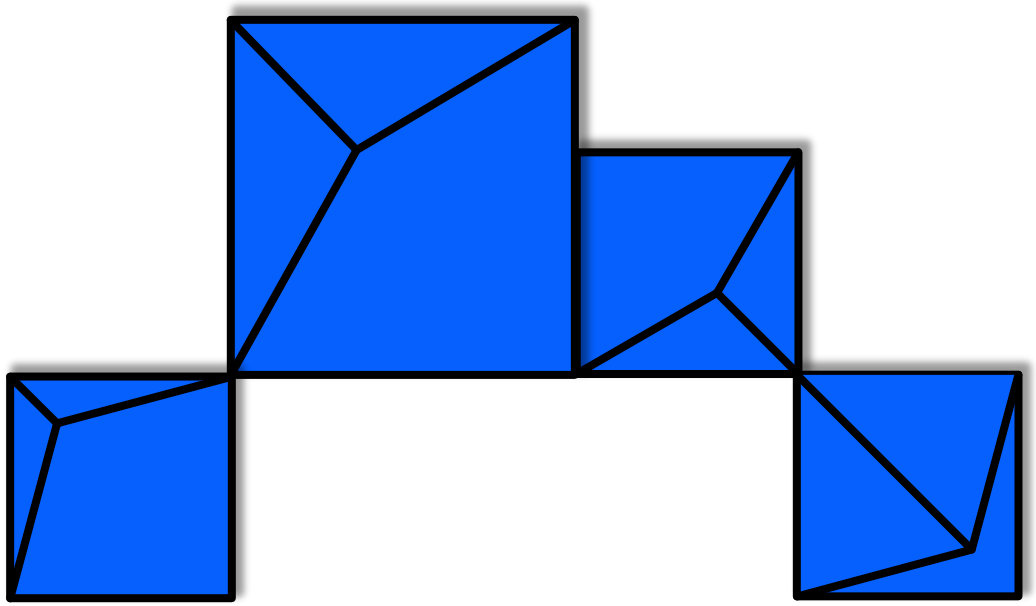
- Background
- Exclusion of non-required fields
- New document formats
- Support for C constructs
- DFDL generation changes
- Noteworthy updates
- Conclusion
- What’s next?

Background

- DFDL (Data Format Description Language) is an open standard that provides a universal, shareable, non-prescriptive description for general text and binary data formats.

- DFDL can be used to transform data between native, proprietary formats and standardized formats such as BSON, CSV, Java properties, JSON, and XML.

- DFDL is integrated in z/TPF into business events, support for MongoDB, and REST.



Exclusion of non-required fields (PJ45844 – Sept 2019)

Problem Statement

JSON and XML documents built by DFDL contain every defined field which can include many elements and array items where all fields contain a 0 value and do not need to be transmitted.

Pain Points

- Large documents created by DFDL can cause larger network bandwidth requirements.

- Updating the DFDL to exclude each optional field and array item is time consuming, error prone, and difficult to maintain.

Value Statement

DFDL can create smaller JSON and XML documents by excluding elements that contain default values.

Exclusion example

“exclude”:”none”

```
{
 “apar”:”PJ45427”,
 “reviewers”:[
    {“name”:”John Smith”,
     “date”:”2018-10-11”},
    {“name”:””,
     “date”:””}
 ],
 “coreqs”:false,
 “miginfo”:true
}
```

“exclude”:”all”

```
{
 “apar”:”PJ45427”,
 “reviewers”:[
    {“name”:”John Smith”,
     “date”:”2018-10-11”}
 ],
 “miginfo”:true
}
```

What is a non-required element?

