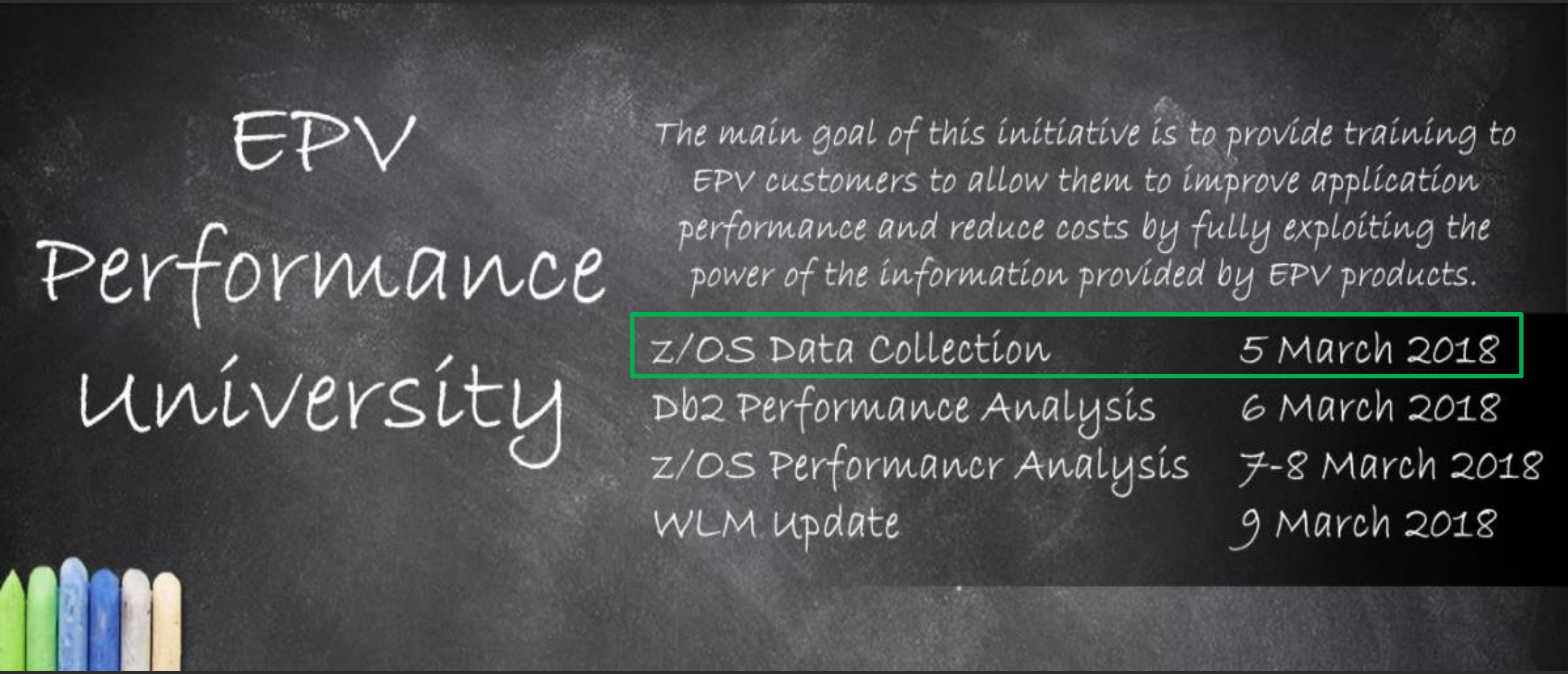
zOS Data Collection for Capacity Management

GSE UK

Sessions LA LB LC LD



EPV Performance University

The main goal of this initiative is to provide training to EPV customers to allow them to improve application performance and reduce costs by fully exploiting the power of the information provided by EPV products.

z/OS Data Collection	5 March 2018
Db2 Performance Analysis	6 March 2018
z/OS Performancr Analysis	7-8 March 2018
WLM update	9 March 2018

Agenda

9:30 - SMF records and more
11:00 - coffee break
11:30 - zParser data collection architecture
12:00 - zParser basic processing
12:30 - zParser advanced processing
13:00 - lunch
14:30 - Processing controls
15:00 - Data recovery
15:30 - coffee break
16:00 - zParser for Big Data
16:30 - end of day

