REST Enhancements

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

—

Bradd Kadlecik

Disclaimer

Agenda

- ➢ Generating packed structures from OpenAPI
- ➢ Throttling REST requests
- ➢ Displaying REST services and artifacts
- ➢ Conclusion
- ➢ What’s next

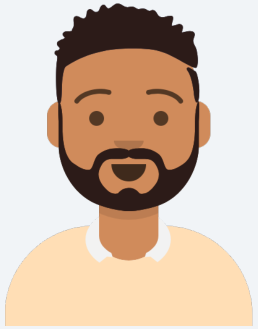
Agenda

- **Generating packed structures from OpenAPI**
- Throttling REST requests
- Displaying REST services and artifacts
- Conclusion
- What’s next

Problem Statement

REST generated structures are created unpacked in the C header files and DFDL schemas.

Users



We'll be creating a new REST service on z/TPF based on an existing OpenAPI description.

Zach
application developer

Okay.

Andrew
new hire application developer

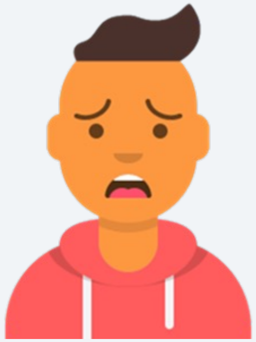
As-Is User Story

While researching REST support on z/TPF, Andrew discovers he can generate REST artifacts for the OpenAPI document by using the tpfrestgen utility.

Great! Hopefully this makes it a lot easier to get started.

As-Is User Story

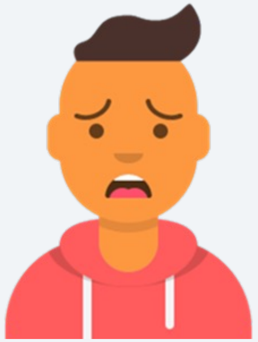
The generated C headers and artifacts helps him get a prototype developed, but he later learns according to company policy that all the C structures need to be packed. After packing the C structures, however, the REST requests start getting errors.



Uh oh! I must have messed it all up somehow.

As-Is User Story

With some help from Zach, they learn the DFDL no longer matches the C structure definitions, and they need to regenerate the DFDL for the packed structures. They'll have to remember to do this each time they regenerate the REST artifacts for changes.



What a hassle.

Pain Points

When generating C header files that must be packed:

1. Pragma pack must be added to the code manually
2. The DFDL schema must be regenerated with the z/TPF DFDL generation utility

Value Statement

An application developer can easily create the REST generated artifacts with whatever data alignment is desired.

To-Be User Story

After learning the C structures need to be packed, Andrew discovers he can use the pack option with tpfrestgen to easily update the alignment of the structures for the REST service.

Nice and simple.

Technical Details – PJ46782 (June 2022)
tpfrestgen pack option (z/TPF OpenAPI code generation utility)

- New option (-p <size>) allows the user to specify the desired alignment for the generated C structures and DFDL schema files.
- #pragma pack(size) will be added before the first structure and #pragma pack() will be added after the last structure to reset the alignment
- If –p is specified without a size, then an alignment of 1 (packed) is used.

```
tpfrestgen <swagger_file> [-h] [-mi <count>] [-ml <size>] [-o <directory>] [-p <size>]
```

Agenda

- ➢ Generating packed structures from OpenAPI
- ➢ **Throttling REST requests**
- ➢ Displaying REST services and artifacts
- ➢ Conclusion
- ➢ What’s next

Problem Statement

The z/TPF system can get overwhelmed because there aren’t sufficient controls to manage the number of REST requests.

Pain Points

There is no way to set REST request limits on a per API basis. z/TPF must rely on proper settings in API management which might not prevent sudden bursts. There can also be some requests that don’t go through API management. This can lead to the system getting overwhelmed.

Value Statement

REST throttling provides greater system stability by being able to set limits on certain REST workloads.

Technical Details – PJ46818 (Oct 2022)
REST throttling support

- REST throttling provides the ability to **limit the number of REST requests** on a per API basis to allow for greater system stability.
- A REST API can be limited by either:
  - Setting the **maximum number of concurrent requests** allowed
  - Setting a **system resource usage threshold** (lodic resource priority class)
- The action taken when the limit is hit is user configurable such as:
  - Specifying a HTTP status code to return
  - Having the request time out
- One can specify the frequency by which operations should be notified by a warning message when action is taken to limit a request.

