



z/OS Connect Enterprise Edition V3.0

Improve Performance by reducing your JSON Payload Size

Version 1.0

September 2020

Alan Hollingshead
alan_hollingshead@uk.ibm.com

z/OS Connect EE
IBM UK Laboratories
Hursley Park
Winchester
Hampshire
SO21 2JN

Licensed Materials - Property of IBM

1.1	Notices	3
1.2	Trademarks and service marks	4
1.3	Terminology	4
2.	Overview.....	6
3.	Test Environment	7
3.1	Workload Driver	7
3.2	Flow of work.....	7
3.3	HTTP GET Requests.....	8
3.4	Think time.....	8
3.5	Performance measurement procedure	8
3.6	Performance metrics.....	8
3.7	Hardware.....	9
3.8	Software	9
4.	COBOL COPYBOOK	10
5.	Service Definitions	12
5.1	Excluding fields.....	12
5.2	Interface Rename	13
5.3	ODO arrays.....	14
6.	API Definitions.....	15
7.	Calling the APIs.....	16
8.	z/OS Connect EE Environment	17
8.1	JCL Configuration for z/OS Connect EE	17
8.2	JVM Configuration	17
8.3	The server.xml Configuration File	18
9.	Scenarios.....	19
10.	Results.....	21
10.1	Transactions Per Second	21
11.	CPU cost per transaction	22
11.1	CPU cost per transaction for each scenario	22
11.2	Varying JSON payload sizes for the ODO car scenario.....	23
12.	Average Response time	25
12.1	Average Response time for each scenario.....	25
13.	zIIP Eligibility.....	27
14.	Scalability.....	28
14.1	CPU cost per transaction for increasing number of clients	28
14.2	Average Response time for increasing number of clients.....	28
14.3	CPU % usage for increasing number of clients	29

15. Conclusions	31
Figure 1 - End-to-end flow for API provider scenario	7
Figure 2 - Example of RMF Workload Activity extract.....	9
Figure 3 - COBOL copybook.....	11
Figure 4 - Excluding fields from the JSON response payload	13
Figure 5 - Using the Interface Rename feature	14
Figure 6 - Definition for car100 API.....	16
Figure 7 - Mapping query parameters to copybook fields.....	16
Figure 8 - The server.xml configuration	18
Figure 9 - CPU cost per transaction (ms) for the different scenarios.	22
Figure 10 - CPU cost per transaction with increasing JSON payload size for the Car ODO scenario	24
Figure 11 - Average Response time (ms) for the different scenarios.....	25
Figure 12 - GCP CPU % and zIIP eligibility for each scenario	27
Figure 13 - CPU cost per transaction for the scenario remaining constant.....	28
Figure 14 - Average Response for 100 to 500 clients running the Interface Rename scenario.	29
Figure 15 - CPU% usage for increasing number of clients and transaction rates.....	30
Table 1 - Summary of scenarios, copybook and JSON response payload sizes.....	19
Table 2 - Difference in CPU cost per transaction even though the Car array has been excluded in the service interface.....	23

1.1 Notices

This report is intended for Architects, Systems Programmers, Analysts and Programmers wanting to understand the performance characteristics of z/OS Connect EE V3.0. The information is not intended as the specification of any programming interfaces that are provided by z/OS Connect EE.

It is assumed that the reader is familiar with the concepts and operation of z/OS Connect EE V3.0.

References in this report to IBM products or programs do not imply that IBM intends to make these available in all countries in which IBM operates.

Information contained in this report has not been submitted to any formal IBM test and is distributed "asis". The use of this information and the implementation of any of the techniques is the responsibility of the customer. Much depends on the ability of the customer to evaluate this data and project the results to their operational environment.

Disclaimer: All performance data this is contained in this report was obtained in the specified operating environment and configurations, and must be considered as an example. Performance characteristics of other operating environments might differ.

IBM does not represent, warrant, or guarantee that the same or similar results will be achieved in a user's environment as the results that are shown in this report.

1.2 Trademarks and service marks

© International Business Machines Corporation, 2020.

CICS, IMS, Db2, MQ, IBM, the IBM logo, zSystems, System z13 and z/OS are trademarks or registered trademarks of International Business Machine Corporation in the United States, other countries or both. Other company, product and service names may be trademarks or service marks of others. All rights reserved.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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.

All statements regarding IBM plans, directions, and intent are subject to change or withdrawal without notice.

1.3 Terminology

CICS	- CICS Transaction Server
Cost per transaction (ms)	- CPU usage per transaction, in milliseconds
CP	- Central Processor
CPU %	- Percentage of CPU time used by transactions running on general purpose processors
Db2	- IBM Relational Database Management System
EE	- Enterprise Edition
GCP	- General purpose Central Processor
IMS	- Information Management System
JIT optimization	- Just-In-Time optimization. The JIT compiles the JVM bytecode to machine code at runtime and uses optimization techniques such as moving calculations and caching them, moving methods to be inline, removing unnecessary locks,

- thus allowing the code to become smaller and faster.
- MQ - IBM Message Queuing
 - PID - Product Identification Number
 - RMF - Resource Measurement Facility
 - SMF - System Management Facility
 - SoR - System of Record; for example, CICS, IMS.
 - TPS - Number of Transactions Per Second
 - Think time - In seconds. For each client this is the time the workload driver waits after a response has been received before sending the next request.
 - Workload Driver - An application written for the purpose of these performance tests to simulate multiple simultaneous client requests
 - zIIP - IBM z Systems Integrated Information Processor

2. Overview

This report contains performance measurements for z/OS® Connect EE V3.0, program number (PID) 5655-CEE, demonstrating how the JSON payload size and transformation affects performance for API provider scenarios.

z/OS Connect EE includes a service package artifact that contains the mappings for the JSON payload to and from the System of Record (SoR) language structure.

When data passes through a z/OS Connect EE server, the server needs to process each byte in the payload between JSON and the SoR language structure. Therefore, the larger the JSON payload, the more CPU processing time is needed. Data transformation applies to both the request body and the response body (where supplied).

Although this data transformation is zIIP eligible, reducing the size of the JSON payload will avoid unnecessary CPU usage.

Using example data, the report demonstrates how you can improve performance through reducing the size of the JSON payload by:

- Excluding unused fields
- Renaming interface fields
- Controlling array sizes

For the purposes of these performance tests, the CICS service provider was used as the SoR, but any SoR that uses data transformation, such as IMS or MQ, could equally have been used.

Useful pre-requisite reading

Read the [Performance considerations](#) topic within the **z/OS Connect EE V3.0 Knowledge Center** to learn about performance best practices and how to implement these to improve overall performance within your environment.

3. Test Environment

This report focuses on an API provider scenario with APIs generated using the API toolkit. The API toolkit enables:

- Importing of COBOL copybook (or PL/I include file)
- Mapping of fields
- Exclusion of unused fields
- Renaming of interface fields
- A visual display of the arrays

Figure 1 shows the flow of requests and responses for the scenario:

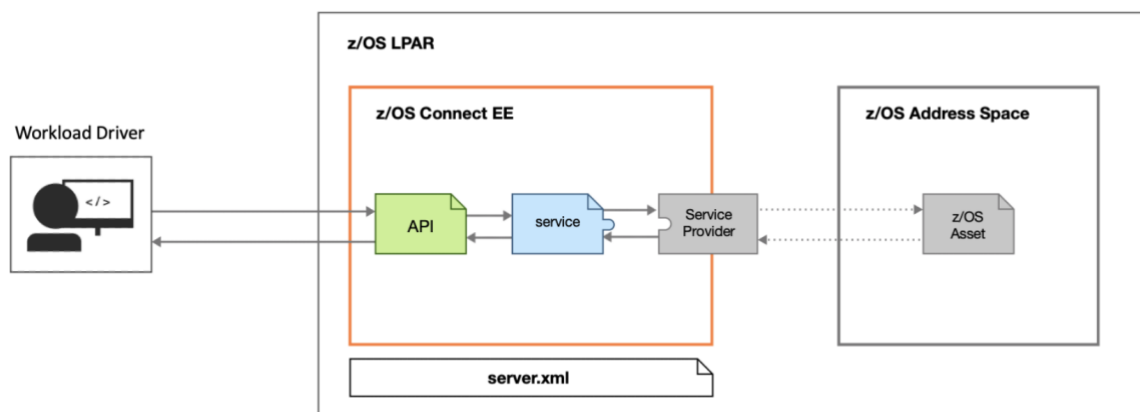


Figure 1 - End-to-end flow for API provider scenario

3.1 Workload Driver

The workload driver used to simulate multiple client requests is a Java application that runs in its own JVM on the same LPAR as z/OS Connect EE. Requests issued by the workload driver specified 127.0.0.1 (i.e. “localhost”) to minimise potential network latency.