- A simple type element with a default value.
  <xs:element name="value" type="xs:unsignedShort" dfdl:lengthKind="explicit" dfdl:length="2" dfdl:lengthUnits="bytes" default="0" />
  <xs:element name="name" type="xs:string" dfdl:lengthKind="explicit" dfdl:length="32" dfdl:lengthUnits="bytes" nillable="true" dfdl:useNilForDefault="yes" dfdl:nilKind="literalCharacter" dfdl:nilValue="%NUL;" />

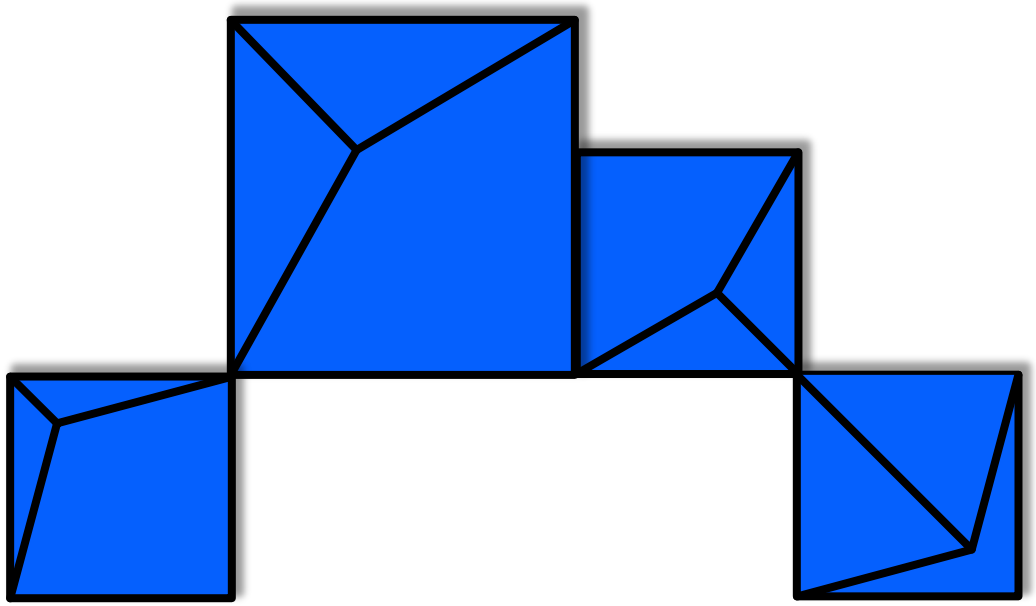
- An parameter is not required in OpenAPI unless either
  "required": true or
  "required":["propertyName"]

Technical Details

- The tpf_dfdl_buildDoc function has options for excluding non-required elements: TPF_DFDL_XDFLT, TPF_DFDL_XNULL, and TPF_DFDL_XEMPTY.

- The z/TPF service descriptor has a property of “exclude”:”all” to allow exclusion of non-required fields for REST provider and REST consumer.

- The business events dispatch adapter has an attribute of “exclude”=“all” to allow exclusion of non-required fields for standard JSON and XML formats.