SMF records

SMF records - Overview

- System Management Facility (SMF) is a base component of z/OS
- It provides a set of macros used by subsystems and applications to pass records to the SMF address space for recording into SMF records
- Each record is identified by a one-character record type (hex value ‘00’ to ‘FF’) between 0 and 255
- IBM has exclusive use of record types 0 through 127
- Third part software and user applications may use 128 to 255

SMF records - Overview

- Extended SMF records in z/OS 2.3
- SMF will now support 2048 unique record types
  - Types 0-127 and 1152-2047 are reserved for IBM
  - Types 128-1151 are available for use external to IBM
- SMFPRMxx: no change in the TYPE option default of TYPE(0:255)
- SMF dump utilities (IFASMFDP and IFASMFDL): new default of TYPE(0:2047)

SMF records - Overview

- EPV zParser supports:
  - All SMF "standard" records (0-127)
  - Many "user" SMF records (128-255)
  - IMS logs, DCOLLECT and BVIR records are also supported
  - CSV files
- Other "user" SMF records are supported on request; customers have to provide some data for testing; record layout information should be available
- Extended SMF records support in development

SMF records - Overview

- Several of the record types also have subtypes to identify unique types of records (e.g. SMF 30)
- SMF records layout is complex and not human readable; it is described in the SMF manual but the specific layout of some important records (CICS, Db2, etc) is not provided; you get only reference such as:

CICS Transaction Server for z/OS writes record type 110 to record transaction data collected at event monitoring points. For more information about record type 110, see the customization documentation available at the following URL:

http://www-01.ibm.com/software/htp/cics/library/

From MVS System Management Facilities (SMF)

SMF records - Overview

- Most important SMF records for Performance Analysis:
  - ✓ SMF 30 records for Address Spaces
  - ✓ RMF (CMF) records 70-78
  - ✓ SMF 113
  - ✓ SMF 110 for CICS, 100-102 for Db2, 120 for WebSphere, 115-116 for MQ
- Very good SMF reference available on Cheryl Watson's web site http://watsonwalker.com/publications/

SMF records - Overview

- Not all the relevant information for z/OS performance analysis are written to SMF; major exceptions are:
  - ✓ IMS transactions information which are written to the IMS log
  - ✓ Disk space information which can be collected by running IDCAMS with the DCOLLECT parameter
  - ✓ IBM VTS information which can be collected via the Bulk Volume Information Retrieval (BVIR) function

SMF records - Parameters

- All the SMF parameters are set in the SMFPRMxx member of SYS1.PARMLIB
- For a record type and subtype to be written it has to be allowed in the TYPE sub-parameter under the SYS or SUBSYS parameters
  - ✓ SYS (… TYPE(0:255) …)
  - ✓ SYS (… TYPE(0:100, 101(0), 102,110,115:120) …)

SMF records - Parameters

- Other SMFPRMxx parameters influencing the content and the format of SMF records:
  - ✓ INTVAL, SYNCVAL, INTERVAL (to be discussed later)
  - ✓ EMPTYEXCPSEC; to suppress empty EXCP sections in SMF 30; default is NOSUPPRESS
  - ✓ DDCONS; to consolidate duplicate EXCP sections in SMF 30 records (same DD and address); default is YES; you should change it to NO
  - ✓ SMF30COUNT; to write instructions counters in SMF 30; default is NOSMF30COUNT

SMF records – New parameters

- NOCOMPRESS | COMPRESS[(PERMFIX(nnnnM))]
  - COMPRESS is an optional parameter. When specified with a zEDC Express feature available, SMF compresses SMF records before writing to the log stream; **log stream only**
  - PERMFIX is an optional parameter when COMPRESS is specified; it specifies the default amount of storage that SMF can keep permanently fixed for communicating with zEDC
  - **Default:** NOCOMPRESS, NOPERMFIX

SMF records – New parameters

- NORECSIGN | RECSIGN(HASH(SHA1|SHA256|SHA384|SHA512), SIGNATURE(RSA|ECDSA),TOKENNAME(tokenname))
  - Specifies whether SMF is to digitally sign the records that are being recorded for the log stream; **log stream only**
  - If RECSIGN is specified, the HASH, TOKENNAME, and SIGNATURE keywords must also be specified
  - **Default:** NORECSIGN