3.2 Flow of work

The flow of work for each client was as follows:

1. The workload driver sent an HTTP JSON request to z/OS Connect EE over a persistent HTTP connection.
2. The JSON request called the API.
3. The API called the service.
4. The service mapped the JSON objects to the fields in the copybook.
5. The service invoked the service provider.
6. The service provider sent the transformed data request to the SoR (e.g. CICS) to call the z/OS asset (e.g. CICS program).
7. The z/OS asset sent a payload in the response.
8. z/OS Connect EE transformed the response into a JSON payload, excluding

fields and/or renaming fields, as configured in the service interface of the API toolkit.

9. The HTTP response containing the JSON payload was sent back to the client.

3.3 HTTP GET Requests

All scenarios used in this report issued HTTP GET requests with no request body. Although this report focuses on the data transformation of the response body, the same benefits can be gained with, for example, an HTTP PUT request with request and response bodies.

3.4 Think time

Requests were issued by the workload driver at a regular pace, using a “think time”. For each client this is the time the workload driver waits after a response has been received before sending the next request. All performance runs in this report used a think time of 500ms to simulate a typical customer environment.

3.5 Performance measurement procedure

The same procedure was carried out for each measurement and consisted of the following steps:

1. z/OS Connect EE was started with the “clean” option.
2. The scenario used CICS as the SoR and was cold started with a TCPIPService installed and CICS program available. Note that the performance profile for data transformation for other service providers, such as IMS or MQ, would be equivalent to that for CICS, although the exact values might vary slightly due to the differences in data structures between different SoRs.
3. The workload was started, and several hundred thousand requests were run before any measurements were taken to ensure the JIT was optimized.
4. IBM z/OS Resource Measurement Facility (RMF) data was collected at one-minute intervals.

3.6 Performance metrics

The following metrics were gathered during the performance runs:

- Request rate (transactions per second)
- Average response time
- CP (general processor) usage
- Potential zIIP usage

CP and potential zIIP usage values were extracted from RMF workload activity data. RMF reports CPU utilisation that is based on uniprocessor capacity, which means that a value of 100% represents the capacity of one processor.

Figure 2 shows an example of a Workload Activity extract from an RMF report for a z/OS Connect EE server. For the one-minute interval, the APPL% section shows

19.94% of general processor time was used (the “CP” value) and that 19.52% (the “IIPCP” value) is potentially offloadable to zIIP processors.

The request rate (transactions per second) and average response time values were gathered by the workload driver.

WORKLOAD ACTIVITY													
z/OS V2R3		SYSPLEX MV22		DATE 09/10/2020		INTERVAL 01.00.000		MODE = GOAL					
		RPT VERSION V2R3 RMF		TIME 13.33.06									
POLICY ACTIVATION DATE/TIME 07/22/2020 15.53.43													
POLICY=NORMAL					REPORT CLASS=ZOSCONN								
					DESCRIPTION =zosConnect reports								

---TRANSACTIONS---	TRANS-TIME	HHH.MM.SS.FFFFFF	TRANS-APPL%	CP-IIPCP/AAPCP-IIP/AAP	---ENCLAVES---								
AVG	1.00	ACTUAL	0	TOTAL	19.94	19.52	0.00	AVG ENC	0.00				
MPL	1.00	EXECUTION	0	MOBILE	0.00	0.00	0.00	REM ENC	0.00				
ENDED	0	QUEUED	0	CATEGORYA	0.00	0.00	0.00	MS ENC	0.00				
END/S	0.00	R/S AFFIN	0	CATEGORYB	0.00	0.00	0.00						
#SWAPS	0	INELIGIBLE	0										
EXCTD	0	CONVERSION	0										
		STD DEV	0										

----SERVICE----	SERVICE TIME	---APPL %---	---PROMOTED---	---DASD I/O---	---STORAGE---	-PAGE-IN RATES-							
IOC	0	CPU	11.733	CP	19.94	BLK	0.000	SSCHRT	0.0	AVG	231428.1	SINGLE	0.0
CPU	873187	SRB	0.231	IIPCP	19.52	ENQ	0.000	RESP	0.0	TOTAL	231428.1	BLOCK	0.0
MSO	0	RCT	0.000	IIP	0.00	CRM	0.000	CONN	0.0	SHARED	21.00	SHARED	0.0
SRB	17207	IIT	0.000	AAPCP	0.00	LCK	0.006	DISC	0.0			HSP	0.0
TOT	890394	HST	0.000	AAP	N/A	SUP	0.000	Q+PEND	0.0				
/SEC	14840	IIP	0.000					IOSQ	0.0				
ABSRPTN	15K	AAP	N/A										
TRX SERV	15K												

Figure 2 - Example of RMF Workload Activity extract

3.7 Hardware

- IBM System z13 2964-NE1 model 7A5
- 10GB of Central Storage (RAM)
- LPAR with 6 dedicated GCPs (no zIIPs)
- OSA-Express5S 10GB Ethernet

3.8 Software

- z/OS V2.3
- z/OS Connect Enterprise Edition (EE) V3.0.36
 - z/OS Connect EE build 20200817-1534
 - Liberty 20.0.0.6 including Angel V14
- IBM 64-bit SDK for z/OS Java Technology Edition, Version 8.0
 - Java 1.8 SR6 FP11
- CICS Transaction Server V5.5

4. COBOL COPYBOOK

The scenario for this performance report was based around a car inquiry service. The client can ask for details about cars that match particular criteria and/or details of dealers in the dealership.

The COBOL copybook used various datatypes and two OCCURS DEPENDING ON (ODO) arrays. Within structure CARMODEL-CARS was an ODO array for the different cars containing 100 elements, whilst within structure CARMODEL-DEALERS was another ODO array containing details for 25 dealers.

```

*****
*                                     CAR100CP                               *
*****

** INPUT: Following fields are INPUT fields
   03 REQ-TXN-REQUEST-BLOCK.
       05 REQ-CAR-MODEL                PIC  X(8).
       05 REQ-LIST-EXTRAS              PIC  X(36).
** PIC 9(09) COMP. take 4 bytes each
       05 REQ-MAX-NBR-OF-CARS         PIC  9(09) COMP.
       05 REQ-MAX-NBR-OF-DEALERS     PIC  9(09) COMP.

** OUTPUT: Following fields are OUTPUT fields
** 88s
   03 RESP-TXN-RESPONSE-BLOCK.
       05 RESP-AVAILABILITY-FLAG      PIC  X(01).
       88 RESP-AVAILABLE              VALUE 'A'.
       88 RESP-NONE                   VALUE 'N'.
       88 RESP-SIMILAR                VALUE 'S'.
** PIC 9(09) COMP. take 4 bytes each
       05 RESP-NBR-OF-CARS-RETURNED   PIC  9(09) COMP.
       05 RESP-NBR-OF-DEALERS-RETURNED PIC  9(09) COMP.
       05 RESP-TXN-INPUT-DATA-LEN     PIC  9(09) COMP.
       05 RESP-TXN-OUTPUT-DATA-LEN    PIC  9(09) COMP.
       05 RESP-TXN-DATA-LEN           PIC  9(09) COMP.
       05 FILLER                      PIC  X(08).

** PIC S9(04) COMP.
       05 RESP-MAJ-RET-CODE           PIC  S9(04) COMP.
       05 RESP-MNR-RET-CODE           PIC  S9(04) COMP.

** TIMESTAMP MESSAGE
       05 RESP-TIMESTAMP.
           15 THE-DATE                 PIC  X(8)  VALUE SPACES.
           15 FILLER                   PIC  X(2)  VALUE '--'.
           15 THE-TIME                 PIC  X(8)  VALUE SPACES.
           15 FILLER2                  PIC  X(25) VALUE SPACES.

**
   03 CARMODEL-CAR-FINDER.

** ODO array
       05 CARMODEL-CARS.
           10 CARMODEL-CAR-DATA OCCURS 1 TO 100 TIMES
              DEPENDING ON RESP-NBR-OF-CARS-RETURNED.
           15 CARMODEL-1-REF          PIC  X(12).

```

```

15 CARMODEL-2-MODEL          PIC X(12).
15 CARMODEL-3-COST          PIC X(12).
15 CARMODEL-4-LEATHERSEATS  PIC X(12).
15 CARMODEL-5-SUNROOF      PIC X(12).
15 CARMODEL-6-REARCAMERA   PIC X(12).
15 CARMODEL-7-SATNAV       PIC X(12).
15 CARMODEL-8-HEATEDSEATS  PIC X(12).
15 CARMODEL-9-BLUETOOTH    PIC X(12).
15 CARMODEL-10-PARKINGSENSORS PIC X(12).
15 CARMODEL-11-ALLOYS      PIC X(12).
15 CARMODEL-12-REMOTESTART PIC X(12).
15 CARMODEL-13-BLINDSPOT   PIC X(12).
15 CARMODEL-14-THIRDROWSEATING PIC X(12).
15 CARMODEL-15-HEATEDMIRRORS PIC X(12).
15 CARMODEL-16-FRONTCAMERA  PIC X(12).
15 CARMODEL-17-HEATEDWINDSCREEN PIC X(12).
15 CARMODEL-18-ALARM       PIC X(12).
15 CARMODEL-19-TRACKER     PIC X(12).
15 CARMODEL-20-POWERSEATS  PIC X(12).

** ODO array
  05 CARMODEL-DEALERS.
    10 CARMODEL-DEALER-DATA OCCURS 1 TO 25 TIMES
        DEPENDING ON RESP-NBR-OF-DEALERS-RETURNED.
      15 CARMODEL-DEALERSHIP          PIC X(16).
      15 CARMODEL-DEALER-COUNTRY     PIC X(2).
        88 LOC-A                      VALUE 'UK'.
        88 LOC-B                      VALUE 'FR'.
        88 LOC-C                      VALUE 'DE'.
      15 CARMODEL-DEALER-ADDRES      PIC X(90).
      15 CARMODEL-DEALER-PHONE-NUMBE PIC X(16).

    05 FILLER                        PIC X(2900).