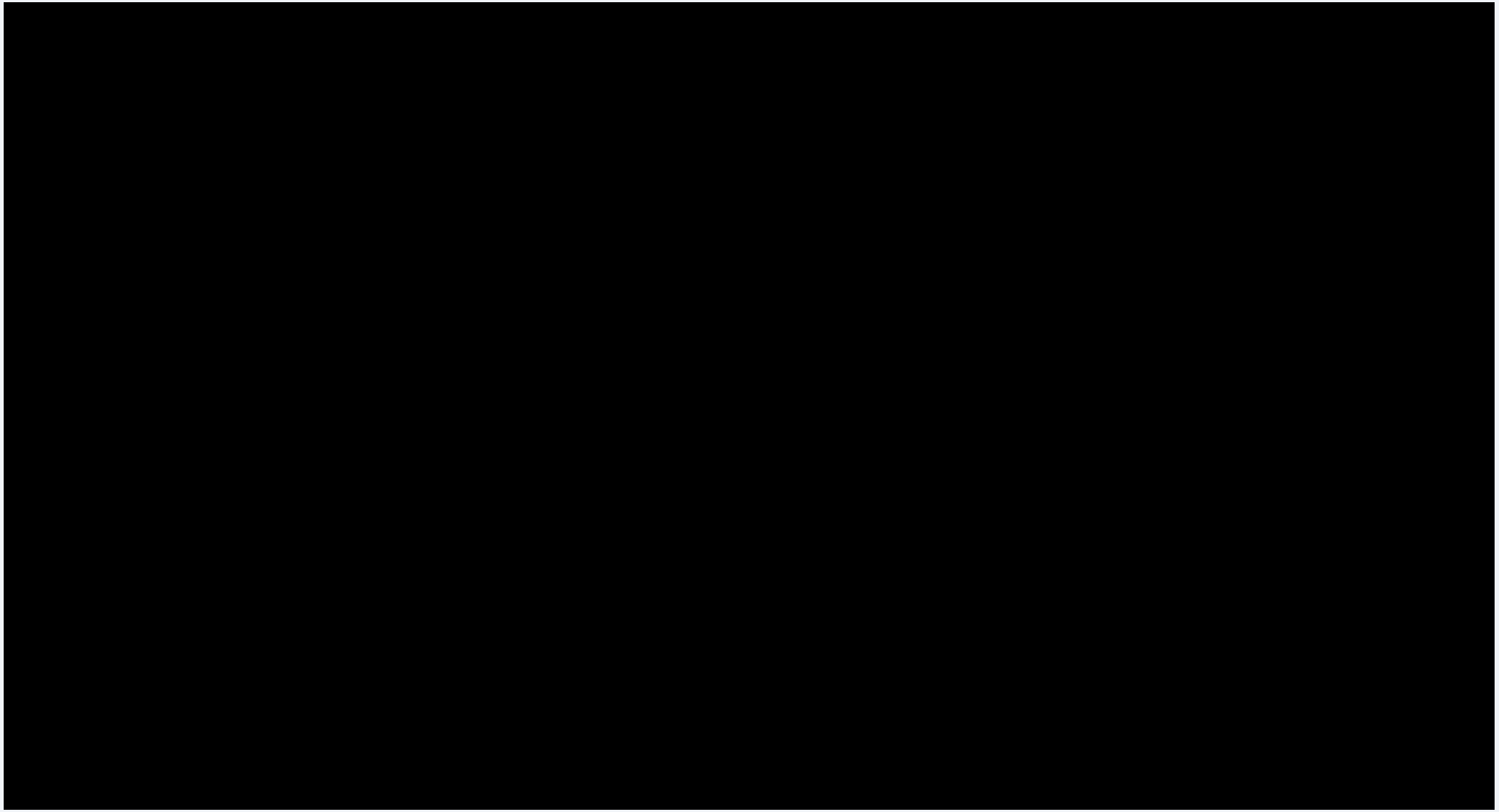
New document formats (PJ45826 – Nov 2019)

Value Statement

DFDL can enable z/TPF to send and receive BSON data to more efficiently interact with external MongoDB databases.

DFDL can make large tabular data more human readable

CSV for tabular data



Java properties for non-tabular data

```
istuseis=3
istactis=3
istwold=80
istwnew=20
istcmiuc=0
istcmacutl=4
istcmgputl=4
istcmgpcap=1000
istcmgpeng=1
istcmcapis=3
istcmflooris=2
istcmutod=00D7803CAC9139FD
istcmistt_ptr=00000001911E3700
istcmputl_wc_cur=0
istcmputl_idle_cur=0
istcmlodutl=
```