SMF records – New parameters

- The steps needed to set up and use digitally signed SMF records are:
  - Create the public/private key pair
  - Update the SMFPRMxx member of parmlib to specify that you want SMF to sign records
  - Use the IFASMFDL dump program to carry signature records to data sets (default: SIGSTRIP)
  - Use the IFASMFDP dump program to carry signature records to data sets and validate records (default: SIGSTRIP, NOVALIDATE)

See "SMF Digital Signatures in zOS 2.2"

SMF records – New parameters

- INMEM(rname,RESSIZMAX({nnnnM|nG}),{TYPE({aa,bb|aa,bb:zz|aa,bb:zz,...})|NOTYPE({aa,bb|aa,bb:zz|aa,bb:zz,...})}
  - Defines an in-memory resource to record SMF records in memory for real-time processing; **log stream only**
  - A maximum of 32 in-memory resources is supported
  - Record types specified on an INMEM parameter are not written **to the default logstream**; if you want certain record types to be written also to a logstream, specify those record types on both an LSNAME parameter
- **Default:** none

SMF records – New parameters

- rname
  - Name of the in-memory resource. The resource name must begin with IFASMF. and can be up to 26 characters long; the resource name must be unique across resource names on other INMEM statements and logstream names on LSNAME statements
  - You must also define a SAF resource in the FACILITY class to protect the in-memory resource; the SAF resource profile name must start with the IFA. high-level qualifier, followed by the same name that you specify for rname

Default: none

SMF records – New parameters

- RESSIZMAX({nnnnM|nG})
  - Specifies the size of the buffer available for this in-memory resource, in megabytes or gigabytes.
  - The in-memory resource acts as a wrap-around buffer. When the buffer is full, older records are discarded as newer records are written.

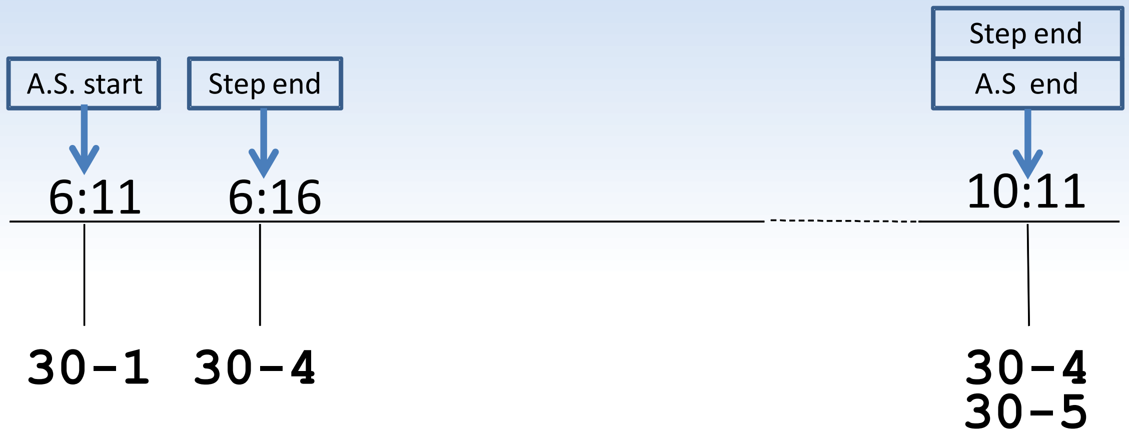
Default: 2 GB

SMF records – SMF 30

- SMF type 30 is one of the most important SMF records. It provides extensive information about any kind of address space (JOBS, STC, TSO) activity and resource usage
- By default the following SMF 30 record subtypes are produced:
  - ✓ subtype 1 – written at address space initiation
  - ✓ subtype 4 – written at step termination
  - ✓ subtype 5 – written at address space termination

SMF records – SMF 30

- Milestones of a two steps address space execution running from 6:11 up to 10:11



- A total of 4 SMF 30 records are created:
  - ✓ 1 subtype 1 at address space start
  - ✓ 2 subtype 4 at each step end
  - ✓ 1 subtype 5 at address space end