```

Figure 3 - COBOL copybook

5. Service Definitions

Two services were created for the performance runs using the API toolkit v3.0.8 and modified as required for the different scenarios:

- Car100service
- Car100serviceRename.

Car100service was used to obtain performance results for when:

- No fields are excluded
- Some fields are excluded
- ODO arrays restrict the number of array elements.

Car100serviceRename was used to show the performance benefits of using the Interface Rename feature of the API toolkit.

For each service, the copybook CAR100CP (see Figure 3 - COBOL copybook) was imported into the API toolkit for both the request and the response.

5.1 Excluding fields

To exclude some fields for a performance run, selected fields had the “Include” checkbox unchecked, as shown in the service interface for the Car100service response:

Service Interface Editor

Define and customize your request and response service interfaces. Right-click a row and select the appropriate action from the context menu, or select a row and clic

Search:

Fields	Include	Interface Rename	Default...	Data Type	Field Length	Start
▼ COMMAREA						
▼ CAR100CP		CAR100CP				
> REQ_TXN_REQUEST_BLOCK	<input type="checkbox"/>	REQ_TXN_REQUEST_BLOCK		STRUCT	52	1
> RESP_TXN_RESPONSE_BLOCK	<input checked="" type="checkbox"/>	RESP_TXN_RESPONSE_BLOCK		STRUCT	76	53
▼ CARMODEL_CAR_FINDER	<input checked="" type="checkbox"/>	CARMODEL_CAR_FINDER		STRUCT	30000	129
▼ CARMODEL_CARS	<input checked="" type="checkbox"/>	CARMODEL_CARS		STRUCT	24000	129
▼ [📄] CARMODEL_CAR_DATA [1..100]	<input checked="" type="checkbox"/>	CARMODEL_CAR_DATA		ARRAY	24000	129
CARMODEL_1_REF	<input checked="" type="checkbox"/>	CARMODEL_1_REF		CHAR	12	
CARMODEL_2_MODEL	<input checked="" type="checkbox"/>	CARMODEL_2_MODEL		CHAR	12	
CARMODEL_3_COST	<input checked="" type="checkbox"/>	CARMODEL_3_COST		CHAR	12	
CARMODEL_4_LEATHERSEATS	<input checked="" type="checkbox"/>	CARMODEL_4_LEATHERSEATS		CHAR	12	
CARMODEL_5_SUNROOF	<input checked="" type="checkbox"/>	CARMODEL_5_SUNROOF		CHAR	12	
CARMODEL_6_REARCAMERA	<input checked="" type="checkbox"/>	CARMODEL_6_REARCAMERA		CHAR	12	
CARMODEL_7_SATNAV	<input checked="" type="checkbox"/>	CARMODEL_7_SATNAV		CHAR	12	
CARMODEL_8_HEATEDSEATS	<input checked="" type="checkbox"/>	CARMODEL_8_HEATEDSEATS		CHAR	12	
CARMODEL_9_BLUETOOTH	<input checked="" type="checkbox"/>	CARMODEL_9_BLUETOOTH		CHAR	12	
CARMODEL_10_PARKINGSSENSORS	<input checked="" type="checkbox"/>	CARMODEL_10_PARKINGSSENSORS		CHAR	12	
CARMODEL_11_ALLOYS	<input checked="" type="checkbox"/>	CARMODEL_11_ALLOYS		CHAR	12	
CARMODEL_12_REMOTESTART	<input checked="" type="checkbox"/>	CARMODEL_12_REMOTESTART		CHAR	12	
CARMODEL_13_BLINDSPOT	<input checked="" type="checkbox"/>	CARMODEL_13_BLINDSPOT		CHAR	12	
CARMODEL_14_THIRDROWSEATING	<input checked="" type="checkbox"/>	CARMODEL_14_THIRDROWSEATING		CHAR	12	
CARMODEL_15_HEATEDMIRRORS	<input checked="" type="checkbox"/>	CARMODEL_15_HEATEDMIRRORS		CHAR	12	
CARMODEL_16_FRONTCAMERA	<input checked="" type="checkbox"/>	CARMODEL_16_FRONTCAMERA		CHAR	12	
CARMODEL_17_HEATEDWINDSCREEN	<input checked="" type="checkbox"/>	CARMODEL_17_HEATEDWINDSCREEN		CHAR	12	
CARMODEL_18_ALARM	<input checked="" type="checkbox"/>	CARMODEL_18_ALARM		CHAR	12	
CARMODEL_19_TRACKER	<input checked="" type="checkbox"/>	CARMODEL_19_TRACKER		CHAR	12	
CARMODEL_20_POWERSEATS	<input checked="" type="checkbox"/>	CARMODEL_20_POWERSEATS		CHAR	12	
▼ CARMODEL_DEALERS	<input type="checkbox"/>	CARMODEL_DEALERS		STRUCT	3100	
▼ [📄] CARMODEL_DEALER_DATA [1..25]	<input type="checkbox"/>	CARMODEL_DEALER_DATA		ARRAY	3100	
CARMODEL_DEALERSHIP	<input type="checkbox"/>	CARMODEL_DEALERSHIP		CHAR	16	
CARMODEL_DEALER_COUNTRY	<input type="checkbox"/>	CARMODEL_DEALER_COUNTRY		CHAR	2	
CARMODEL_DEALER_ADDRESS	<input type="checkbox"/>	CARMODEL_DEALER_ADDRESS		CHAR	90	
CARMODEL_DEALER_PHONE_NUMBER	<input type="checkbox"/>	CARMODEL_DEALER_PHONE_NUMBER		CHAR	16	
FILL_2	<input type="checkbox"/>	FILL_2		CHAR	2900	

Figure 4 - Excluding fields from the JSON response payload

By excluding fields, the JSON payload size was reduced. The smaller the JSON payload size, the less CPU processing was required, thereby improving performance.

In the example above, the response did not require the request fields to be in the JSON schema, and the service creator had chosen not to include the dealers either.

5.2 Interface Rename



When a copybook is imported in the API toolkit, the field names that are shown in the **Interface Rename** column default to the same name as the original field name. Using these field names is not good practice, an object-oriented programmer fluent in JSON would expect to see hierarchical camel-case names. The Interface Rename feature of the API toolkit allows you to rename these fields to:









- Names a JSON object-oriented programmer would be familiar with.
- Shorter field names, thereby reducing the size of the JSON payload which in turn can improve performance.

In the example below the request for the Car100serviceRename service shows how fields were renamed to shorter and more object-oriented names.

Service Interface Editor

Define and customize your request and response service interfaces. Right-click a row and select the appropriate action from the context menu,

Search:  