Technical Details – PJ46818 (Oct 2022)
HTTP server message flow

1. A REST request message is sent by a client and received by the z/TPF HTTP server. By using the request URL, the z/TPF HTTP server identifies **the OpenAPI descriptor** for the request.
2. The z/TPF HTTP server locates the OpenAPI descriptor by using common deployment. Based on the URL and the HTTP method in the request message, the z/TPF HTTP server determines **the operation ID (service name)** of the service that is used to handle the request.
3. The z/TPF HTTP server locates **the z/TPF service descriptor** by using common deployment for the corresponding operation ID.
4. **Check for REST throttling**
5. Read the HTTP request headers.
6. Read the HTTP request body.
7. Call the HTTP server request/reply for logging user exit.
8. Set ISrvcName and ISrvcVersion name-value pairs.

Technical Details – PJ46818 (Oct 2022)
z/TPF service descriptor updates

- **maxRequests** – the maximum number of concurrent requests.
- **maxRequestsError** – specify the status code to return when a request exceeds the max allowed (default is 503 – Service Unavailable).
- **maxRequestsWarningInterval** – specify the number of minutes before a warning message can be issued for max requests exceeded (default is 0 meaning no warning message).
- **priorityClass** – specify the resource priority class to use for determining if a request should be rejected (eg BATCH, USRLOD1, etc).
- **priorityError** – specify the status code to return when a request is rejected due to system resource usage.
- **priorityWarningInterval** – specify the number of minutes before a warning message can be issued due to requests rejected by system resource usage.

Technical Details – PJ46818 (Oct 2022)
ZSRVC Command

ZSRVC Alter MAXREQ-num Name-service_name (Version-version)

- Allows the maximum request value to be changed in memory.
- New value will be lost when the system is IPLed. To make a permanent change, update the service descriptor.

ZSRVC Stats (RESET) Name-service_name (Version-version)

- Displays usage and error statistics for the specified service
- The error counts and highwater value can be reset with the RESET option.

Technical Details – PJ46818 (Oct 2022)
ZSRVC STats example

```
User:   ZSRVC STATS NAME-itpftest*

System: SRVC0004I 13.17.33 REST SERVICES STATISTICS DISPLAY
NAME                VERSION   ACT     HW    MAX  REJECT TIMEOUT IFERROR SCERROR
itpftest2             1.0.0    44     50     50     164      17       0       0
itpftest3             1.0.0    23     23    100       0       1       0       5
itpftest1             1.0.0     0     10      0       0       2       0       0
itpftest              1.0.0     0      0     50       0       0       0       0
END OF DISPLAY+
```

ACT – Active requests
HW – highwater count (sort key)
MAX – max concurrent allowed

REJECT – number of requests rejected by throttling
TIMEOUT – number of requests timed out
IFERROR – number of interface errors encountered
SCERROR – number of service errors (status returned >= 300)

Agenda

- ➢ Generating packed structures from OpenAPI
- ➢ Throttling REST requests
- ➢ **Displaying REST services and artifacts**
- ➢ Conclusion
- ➢ What’s next

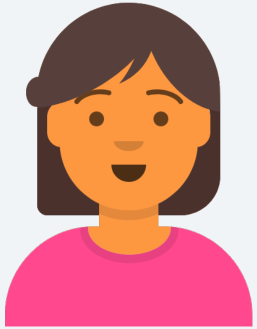
Problem Statement

It is difficult to determine what update might have introduced the problem for a REST service when an error occurs.

Users

A problem has occurred with a REST service. Can you fix it?

Derrick
operator



Yes, I’ll look into that now.

Sophie
system programmer

As-Is User Story

Sophie realizes it is going to be harder than she thought to find the bug. There are so many newly activated loadsets, but which contain files relevant to this service?

Finding this bug is going to take a while.

Pain Points

The only way to find all affected loadsets is by:

1. Looking through the service’s OpenAPI file and all the DFDL files to find all related files and their imported files.
2. Going through the contents of each loadset and comparing them to the related files to find the newest version of each file.

Value Statement

The REST artifacts display provides the ability to view when a service’s files are updated, and which loadsets those updates were in. Now if a problem occurs, it will be easy to pinpoint where the problem occurred.

To-Be User Story

Now all Sophie needs to do is use the ZSRVC command with the Display parameter to see all the affected loadsets and the times they were activated.

Wow, that’s so much easier!

Technical Details – PJ46941 (Feb 2023)
REST artifact display

```
ZSRVC Display Name-service_name Version-version
              File-filename
              URI-uri
```

1. You can display active services by service name, URI, or descriptor file name (wildcards supported at begining or end).
2. The display contains properties of the service and the file name, loadset, and activation time of all related files.
3. Loadset and activation times are of the most recent version of the file.
4. If a loadset containing a descriptor file is deactivated, the display will show the previous version of the file.