SMF records – SMF 30

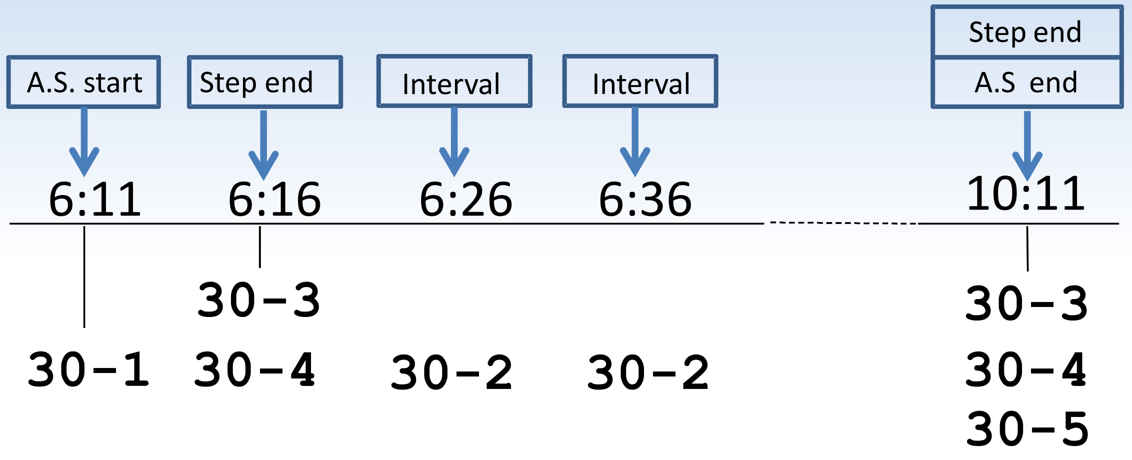
- The first SMF 30 subtype 4 record provides information about address space activity in the first step (5 minutes elapsed)
- The second SMF 30 subtype 4 record provides the same information about the second step (remaining 3 hours and 55 minutes)
- Now let's suppose this second step used 2.000 CPU seconds and you have different accounting policies for CPU used before and after 8:00, how can you manage that ?

SMF records – SMF 30

- To address this kind of issue, SMF interval accounting has been introduced; SMF interval accounting produces the following additional 30 records:
  - ✓ subtype 2; it is written at a user defined interval; it collects information about address space activity during the previous interval
  - ✓ subtype 3; it is written when steps end; it collects information about address space activity from the end of the previous interval up to the step end
  - ✓ subtype 6; it is written at a user defined interval; it collects information about some “early” system address spaces, such as CONSOLE or GRS, started before SMF; **note that all these measurements are accumulated**

SMF records – SMF 30

- Milestones of a two steps address space execution running from 6:11 up to 10:11 (interval accounting)



✓ A subtype 2 interval record (every 10 minutes in this example) is produced providing much more granular information

✓ Two subtype 3 records are also produced

SMF records – SMF 30

- By default SMF 30 interval accounting is not activated (subtype 2 and 3 records are not produced)
- To activate SMF interval accounting the following parameters have to be set in the SMFPRMxx member of the SYS1.PARMLIB library:
  - ✓ INTVAL(mm) where mm is the interval duration; suggested values are 10 or 15 minutes;
  - ✓ SYNCVAL(nn) where nn is the minute in the hour that starts the interval; suggested value is 00

SMF records – SMF 30

- It’s very important that you also set a global recording interval; this way all the address spaces will write SMF 30 subtype 2 at the same time at the end of the interval
- To use a global recording interval the following additional parameter have to be set in the SMFPRMxx member of the SYS1.PARMLIB library under the SYS and SUBSYS sections:

```
INTERVAL(SMF,SYNC)
```