Fields	Include	Interface Rename	Defa...	Data Type	Field Len...	Start
▼  COMMAREA						
▼  CAR100CP		CAR100CP				
> REQ_TXN_REQUEST_BLOCK	<input checked="" type="checkbox"/>	Request		STRUCT	52	1
> RESP_TXN_RESPONSE_BLOCK	<input checked="" type="checkbox"/>	Response		STRUCT	76	53
▼ CARMODEL_CAR_FINDER	<input checked="" type="checkbox"/>	CarFinder		STRUCT	30000	129
▼ CARMODEL_CARS	<input checked="" type="checkbox"/>	Cars		STRUCT	24000	129
▼  [Ctrl] CARMODEL_CAR_DATA [1..100]	<input checked="" type="checkbox"/>	Data		ARRAY	24000	129
CARMODEL_1_REF	<input checked="" type="checkbox"/>	Ref		CHAR	12	
CARMODEL_2_MODEL	<input checked="" type="checkbox"/>	Model		CHAR	12	
CARMODEL_3_COST	<input checked="" type="checkbox"/>	Cost		CHAR	12	
CARMODEL_4_LEATHERSEATS	<input checked="" type="checkbox"/>	LeatherSeats		CHAR	12	
CARMODEL_5_SUNROOF	<input checked="" type="checkbox"/>	Sunroof		CHAR	12	
CARMODEL_6_REARCAMERA	<input checked="" type="checkbox"/>	RearCamera		CHAR	12	
CARMODEL_7_SATNAV	<input checked="" type="checkbox"/>	SatNav		CHAR	12	
CARMODEL_8_HEATEDSEATS	<input checked="" type="checkbox"/>	HeatedSeats		CHAR	12	
CARMODEL_9_BLUETOOTH	<input checked="" type="checkbox"/>	Bluetooth		CHAR	12	
CARMODEL_10_PARKINGSENSORS	<input checked="" type="checkbox"/>	ParkingSensors		CHAR	12	
CARMODEL_11_ALLOYS	<input checked="" type="checkbox"/>	Alloys		CHAR	12	
CARMODEL_12_REMOTESTART	<input checked="" type="checkbox"/>	RemoteStart		CHAR	12	
CARMODEL_13_BLINDSPOT	<input checked="" type="checkbox"/>	BlindSpot		CHAR	12	
CARMODEL_14_THIRDROWSEATING	<input checked="" type="checkbox"/>	ThirdRow		CHAR	12	
CARMODEL_15_HEATEDMIRRORS	<input checked="" type="checkbox"/>	HeatedMirrors		CHAR	12	
CARMODEL_16_FRONTCAMERA	<input checked="" type="checkbox"/>	FrontCamera		CHAR	12	
CARMODEL_17_HEATEDWINDSCREEN	<input checked="" type="checkbox"/>	HeatedWindscreen		CHAR	12	
CARMODEL_18_ALARM	<input checked="" type="checkbox"/>	Alarm		CHAR	12	
CARMODEL_19_TRACKER	<input checked="" type="checkbox"/>	Tracker		CHAR	12	
CARMODEL_20_POWERSEATS	<input checked="" type="checkbox"/>	PowerSeats		CHAR	12	
▼ CARMODEL_DEALERS	<input checked="" type="checkbox"/>	Dealers		STRUCT	3100	
▼  [Ctrl] CARMODEL_DEALER_DATA [1..25]	<input checked="" type="checkbox"/>	Data		ARRAY	3100	
CARMODEL_DEALERSHIP	<input checked="" type="checkbox"/>	Dealership		CHAR	16	
CARMODEL_DEALER_COUNTRY	<input checked="" type="checkbox"/>	Country		CHAR	2	
CARMODEL_DEALER_ADDRESS	<input checked="" type="checkbox"/>	Address		CHAR	90	
CARMODEL_DEALER_PHONE_NUMBER	<input checked="" type="checkbox"/>	PhoneNumber		CHAR	16	
FILL_2	<input checked="" type="checkbox"/>	Fill3		CHAR	2900	

Figure 5 - Using the Interface Rename feature

5.3 ODO arrays

A variable-length array is defined in a COBOL copybook by an OCCURS DEPENDING ON (ODO) clause, and in a PL/I include file by a dimension attribute where the lower or upper bounds (or both) are defined by REFER options.

Minimising the number of elements in the request, reduces the size of the JSON payload. For example, rather than returning car details for 100 different cars, the client may only want the first ten cars, or ones with particular features. This considerably reduces the size of the JSON payload to be transformed, thereby saving CPU processing.

6. API Definitions

Two APIs were created for the performance runs using the API toolkit v3.0.8 and modified as required for the different scenarios:

- car100
- car100rename.

API car100 called service Car100service and were used to obtain performance results for when:

- No fields were excluded
- Some fields were excluded
- ODO arrays restricted the number of array elements.

API car100rename called service Car100serviceRename where the Interface Rename feature had renamed and shortened the field names.

7. Calling the APIs

All API calls were made from the workload driver specifying a URL in the following format:

http://<host>:<port>/<basepath>/<relative path>?query_parm1&query_parm2

For example:

http://myHost:2222/car100/models?maxCars=100&maxDealers=25

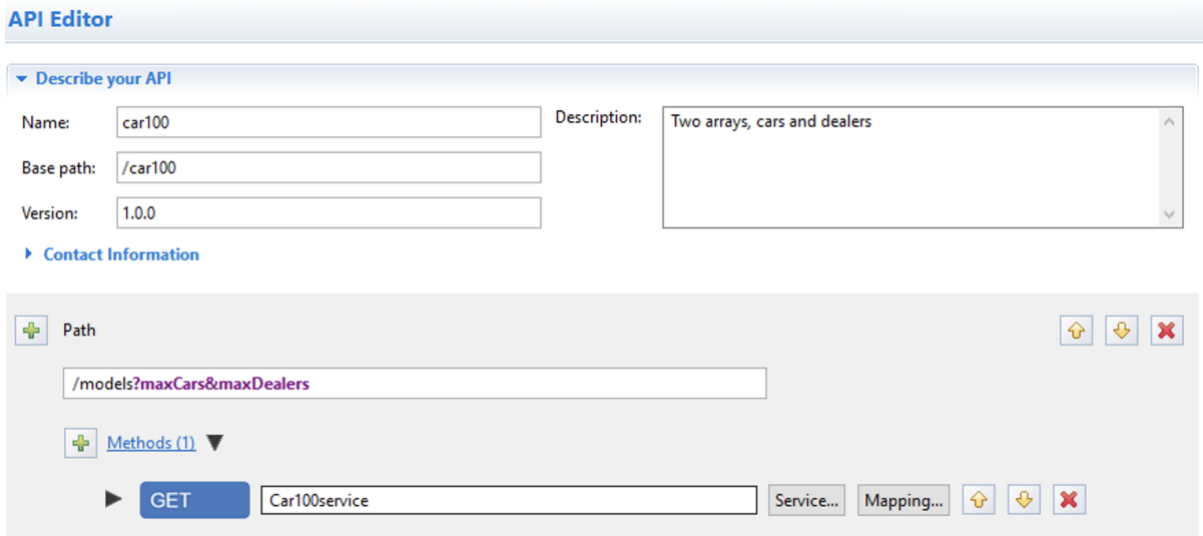


Figure 6 - Definition for car100 API

The query parameters, maxCars and maxDealers, were mapped to fields in the copybook for the request using the mapping feature within the API toolkit.

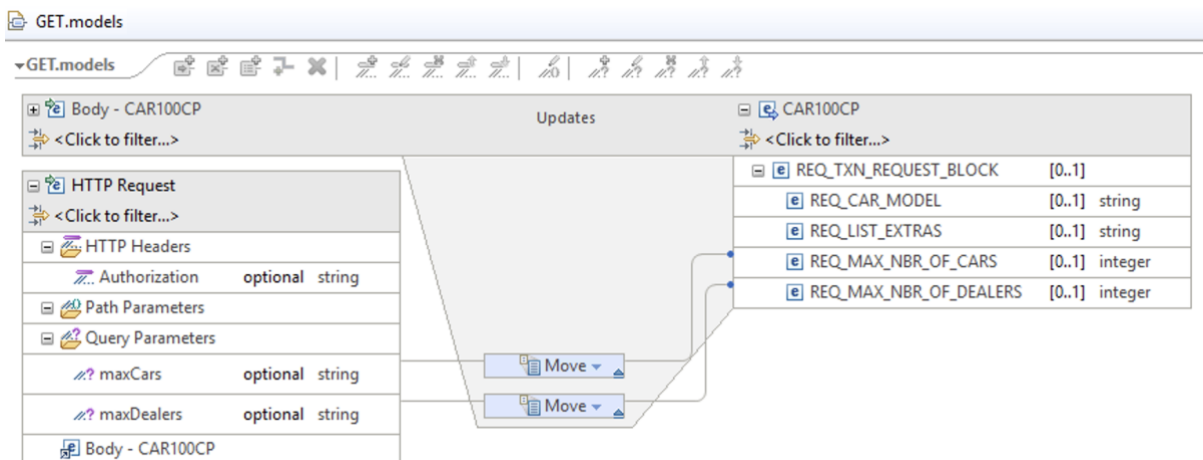


Figure 7 - Mapping query parameters to copybook fields

8. z/OS Connect EE Environment

This chapter describes the z/OS Connect EE environment the performance tests were run in.