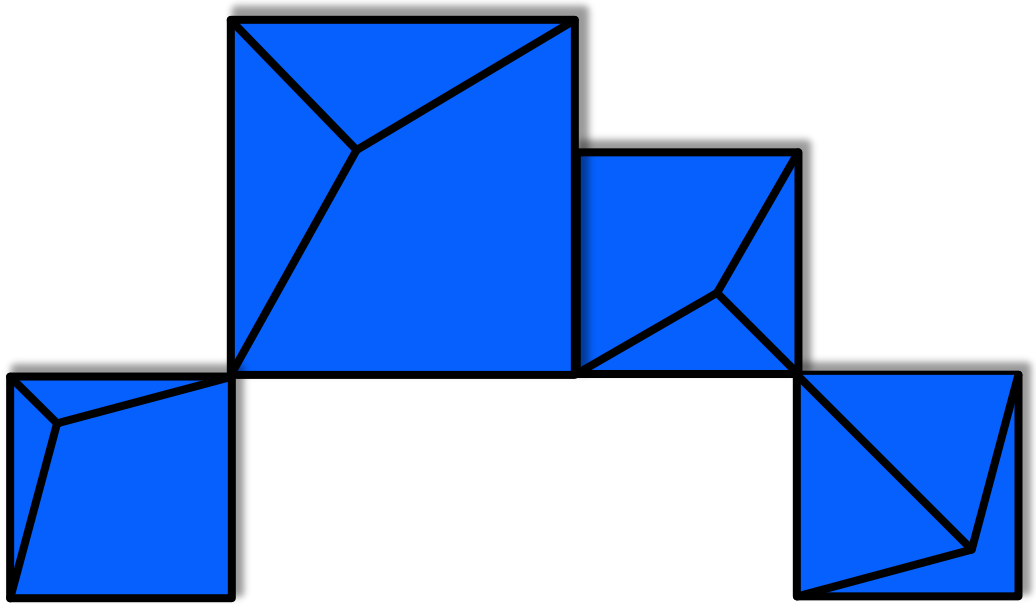
Technical Details

- The tpf_dfdl_serializeDoc function supports the following formats: BSON, Java properties, JSON, and XML.

```
data = tpf_dfdl_serializeDoc(dh, docPtr, docLen, doc_format, &datalen,
                             start_element, 0);
```

- The tpf_dfdl_buildDoc function supports the following formats: BSON, CSV, Java properties, JSON, and XML.

```
docPtr = tpf_dfdl_buildDoc(dh, &docLen, doc_format, options);
```



Support for C constructs (PJ45953 – May 2020)

Value Statement

DFDL can be used to handle common C constructs such as double pointers, pointers to variable size arrays, and pointers to null-terminated strings.

Sample DFDL definitions

1. An 8-byte pointer to a null-terminated string:

```
 char *Name;
```

```
<xs:element name="Name" type="xs:string" tddt:indirectKind="pointer"
tddt:indirectLength="8" dfdl:lengthKind="delimited"
dfdl:lengthUnits="bytes" dfdl:terminator="%NUL;"
dfdl:textTrimKind="none" default=""/>
```

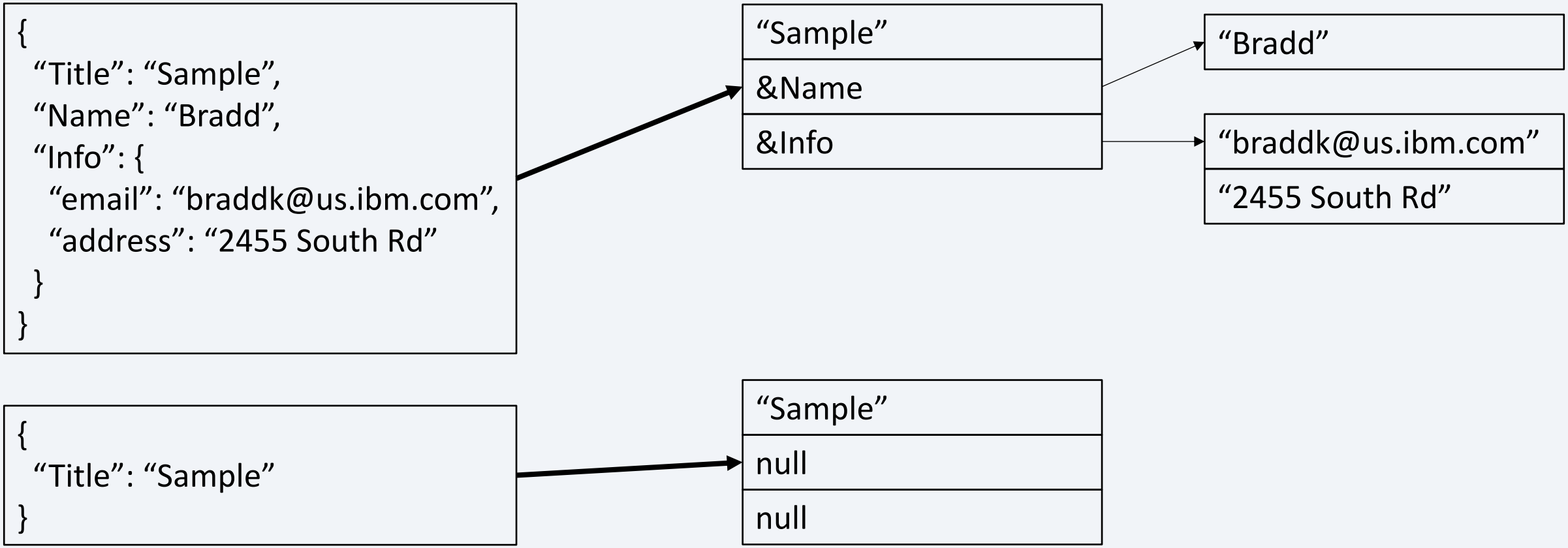
2. An 8-byte pointer to a variable size array of myStruct:

```
 myStruct *Info;
```

```
<xs:sequence tddt:indirectKind="pointer" tddt:indirectLength="8">
    <xs:element name="Info" type="myStruct"
    dfdl:lengthKind="implicit" dfdl:occursCountKind="expression"
    maxOccurs="unbounded" minOccurs="0"
    dfdl:occursCount="{../InfoItemCount}"/>
</xs:sequence>
```

