DFDL Enhancements

Web Services Subcommittee

Abigail DeSantis

2025 TPF Users Group Conference
May 5-7, Austin, TX

IBM Z

IBM

Agenda

- PJ48127 (Jan 2025): Support for time slicing in REST processing
- PJ48141 (Mar 2025): Support to release DFDL defined structures
- Support for choice keys to improve DFDL complex schema processing

Support for time slicing in REST processing (PJ48127)

Executive summary

Yumi
Chief Technology Officer

Very large REST messages no longer cause system timeout errors due to the time needed to build or parse those messages.

Problem Statement

When DFDL builds and parses very large REST messages, a system timeout error (system error 000010) can occur.

Users

Zach
Application developer

Planning a REST service that might involve a large amount of data at times.

Pain Points

For very large REST messages, there is no way to enable time slicing to avoid a system timeout error during DFDL processing.

Value Statement

REST messages can be sent and received without worry of a system timeout error occurring.

Technical Details - PJ48127 (Jan 2025)
z/TPF service descriptor updates

tmslc – A 1- to 8-character time-slice name to use for DFDL processing of the request and response formats.

timeout – DFDL processing is part of the time needed and might be affected if time slicing is active.

```
"operationId": "oasLargeReq",
"tmslc":"IBMTRANS",
"timeout": 3000,
"request": {
    "schema": "oasRequest_t.gen.dfdl.xsd",
    "root": "oasRequest_t“
},
"response": [
    {"schema":"oasResponse_t.gen.dfdl.xsd",
     "root": "oasResponse_t"}
]
```

Technical Details - PJ48127 (Jan 2025)
ZSRVC DISPLAY update

```
ZSRVC DISPLAY n-oasLargeReq
SRVCNAME-oasLargeReq      VERSION-1.0.0
  POST /oasServices/LargeReq
  HOST-http://127.0.0.1:81
  PROXY-NONE
  TIMEOUT-3000        PROVIDERTYPE-Program  PROVIDER-QMR0 _
  UNORDERED-TRUE      DFDLFORMAT-OAS        EXCLUDE-NONE
  MAXREQUESTS-0       MAXREQUESTSERROR-0    MAXREQUESTSWARNINGINTERVAL-0
  PRIORITY-NONE       PRIORITYERROR-0       PRIORITYWARNINGINTERVAL-0
  DFDLVAL-NONE        OASVAL-NONE           TMSLC-IBMTRANS

  FILENAME                                   LOADSET         CREATED ON
  oasServices.srvc.json                      LOADTPF  01/02/25 11.57.14
  oasServices.swagger.json                      BASE  12/23/24 09.32.45
  oasResponse_t.gen.dfdl.xsd                    BASE  12/23/24 09.32.45
  oasRequest_t.gen.dfdl.xsd                     BASE  12/23/24 09.32.45
```

Does your REST service need time slicing?

Normally, an HTTP body of less than 10 MB is fine, but there can be exceptions depending on the DFDL definition and JSON values.

HTTP send operation (DFDL parse):

```
12ms, 800K JSON, 25944 properties
45ms, 2.4M JSON, 116244 properties
117ms, 8M JSON, 259224 properties
```

HTTP recv operation (DFDL serialize):

```
14ms, 800K JSON, 25944 properties
53ms, 2.4M JSON, 116244 properties
138ms, 8M JSON, 259224 properties
```

Note: results on 700 series z16

Impacts of time slicing to REST service

Giving up control by time slicing during DFDL processing (both parse and serialize) might impact the REST service time. The REST service timeout might need to be increased.

Effects of time-slice properties:

- MAXTIME – A REST service timeout will not stop the DFDL processing so have this set.
- MINSUSP – For transactional work, set the value to 0.
- RUNTIME – Have a larger value set for transactional work than for non-transactional work (more work per slice).

Support to release DFDL defined structures (PJ48141)

Executive summary

Yumi
Chief Technology Officer

Easily release memory for complex DFDL structures with a single function call, reducing the risk of memory leaks and improving code maintainability.

Problem Statement

Complex DFDL structures can cause memory leaks if pointers are not properly released, but the code customers would need to write to handle the memory leaks is difficult to maintain.

Users

Zach
Application developer

To send this request, I need to generate the DFDL structures from the OpenAPI descriptor as part of the request structure. Then I must free all embedded pointers for the DFDL structures.

Pain Points

There is no easy way to make sure all pointers associated with a DFDL defined structure are released. Customers would need to write code specific to each DFDL schema and modify the code any time the schema changes, which is hard to maintain.

Value Statement

DFDL defined structures can easily be released with one call to the tpf_dfdl_free API, improving code maintainability and reducing memory leak risk.

Technical Details - PJ48141 (Mar 2025)
tpf_dfdl_free()

To release the memory for all embedded pointers in a DFDL defined structure, simply call the tpf_dfdl_free function with the DFDL handle for that structure.

```
#include <tpf/cdfdl.h>
#include <exception>
...
#define DFDL_FILE "mydata.gen.dfdl.xsd"
#define DFDL_ROOT "mydata"
...
struct mydata buf;
...
try {
   tpf_dfdl_initialize_handle(&dh, DFDL_FILE, DFDL_ROOT, 0);
   buf = (struct mydata *)
          tpf_dfdl_serializeDoc(dh, JSONdoc, docLen,
                                TPF_DFDL_JSON, NULL, NULL,
0);
}
catch (std::exception &e) {
   // error message in e.what()
}
...

if (dh) {
    tpf_dfdl_free(dh);
    tpf_dfdl_terminate_handle(dh);
}
```

Technical Details - PJ48141 (Mar 2025) REST structure-to-structure mapping update

With this update, embedded pointers associated with internal structures are released during processing, resolving a potential memory leak.

No application changes needed.

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.

Support for choice keys to improve DFDL complex schema processing *(future plan)*

Executive summary

Yumi
Chief Technology Officer

Reduces the CPU consumption used by DFDL to parse complex structures with a large number of choice branches in the DFDL schema.

Problem Statement

Parsing very large messages that have many LRECS with a complex DFDL schema is very CPU intensive. This processing consumes large amounts of CPU time and increases latency.

Users

Zach
Application developer

Zach is parsing a z/TPFDF subfile into a JSON document and the subfile contains hundreds of extended LRECs with many kinds of subLRECs.

Pain Points

DFDL parsing uses discriminators to differentiate among each extended LREC format and each subLREC format where each branch is evaluated in order until an expression evaluates to true, which is very CPU intensive.

Value Statement

Enhances the DFDL parse process to be more efficient by using choiceDispatchKey and choiceBranchKey DFDL annotations.

- The choiceDispatchKey would cause this evaluation to happen only once for all choice branches.
- The choiceBranchKey would provide a more efficient comparison match for each branch.

Conclusion

PJ48127 (Jan 2025): Support for time slicing in REST processing

- REST messages can be sent and received without worry of a system timeout occurring.

PJ48141 (Mar 2025): Support to release DFDL defined structures

- DFDL defined structures can easily be released, reducing the risk of memory leaks and improving maintainability.

(*Future*): Support for choice keys to improve DFDL complexity schema processing

- Use choice keys to reduce CPU consumption for DFDL parsing.

Thank you