8.1 JCL Configuration for z/OS Connect EE

The started procedure for z/OS Connect EE was configured with:

- REGION size 0M
- MEMLIMIT 4G

8.2 JVM Configuration

The JVM used by z/OS Connect EE was configured with:

- IBM 64-bit SDK for z/OS Java Technology Edition, Version 8.0
 - Java 1.8 SR6 FP11
- Maximum Java heap size (Xmx) 1G
 - Other Java heap (Garbage Collector) values set to minimise the impact of the heap variation during performance runs:
 - -Xms1G to match the maximum Java heap size
 - -Xmnx512M to set the maximum new (nursery) size
 - -Xmns512M to set the minimum new (nursery) size
 - -Xmox512M to set the maximum old (tenured) size
 - -Xmos512M to set the minimum old (tenured) size
- All other JVM system properties used default values (including garbage collection mode: gencon)

8.3 The server.xml Configuration File

The server.xml file used for the z/OS Connect EE server contained the following definitions:

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="z/OS Connect EE 3.0. Polling not enabled.">

  <config updateTrigger="mbean"></config>

  <!-- Enable features -->
  <featureManager>
    <feature>zosconnect:cicsService-1.0</feature>
    <feature>zosconnect:zosConnectCommands-1.0</feature>
  </featureManager>

  <!-- ZosConnectManager definition -->
  <zosconnect_zosConnectManager
    requireSecure="false"
    requireAuth="false"
    operationMode="ASYNC">
  </zosconnect_zosConnectManager>

  <zosconnect_cicsIpicConnection id="cicsConn"
    host="12.34.56.789" port="1155" sendSessions="500" />

  <!-- zosConnect APIs -->
  <zosconnect_zosConnectAPIs updateTrigger="mbean"/>

  <!-- zosConnect services -->
  <zosconnect_services updateTrigger="mbean"/>

  <!-- applicationMonitor is not applicable for z/OS Connect EE servers -->
  <applicationMonitor updateTrigger="disabled" dropinsEnabled="false"/>