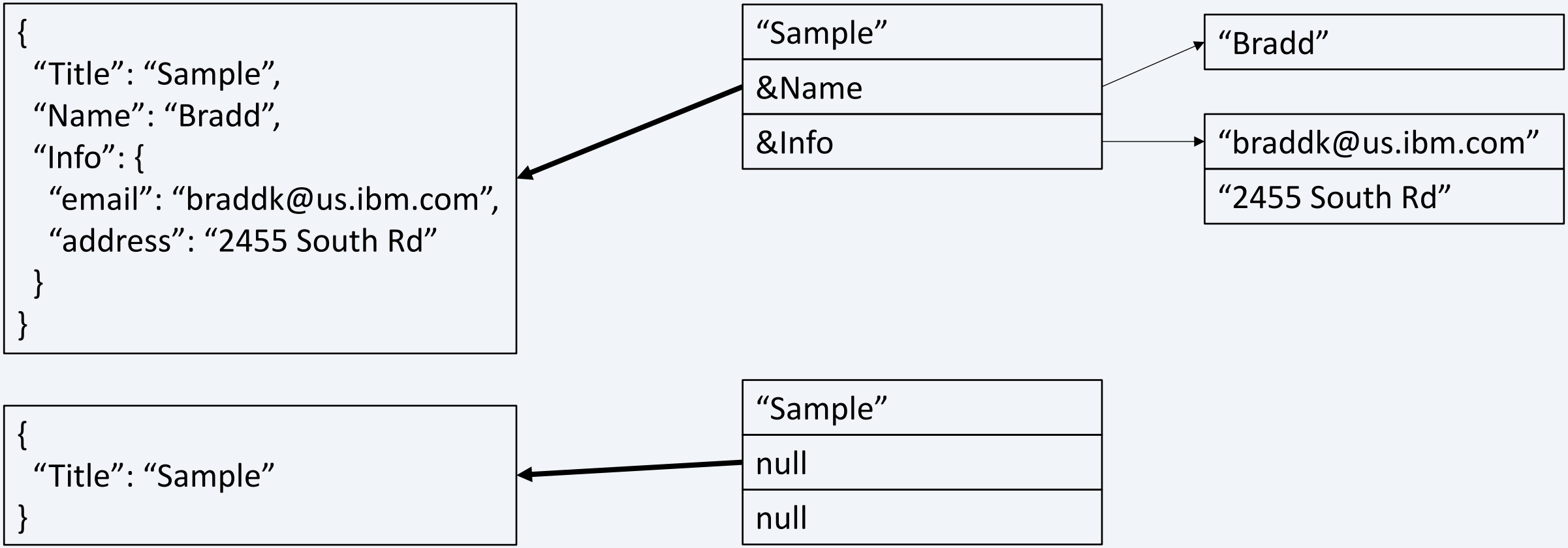
Null pointer considerations

During unparse (serialize), a null pointer is created if the underlying data is missing from the document and elements are non-required.