Example
REST artifact display – by service name

```
User:    ZSRVC DISPLAY NAME-oasComplexReq1 VERSION-2.0
System:  SRVC0006I 13.19.12 REST SERVICES ARTIFACTS DISPLAY
SRVCNAME-oasComplexReq1 VERSION-2.0
  PUT /oasServices/v2/complex/v1
  HOST-http://127.0.0.1:81
  TIMEOUT-5000      PROVIDERTYPE-Program   PROVIDER-QMR0
  UNORDERED-TRUE    DFDLFORMAT-OAS         EXCLUDE-ALL
  MAXREQUESTS-15    MAXREQUESTSERROR-503   MAXREQUESTSWARNINGINTERVAL-10
  PRIORITY-NONE     PRIORITYERROR-503      PRIORITYWARNINGINTERVAL-10
  FILENAME                                  LOADSET  CREATED ON
  oasServices2.srvc.json                   LOADTPF1  11/03/22 13.48.15
  oasServices2.openapi.json                LOADTPF1  11/03/22 13.48.15
  oasComplexReq1Request_t.gen.dfdl.xsd         BASE  11/01/22 08.00.00
  oasComplexReq1_200Response_t.gen.dfdl...     BASE  11/01/22 08.00.00
  tpfbase.lib.dfdl.xsd                         BASE  11/01/22 08.00.00
```

Example
REST artifact display – by URI

```
User:    ZSRVC DISPLAY URI-/flightrules/priceFlight
System:  SRVC0006I 13.19.12 REST SERVICES ARTIFACTS DISPLAY
SRVCNAME-priceFlight      VERSION-1.0.0
  POST /flightrules/priceFlight
  HOST-http://localhost:81
  TIMEOUT-5000       PROVIDERTYPE-JAM      PROVIDER-flightrules _
  UNORDERED-FALSE    DFDLFORMAT-NONE       EXCLUDE-NONE
  MAXREQUESTS-0      MAXREQUESTSERROR-0    MAXREQUESTSWARNINGINTERVAL-0
  PRIORITY-NONE      PRIORITYERROR-0       PRIORITYWARNINGINTERVAL-0
  FILENAME                                  LOADSET         CREATED ON
  flightrules.jam.xml                          BASE  02/07/23 09.33.03
  flightrules.srvc.json                        BASE  02/07/23 09.33.03
  flightrules.swagger.json                     BASE  02/07/23 09.33.03
  ticket_request.gen.dfdl.xsd                  BASE  02/07/23 09.33.03
  ticket_response.gen.dfdl.xsd                 BASE  02/07/23 09.33.03
  tpfbase.lib.dfdl.xsd                         BASE  02/07/23 09.28.44
```

Example
REST artifact display – by descriptor filename

```
User:    ZSRVC DISPLAY FILE-loophttp.ept.xml
System:  SRVC0006I 13.19.12 REST SERVICES ARTIFACTS DISPLAY
SRVCNAME-tpf1SrvcBDDef    VERSION-1.1.0
  POST /tpf1Services/serviceBDDef
  HOST-http://host.example.com
  TIMEOUT-10000      PROVIDERTYPE-Program  PROVIDER-QHH9 _
  UNORDERED-FALSE    DFDLFORMAT-NONE       EXCLUDE-NONE
  MAXREQUESTS-0      MAXREQUESTSERROR-503  MAXREQUESTSWARNINGINTERVAL-0
  PRIORITY-NONE      PRIORITYERROR-503     PRIORITYWARNINGINTERVAL-0
  FILENAME                                  LOADSET         CREATED ON
  loophttp.ept.xml                             BASE  02/07/23 09.33.03
  srvcReq.drvr.dfdl.xsd                        BASE  02/07/23 09.33.03
  srvcResp.gen.dfdl.xsd                        BASE  02/07/23 09.33.03
  tpf1Services.srvc.json                       BASE  02/07/23 09.33.03
  tpf1Services.swagger.json                    BASE  02/07/23 09.33.03
  tpfbase.lib.dfdl.xsd                         BASE  02/07/23 09.28.44
```

Example
REST artifact display with undeployed descriptor

```
User:    ZSRVC DISPLAY NAME-oasComplex*
System:  SRVC0006I 18.37.40 REST SERVICE PROPERTIES AND ARTIFACTS DISPLAY
THE FOLLOWING SERVICES HAVE AN UNDEPLOYED OPENAPI DESCRIPTOR
  oasComplexReq1          oasComplexReq2          oasComplexReq3
  /oasServices/v2/oasComplexReq1 _
  /oasServices/v2/oasComplexReq2
  /oasServices/v2/oasComplexReq3
END OF DISPLAY+
```

Conclusion

PJ46782 (June 2022): tpfrestgen pack option

- An application developer can easily create the REST generated artifacts with whatever data alignment is desired.

PJ46818 (Oct 2022): REST throttling

- Provides greater system stability by being able to set limits on certain REST workloads.

PJ46941 (Feb 2023): REST artifact display

- Easily see which artifacts have changed when a problem occurs with a REST service.

What’s next
Future possibilities

- Message handlers for REST provider and consumer
- REST support for DFDL structure to structure mapping
- Support OpenAPI 3.0

Thank you