  <!--Server port -->
  <httpEndpoint id="defaultHttpEndpoint"
    enabled="true"
    host="127.0.0.1"
    httpPort="2222" />
</server>
```

Figure 8 - The server.xml configuration

Note that polling was not enabled in the server.xml configuration file as this can have an impact on performance and is not recommended in a production environment. This includes proactively disabling the applicationMonitor which is not used by z/OS Connect EE servers, but if unspecified, polling is enabled by default by Liberty.

Refer to the [z/OS Connect EE 3.0 Knowledge Center](#) to learn about the other elements and attributes set in the server.xml file.

9. Scenarios

Seven scenarios were used in this report to understand the benefits of excluding fields, renaming fields and using ODO/REFERS arrays to limit the size of the JSON payload.

Scenario	Copybook size	# car elements (20 fields)	# dealer elements (4 fields)	Interface rename	JSON response payload size	JSON payload size compared to Baseline
Baseline	30128	100	25	No	63949	(100%)
Exclude dealer array	30128	100	0	No	59634	93%
Exclude car array	30128	0	25	No	4441	7%
Exclude 10 of 20 fields from car array	30128	100 (10 of 20 fields excluded)	25	No	35649	56%
Interface rename	30128	100	25	Yes	38487	60%
ODO 50% cars	30128	50	25	No	34298	53%
ODO 10% cars	30128	10	25	No	10578	17%

Table 1 - Summary of scenarios, copybook and JSON response payload sizes.

Explanation of scenarios:

- **Baseline**

All fields imported into the API toolkit were left untouched. None were excluded, none were renamed, all array elements and their fields were used in the JSON payload. This is equivalent to the service being created through the build toolkit (not recommended) which costs the same in terms of CPU cost per transaction. The 30128 byte copybook maps to a JSON payload size of 63949 bytes.

- **Exclude dealer array**

After importing the copybook the CARMODEL_DEALER_DATA array was excluded from the JSON payload by unchecking the Include checkbox. This array, which has 25 elements, each with four fields, reduced the size of the JSON payload by just over 4000 bytes.

- **Exclude car array**

After importing the copybook the CARMODEL_CAR_DATA array was excluded from the JSON payload by unchecking the Include checkbox. This array, which has 100 elements, each with 20 fields, reduced the size of the

JSON payload by over 59000 bytes.

- **Exclude 10 of 20 fields from car array**

After importing the copybook, 10 of the 20 fields for the CARMODEL_CAR_DATA array were excluded from the JSON payload by unchecking the Include checkbox. This array, which has 100 elements, now each with 10 fields, reduced the size of the JSON payload by over 28000 bytes.

- **Interface rename**

After importing the copybook each field, where possible, was renamed to a shorter name that would be more familiar to a JSON object-oriented programmer. This reduced the JSON response payload size by over 25000 bytes.

- **ODO 50% cars**

In this scenario the number of array elements for the CARMODEL_CAR_DATA array used ODO to request 50 elements rather than 100. The CARMODEL_DEALER_DATA array remained at 25 elements. The overall JSON response payload was over 29000 bytes smaller than the baseline.

- **ODO 10% cars**

In this scenario the number of array elements for the CARMODEL_CAR_DATA array used ODO to request 10 elements rather than 100. The CARMODEL_DEALER_DATA array remained at 25 elements. The overall JSON response payload was over 53000 bytes smaller than the baseline.

10. Results

The following sections of this report show the results of the performance runs for each scenario.

Disclaimer: All performance data in this report was obtained in the specified operating environment and configurations, and must be considered as an example. Performance characteristics of other operating environments might differ.

IBM does not represent, warrant, or guarantee that the same or similar results will be achieved in a user's environment when compared to the results that are shown in this report.

10.1 Transactions Per Second

For each scenario the performance runs simulated:

- 100 clients
- 200 clients
- 300 clients
- 400 clients
- 500 clients

The transactions per second (TPS) for each run was maintained at the following rate:

	100 clients	200 clients	300 clients	400 clients	500 clients
TPS:	200	400	600	800	1000

11. CPU cost per transaction

The cost per transaction in CPU terms is measured in milliseconds (ms).

The calculation is made by dividing the CPU service time by the number of seconds in the SMF interval; this is then divided by the number of transactions per second (TPS) for the same SMF interval; and finally multiplied by one thousand to get the result in milliseconds.