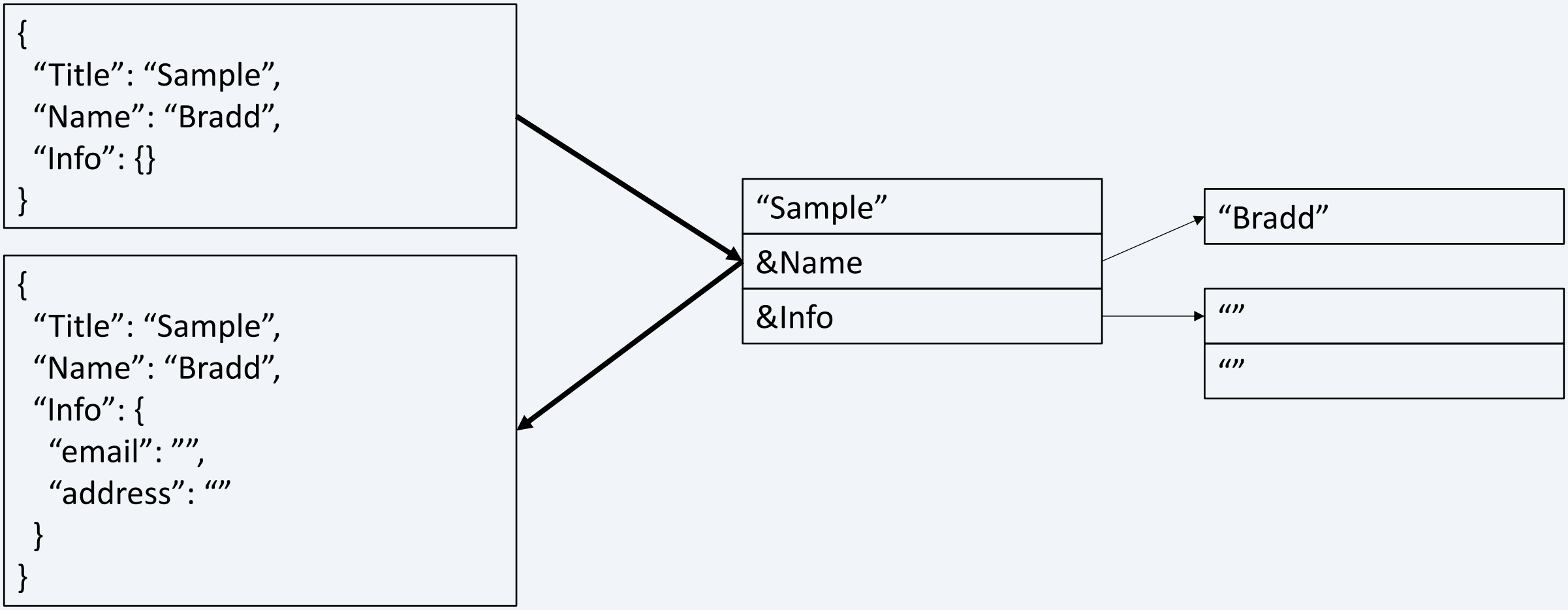
Null pointer considerations

During parse, if a null pointer is encountered in the data, the underlying data will not appear in the document.