The CPU cost per transaction was the same for 100 to 500 clients running concurrently for any given scenario.

11.1 CPU cost per transaction for each scenario

For each scenario the CPU cost per transaction was measured. The variation in cost was due to the size of the JSON payload (shown in parenthesis under each column in Figure 9).

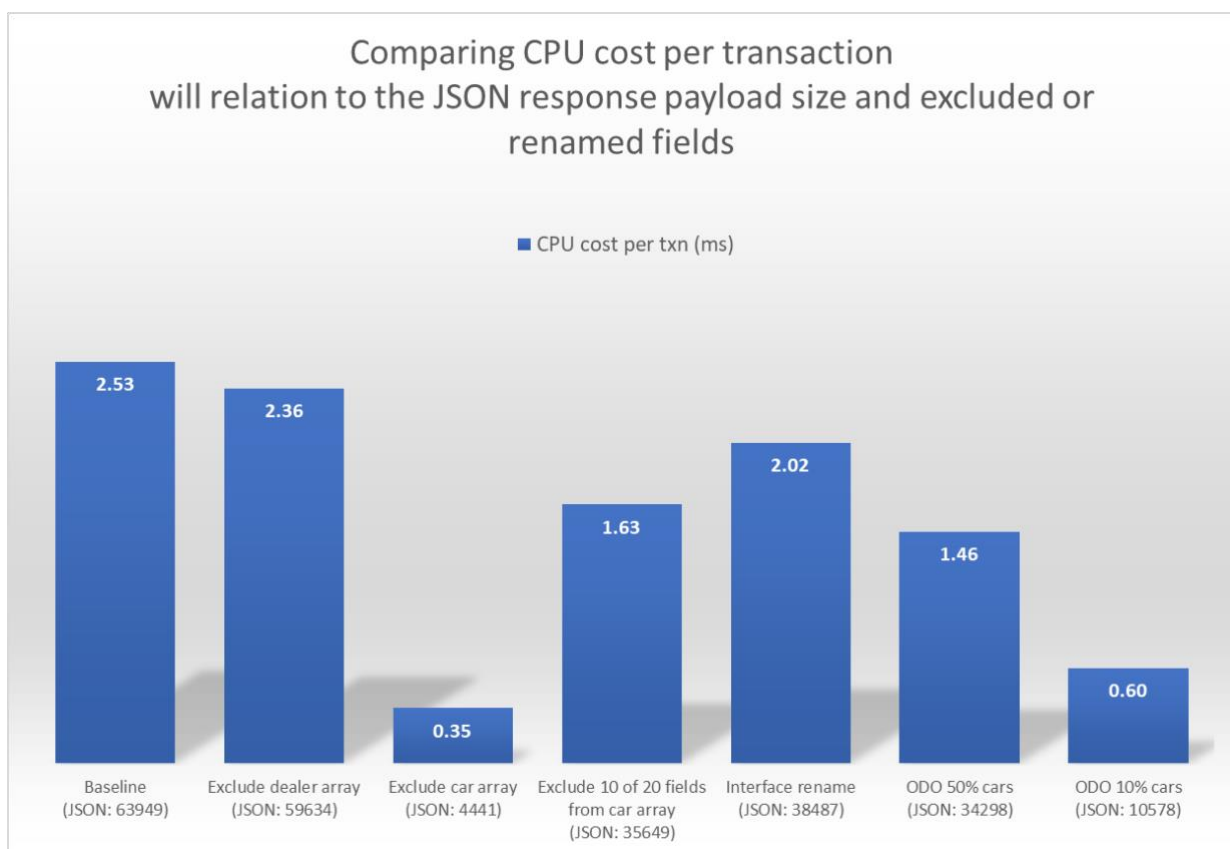


Figure 9 - CPU cost per transaction (ms) for the different scenarios.

Observations

- ✓ Reducing the size of the JSON payload considerably reduced the CPU cost per transaction for some scenarios.
- ✓ Using the Interface rename feature of the API toolkit to reduce the JSON payload size saved some CPU processing, as shown in the *Interface rename* scenario.
- ✓ Using ODO arrays and limiting the amount of required data to be returned to

the client reduced the CPU cost per transaction, as shown in the *ODO 50% cars* and *ODO 10% cars* scenarios.

- ✓ Excluding large unwanted arrays saved a lot of CPU processing, as shown in the *Exclude car array* scenario.
- ✓ Excluding fields in the service interface improved performance, but to achieve greater performance gains, limiting the amount of data retrieved from the SoR is important. For example, in the “Exclude car array” scenario, the query parameter for maxCars was set to zero.
 - When maxCars was set to 100 in the request, even though the car array had been excluded in the service interface, z/OS Connect EE received the data for 100 cars from the SoR due to this value on the maxCars query parameter. Only during the transformation of the response in z/OS Connect EE was the data excluded from the JSON response.
 - When maxCars was set to zero in the request, then the CPU cost per transaction was much improved in z/OS Connect EE and reduced cost in the SoR too.

In the performance test environment used, setting maxCars to 100 resulted in a CPU cost per transaction of 0.64ms, generating wasted CPU usage, whereas when set to zero, the CPU cost per transaction decreased to 0.35ms, as shown in the table below.

Path with query parameters	CPU cost per transaction
models?maxCars=100&maxDealers=25	0.64ms
models?maxCars=0&maxDealers=25	0.35ms

Table 2 - Difference in CPU cost per transaction even though the Car array has been excluded in the service interface

11.2 Varying JSON payload sizes for the ODO car scenario

To understand the effect to the CPU cost per transaction when the request for the number of cars was reduced from the maximum number (100), the *ODO Car* scenario was using varying the number of cars from 20 to 100 in increments of 20.

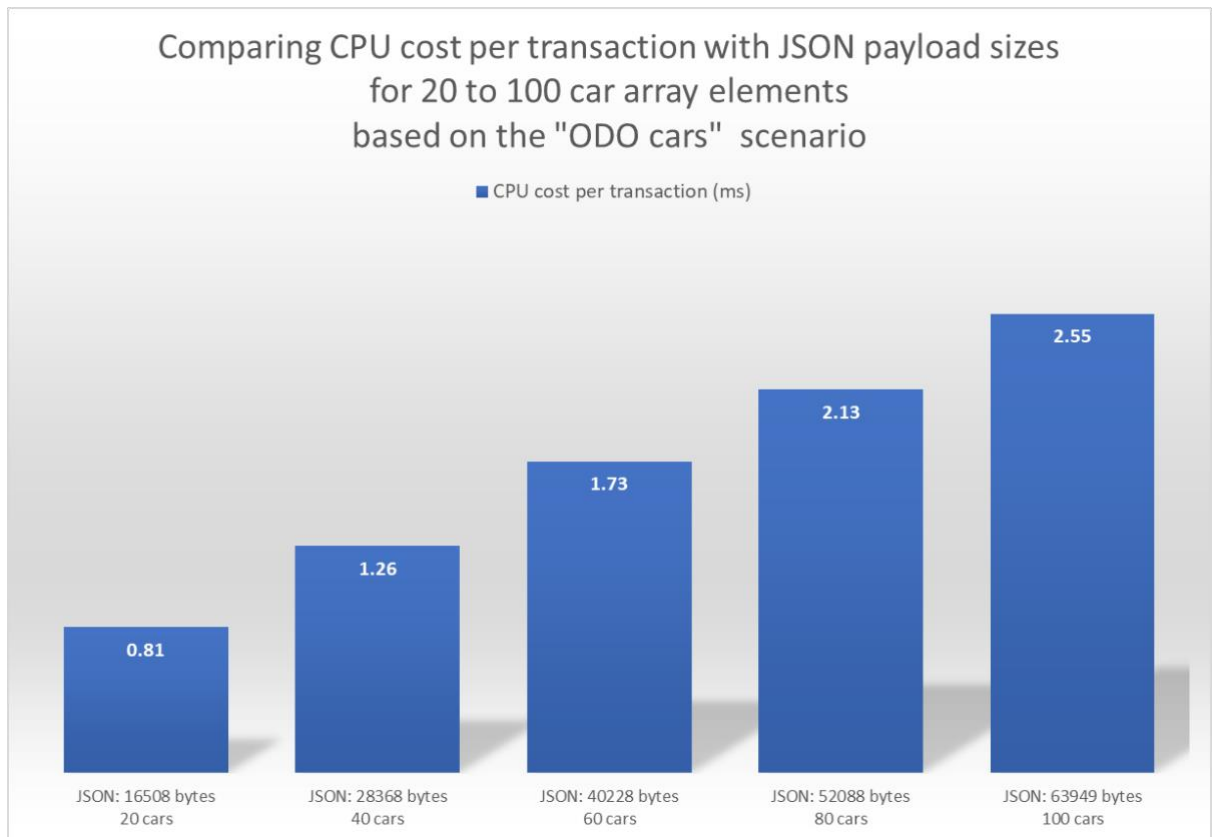


Figure 10 - CPU cost per transaction with increasing JSON payload size for the Car ODO scenario

Observations

- ✓ By using an ODO array and only requesting the amount of data required, reduced the size of the JSON response payload and therefore the CPU cost per transaction.
- ✓ The reduction in JSON payload size scaled linearly as the request for the number of cars was reduced from 100 down to 20.

12. Average Response time

The average response time is measured in milliseconds by the workload driver and calculates the average round trip for the client requests.

12.1 Average Response time for each scenario

The bar chart below shows the average response time for 300 clients. The scenarios were run with 100, 200, 300, 400 and 500 clients. The ratio between the different scenarios remained consistent regardless of the number of clients used in each scenario.

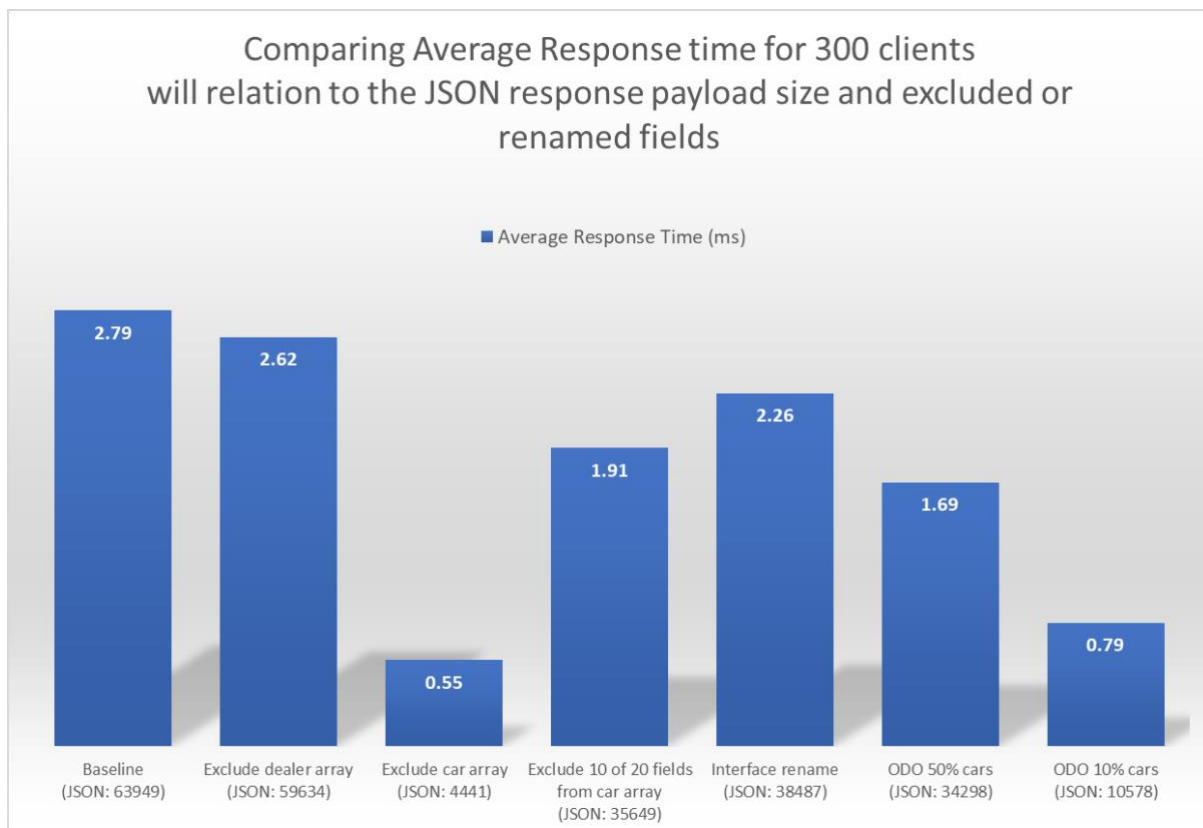


Figure 11 - Average Response time (ms) for the different scenarios

Observations

- ✓ Reducing the size of the JSON payload considerably reduced the average response time for some scenarios.
- ✓ Using the Interface rename feature of the API toolkit to reduce the JSON payload size improved the average response time, as shown in the *Interface rename* scenario.
- ✓ Using ODO arrays and limiting the amount of required data to be returned to the client reduced the average response time, as shown in the *ODO 50% cars* and *ODO 10% cars* scenarios.

- ✓ Excluding large unwanted arrays reduced the average response time, as shown in the *Exclude car array* scenario.
- ✓ Excluding fields in the service interface reduced the average response time, as shown in the *Exclude 10 of 20 fields from car array* scenario.

13. zIIP Eligibility

The zIIP eligibility values were obtained from the SMF 72 records.

As z/OS Connect EE V3.0 is almost entirely written in Java the zIIP eligibility was observed to understand the benefits of offloading work to zIIP engines.

The environment is configured with six GCPs. Although physical zIIPs have not been used, zIIP eligibility was captured and is shown alongside the actual GCP usage.

Figure 12 shows the GCP % usage and zIIP eligibility for the different scenarios.

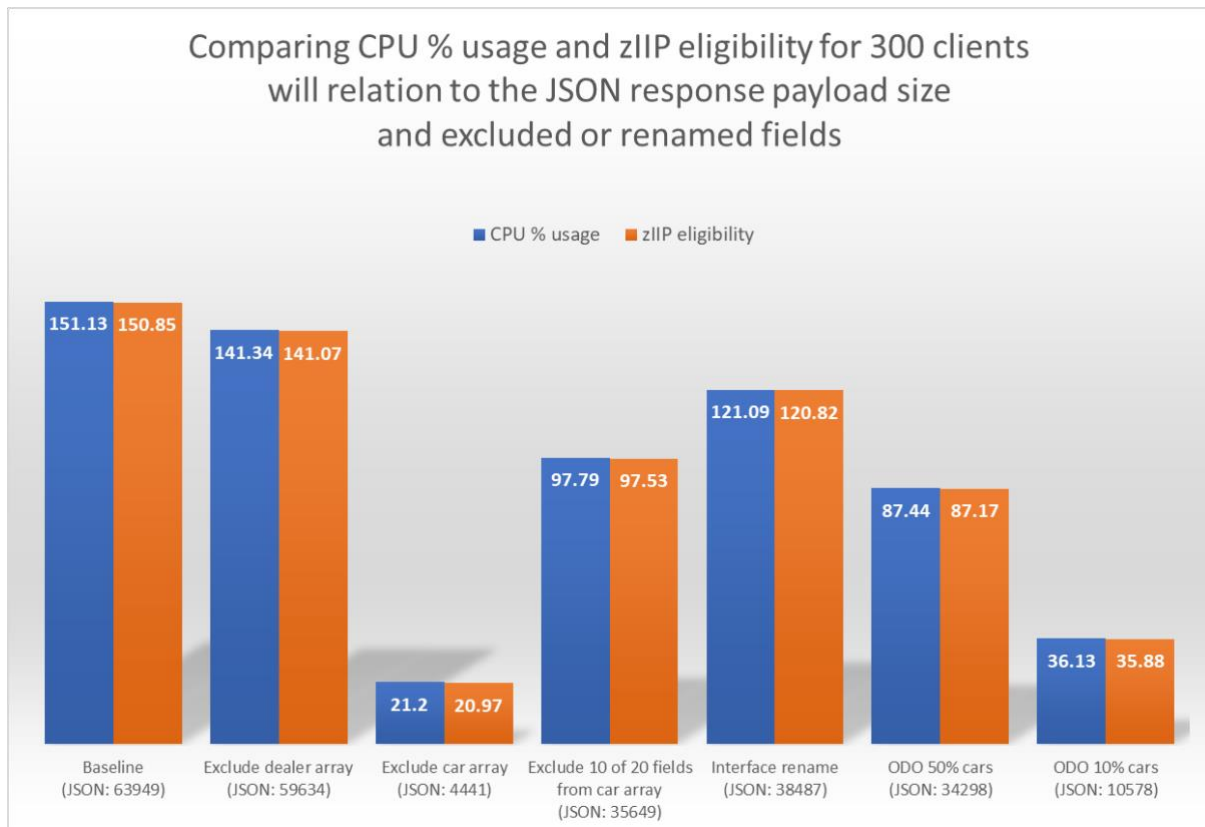


Figure 12 - GCP CPU % and zIIP eligibility for each scenario

Observations

- ✓ z/OS Connect EE is a Java-based product and the majority of the product is eligible to be offloaded to zIIP. Native code is not zIIP eligible.
- ✓ It is important to ensure that the zIIP engines do not become overwhelmed with work. When this happens, zIIP eligible requests are sent back to run on a GCP which is inefficient and costly in terms of overall CPU processing required.

14. Scalability

Some further tests were run to see how z/OS Connect EE scaled with an increasing number of clients to see if there was an impact using multiple arrays, array elements, and fields.

14.1 CPU cost per transaction for increasing number of clients

In each scenario, using a think time of 500ms to gain consistent throughput (transactions per second), the CPU cost per transaction remained constant.

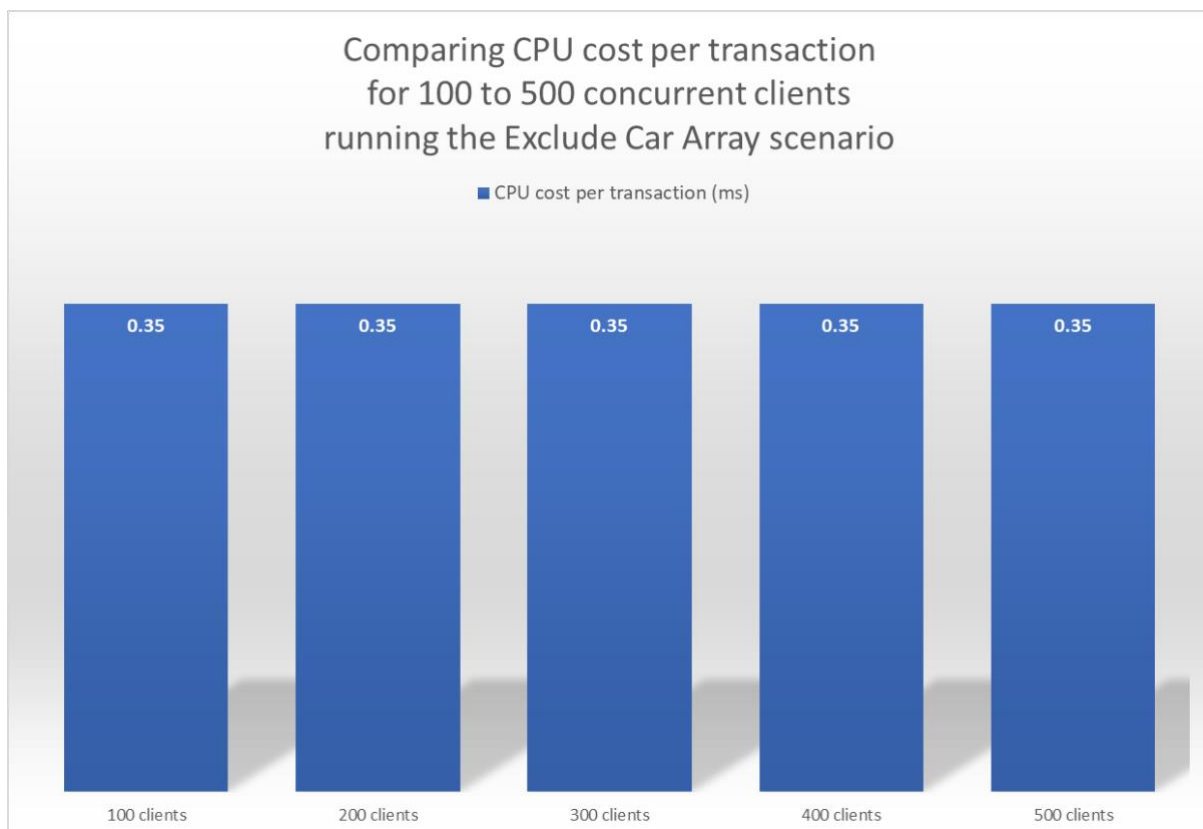


Figure 13 - CPU cost per transaction for the scenario remaining constant

Observation

- ✓ Figure 13 shows that for the *Exclude car array* scenario the CPU cost per transaction remained constant at 0.35ms despite the number of clients increasing from 100 up to 500 clients.

14.2 Average Response time for increasing number of clients

Ideally the average response time should be the same whether there is one client or

500 clients. Realistically, as the number of clients increase, the average response time can be expected to also increase due to resource contention in the system.

Typically a maximum throughput in the system will be reached somewhere in any configuration at which point requests will begin to queue, thus increasing the average response time.

Figure 14 shows the average response for the responses in the *Interface rename* scenario for different numbers of clients and transaction rates.

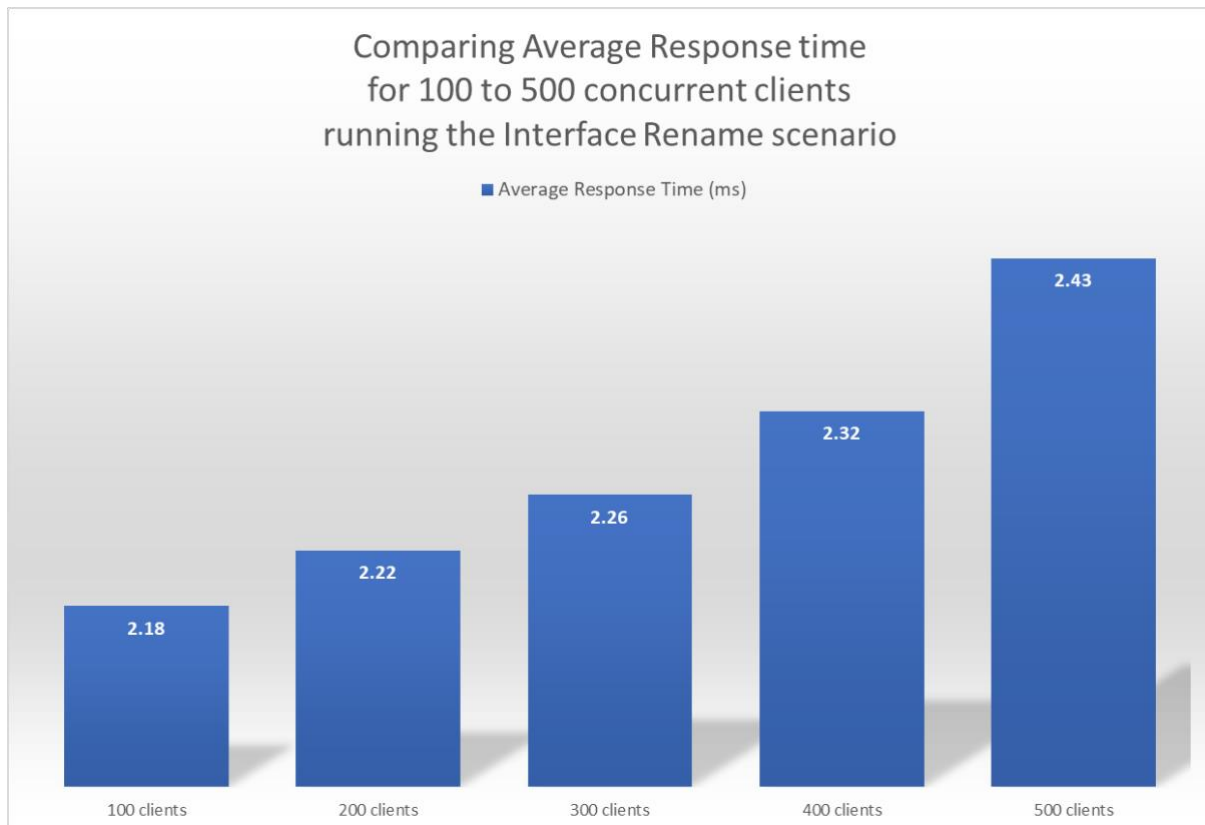


Figure 14 - Average Response for 100 to 500 clients running the Interface Rename scenario.

Observations

- ✓ z/OS Connect EE demonstrated acceptable scalability for average response times.
- ✓ Increasing the number of clients to run in parallel did not create unacceptable response times.

14.3 CPU % usage for increasing number of clients

The CPU % usage will naturally increase as the number of clients running in parallel also increase in line with the increased number of transactions per second.

The CPU % includes all the GCPs (six in this environment), allowing a theoretical maximum of 600% for all products (including z/OS Connect EE, SoR and the

workload driver, which all ran on the same LPAR).

Figure 15 shows the CPU % usage for the z/OS Connect EE server for 100 to 500 clients running concurrently for the *ODO 50% cars* scenario.

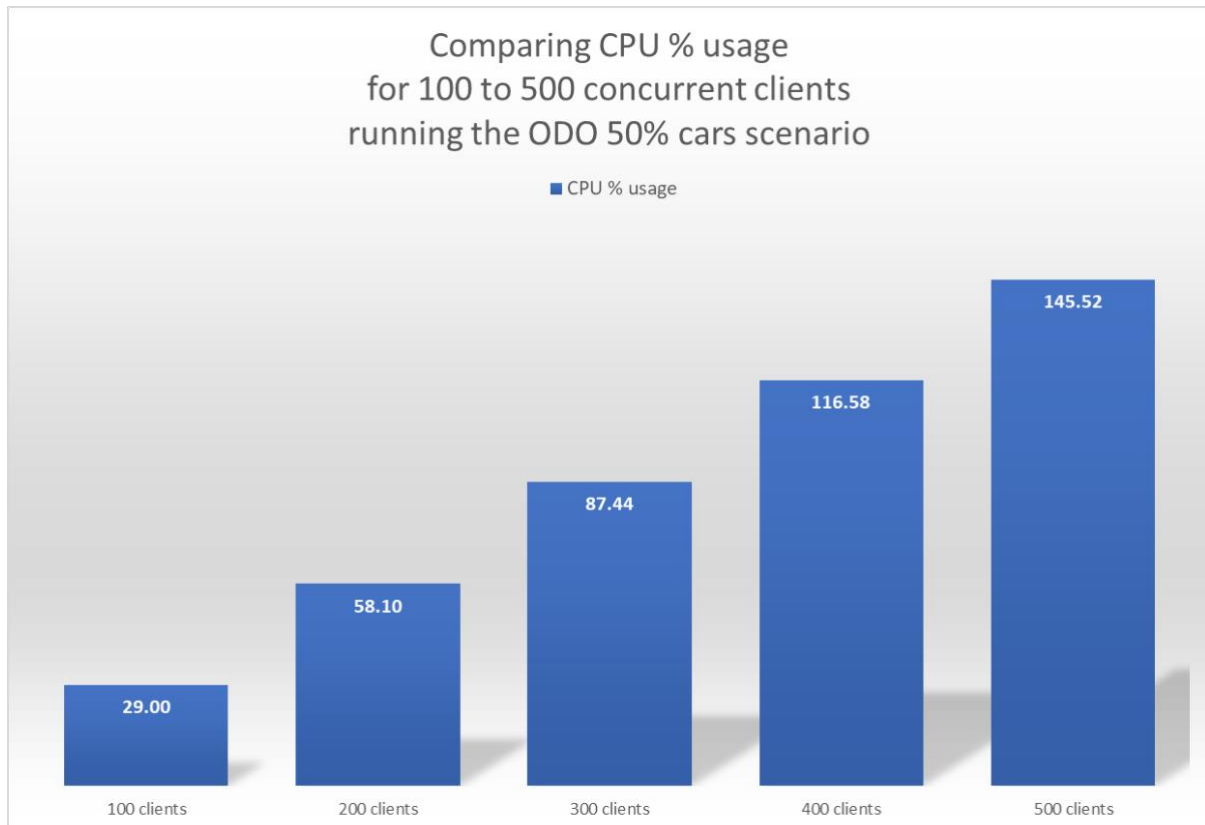


Figure 15 - CPU% usage for increasing number of clients and transaction rates

Observations

- ✓ z/OS Connect EE demonstrated good scalability.
- ✓ The CPU % usage increased linearly as the number of clients running concurrently were increased.

15. Conclusions

The following conclusions can be drawn from this report:

1. Reducing the size of the JSON payload reduces the amount of CPU processing required, which in turn reduces the CPU cost per transaction.
2. The JSON payload can be reduced in size through
 - ✓ Excluding unrequired fields from the JSON payload using the API toolkit *Include* feature.
 - ✓ Using the Interface rename feature within the API toolkit to shorten field names.
 - ✓ Using ODO/REFER arrays and only selecting the number of array elements actually required in the request
3. z/OS Connect EE V3.0 scales well, even when payloads include multiple arrays and large numbers of elements and fields.
4. The majority of z/OS Connect EE V3.0 is written in Java and so benefits can be made by offloading work to a sufficient number of zIIP engines.

Note:

1. Analysis of other payload sizes, datatypes, or running on different hardware, have not been completed at this time, so there is no guarantee that equivalent observations will be seen in other configurations.
2. Due to the effects on system performance of machine hardware, levels of software configuration and variations in payload, equivalent observations might not be seen on other systems.