Empty object considerations

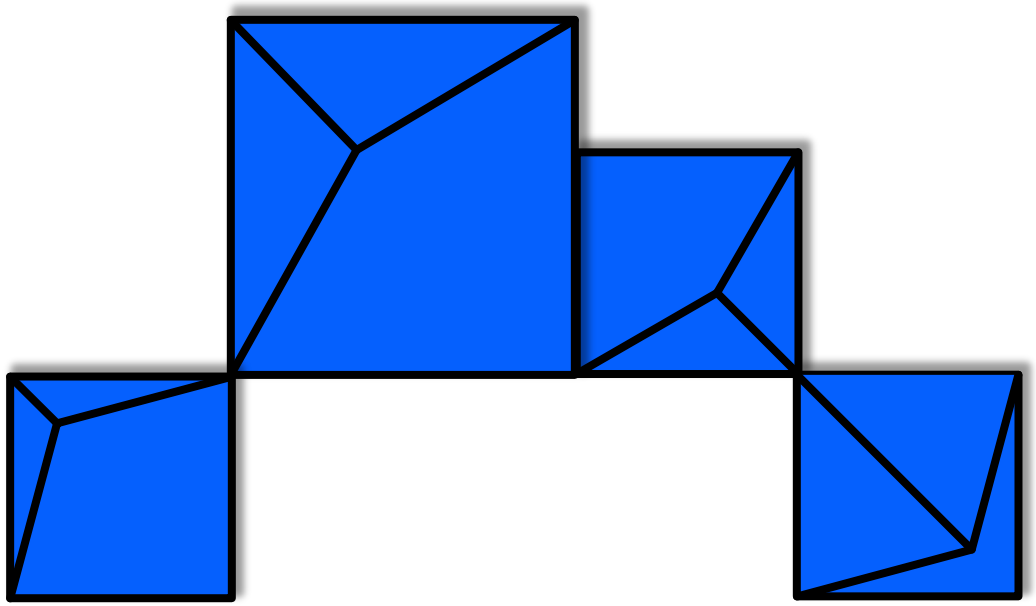
An empty object will create a pointer with default values.



Technical Details

- New DFDL annotations:
  - dfdl:lengthKind=“delimited”
  - dfdl:terminator

- Pointer annotations (indirectKind, indirectLength) allowed on the



DFDL generation changes (PJ45994 – June 2020)

Value Statement

Easily generate DFDL for non-contiguous data using C pointers.

New option for pointers

DFDL generation for:

```
 char *Name;
```

```
tpfdfdlgen
<xs:element name="Name" type="xs:hexBinary"
dfdl:lengthKind="explicit" dfdl:lengthUnits="bytes"
dfdl:length="8" default="0000000000000000"/>
```

```
tpfdfdlgen -p
<xs:element name="Name" type="xs:string"
tddt:indirectKind="pointer"  tddt:indirectLength="8"
dfdl:lengthKind="delimited" dfdl:lengthUnits="bytes"
dfdl:terminator="%NUL;" dfdl:textTrimKind="none" default=""/>
```

New option for single structure DFDL generation

DFDL generation for:

```
#include <cdfdl.h>
 struct tpf_dfdl_node_info;


tpfdfdlgen
dfdlBitInfo.gen.dfdl.xsd
dfdl_data.gen.dfdl.xsd
tpf_dfdl_node_info.gen.dfdl.xsd
...


tpfdfdlgen -n tpf_dfdl_node_info
tpf_dfdl_node_info.gen.dfdl.xsd
```

New maketpf environment variable

DFDL environment variables for maketpf.cfg

```
# Output directory for DFDL generation
TPF_DFDL_DIR := /home/braddk/PJ45994/gen

# Additional DFDL generation options
TPF_DFDL_OPTS := -n tpf_dfdl_node_info
TPF_DFDL_OPTS += -p
```

Noteworthy Updates

- PJ45579 (June 2019) added the ability to define DFDL variables.

  An external DFDL variable must be defined to use tpf_dfdl_setVariable.

```
<xs:annotation>
    <xs:appinfo source="http://www.ogf.org/dfdl/">
        <dfdl:defineVariable name="var1" type="xs:byte" external="true"/>
    </xs:appinfo>
</xs:annotation>
```

- PJ45844 (Sept 2019) added validation for minOccurs and maxOccurs when documents are created.

```
<xs:element name="itemList" dfdl:lengthKind="implicit" minOccurs="01" maxOccurs="10" dfdl:occursCountKind="expression" dfdl:occursCount="{../numItems}">
```

Conclusion

- PJ45844 (Sept 2019):
  - Provides a way to reduce network bandwidth by easily reducing the size of the JSON and XML documents that are built and transmitted.

- PJ45826 (Nov 2019):
  - Provides a way to more efficiently interact with an external MongoDB database by transmitting BSON instead of JSON.
  - Provides for easier formats to understand data for debugging and analysis.
  - Provides for easier consumtion of configuration files in Java properties format.

- PJ45953, PJ45994 (May, June 2020):
  - Provides a way to represent common C constructs for non-contiguous data.

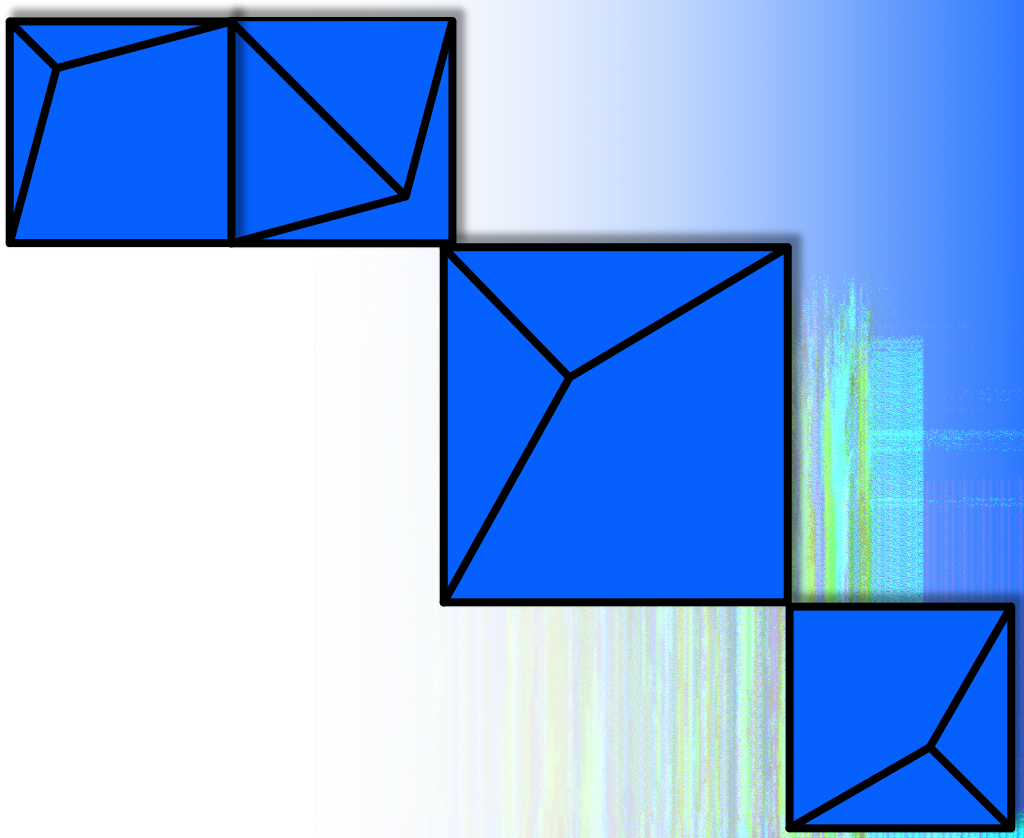
What’s next?

- Current Work:
  - PJ46213: DFDL structure to structure mapping support

- Future possibilities (interested in your feedback):
  - Support array notation for DFDL expressions
  - Allow customization for float/double document format
  - Flatten/unflatten to interchange between pointers and offsets
  - DFDL display command to help with diagnosis and validation
  - DFDL offline validation and library support (run the z/TPF DFDL parser on linux on Z).

Thank You

Questions? Comments?



Virtual TPFUG Q&A

Summary of Q&A from the virtual TPFUG event:

Question	Answer
Q: Will tpfdfdlgen handle assembler dsects?	A: The tpfdfdlgen utility is just for C/C++, but there are some utilities that can convert assembler DSECTS to C structures. The problem with these utilities or in converting DSECTs to DFDL is that the structure of the data needs to be understood, not just the offsets and lengths. The varying use of ORGs and 0 length fields in DSECTs causes problems in understanding what the data layout may be by just looking at the DSECT unless a convention is followed such as with TPFDF. If this is of interest to your company, please contact us so that we can better understand your business needs.
Q: Is there a consolidated view of these featuress or do we have to go and find each APAR to find them?	A: The IBM Knowledge Center contains a “What’s new” section for each year that lists all the enhancements for that year.

Trademarks

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.