SMF records – SMF 30

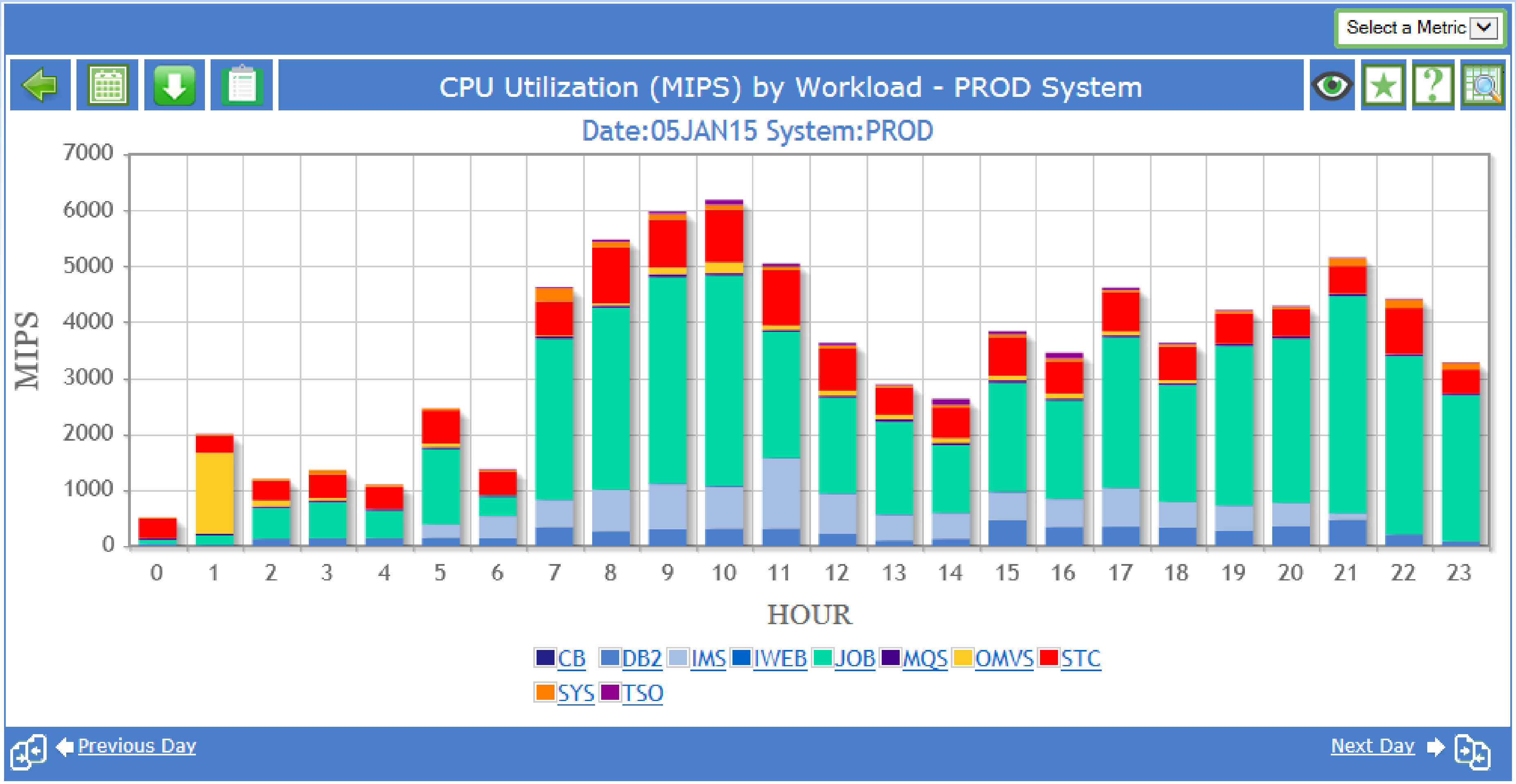
TOP 10 JOBS - MIPS used																								
JOB NAME	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
job1										169	524	377	283	221	520	404	329							
job2				150	588	114						34	468	483	501	432								
job3						343	259						422											
job4																				295				
job5					282	31		405	427															
job6										355	207													
job7																					265			
job8								253																
job9									222	250	318	172	306	136	293	205	148	332	272					
job10	132																			5	323	318	228	353

- Using the global recording interval will make SMF 30 very useful for performance analysis

SMF records – SMF 30

- By exploiting some of the metrics available in SMF 30 interval records such as the task type, the program name, the Address Space name you can aggregate Address Spaces in workloads
- EPV does that automatically but you can customize that
- This aggregation can be very useful for performance analysis

SMF records – SMF 30



SMF records – SMF 30

- Exercise 1

  A two step batch job runs in a system with the following SMF settings: INTVAL(10), SYNCVAL(00), INTERVAL(SMF, SYNC)

  First step starts at 8:01 and ends at 8:02, second step starts at 8:02 and ends at 9:05. How many SMF 30 records will be written for:

  ✓ Subtype 1

  ✓ Subtype 2

  ✓ Subtype 3

  ✓ Subtype 4

  ✓ Subtype 5

  ✓ Subtype 6

SMF records – SMF 30

- Exercise 1

  A two step batch job runs in a system with the following SMF settings: INTVAL(10), SYNCVAL(00), INTERVAL(SMF, SYNC)

  First step starts at 8:01 and ends at 8:02, second step starts at 8:02 and ends at 9:05. How many SMF 30 records will be written for:

  ✓ Subtype 1 → 1

  ✓ Subtype 2 → 6

  ✓ Subtype 3 → 2

  ✓ Subtype 4 → 2

  ✓ Subtype 5 → 1

  ✓ Subtype 6 → 0

SMF records - RMF

- RMF records historically produced by RMF MON I in SMF 70:78
- Many new record subtypes added by RMF MON III in SMF 72 and especially in SMF 74
- It’s also very important to synchronise SMF and RMF data by setting the SYNC(SMF) parameter in the ERBRMFxx member, used by RMF Monitor I and III, in your SYS1.PARMLIB library
- SMF 79 can also be used to store RMF MON II information; you have to start a Monitor II background session with the RECORD option

SMF records - RMF

Type	SubType	Description	Default	Used by EPV	Notes
70	1	CPU, PR/SM and ICF Activity	CPU	YES	
	2	Cryptographic Hardware Activity	CRYPTO	YES	
71	1	Paging Activity	PAGING	YES	
72	3	Workload Activity	WKLD	YES	
	4	Storage Data		NO	RMF MON III
	5	Serialization Delay		NO	RMF MON III
73	1	Channel Path Activity	CHAN	YES	
74	1	Device Activity	DEVICE(DASD)	YES	
	2	XCF Activity		YES	RMF MON III
	3	OMVS Kernel Activity		YES	RMF MON III
	4	Coupling Facility Activity		YES	RMF MON III
	5	Cache Subsystem Device Activity	CACHE	YES	

- Serialization delay views will be included in future EPV for z/OS versions

SMF records - RMF

Type	SubType	Description	Default	Used by EPV	Notes
74	6	HFS Statistics		NO	RMF MON III
	7	FICON Director Statistics	NOFCD	YES	
	8	Enterprise Disk System Statistics	NOESS	YES	
	9	PCI Express Based Function Activity		YES	RMF MON III
	10	Storage Class Memory Statistics			RMF MON III
75	1	Page Data Set Activity	PAGESP	YES	
76	1	Trace Activity	NOTRACE	NO	
77	1	Enqueue activity	NOENQ	NO	
78	2	Virtual Storage Activity	VSTOR(S)	YES	
	3	I/O Queuing & HyperPAV Activity	IOQ(DASD)	YES	

- PCI Express and Storage Class Memory views have been included in EPV for z/OS V14

SMF records – SMF 113

- The CPU Measurement Facility has been introduced with z10 machines.
- Together with the z/OS Hardware Instrumentation Services (HIS) it provides the ability to gain measurements (counters and samples) on the processor cache architecture components
- Output is provided in SMF 113
- SMFINTVAL=SYNC to synchronize with other SMF interval records is provided

SMF records – SMF 113

- To collect these measurements you need to perform the following steps:
  - ✓ authorize the sampling facilities and counter set through the support element (SE) console;
  - ✓ define a user ID for the HIS started task;
  - ✓ create the HOME directory of the user ID;
  - ✓ enable SMF record type 113 in SMFPRMxx;
  - ✓ start the HIS by executing the S HIS command;
  - ✓ activate data collection by issuing the following command:

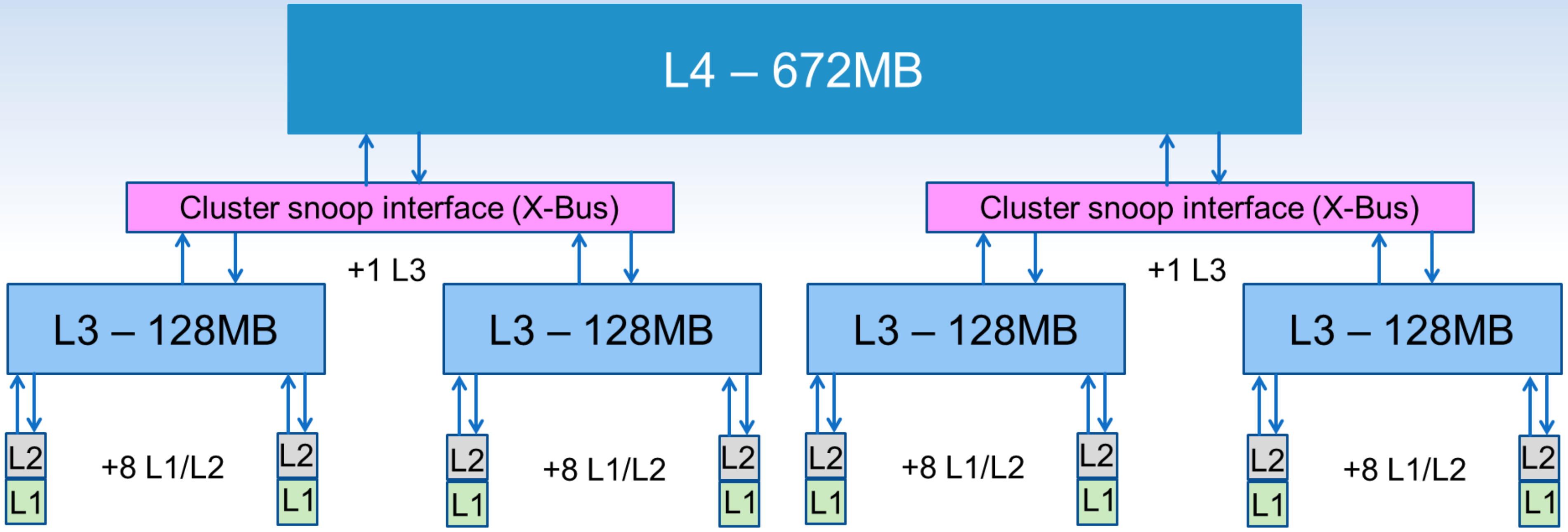
F HIS,B,TT=‘text',PATH='/var/his',CTRONLY,CTR=ALL,SI=SYNC

SMF records – SMF 113

- The main goal of HIS counters is to provide measurements on processor cache effectiveness to calculate L1 Miss and RNI and choose the right benchmark for your workload
- Starting the HIS address space you can collect accumulated measurements and record them in SMF 113 subtype 2
- Since z/OS 2.1 SMF 113 subtype 1 is also written providing de-accumulated measurements

SMF records – SMF 113

z14 processor cache architecture



L2	Instructions	2MB
	Data	4MB
L1	Instructions	128K
	Data	128K

SMF records – SMF 113

- Most important groups of counters are:
  - ✓ basic counters; which should be used to calculate the percentage of L1 misses; **same counters for every machine**
  - ✓ extended counters; which should be used to calculate the percentage of L1 misses sourced by each cache level and, starting from them, the RNI value; **specific counters for each machine**

SMF records – SMF 113

- BASIC COUNTERS:
  - ✓ B0, CYCLE COUNT
  - ✓ B1, INSTRUCTION COUNT
  - ✓ B2, L1 I-CACHE DIRECTORY-WRITE COUNT
  - ✓ B3, L1 I-CACHE PENALTY CYCLE COUNT
  - ✓ B4, L1 D-CACHE DIRECTORY-WRITE COUNT
  - ✓ B5, L1 D-CACHE PENALTY CYCLE COUNT

$$\%L1\ Miss = ((B2 + B4) / B1) * 100$$

SMF records – SMF 113

- z14 EXTENDED COUNTERS (1/2)

The number of L1 misses sourced by each cache levels can be calculated as follows:

- ✓ L2d, data sourced from L2 = E133;
- ✓ L2i, instructions sourced from L2 = E136;
- ✓ L3d, data sourced from L3 = E144 + E146 + E158;
- ✓ L3i, instructions sourced from L3 = E162 + E164;
- ✓ L4Ld, data sourced from L4 Local = E147 + E149 + E156 + E150 + E152;
- ✓ L4Li, instructions sourced from L4 Local = E165 + E167 + E174 + E168 + E170;

SMF records – SMF 113

- z14 EXTENDED COUNTERS (2/2)

The number of L1 misses sourced by each cache levels can be calculated as follows:

- ✓ L4Rd, data sourced from L4 Remote = E153 + E155 + E157;
- ✓ L4Ri, instructions sourced from L4 Remote = E171+ E173 + E175;
- ✓ MEMLd, data sourced from Local Memory = E145 + E148 + E151;
- ✓ MEMRd, data sourced from Remote Memory = E154;
- ✓ MEMLi, instructions sourced from Local Memory = E163 + E166 + E169;
- ✓ MEMRi, instructions sourced from Remote Memory = E172.

SMF records – SMF 113

- Starting from these measurements the percentage of z14 L1 misses sourced by each cache level can be calculated by using the following formulas:
  - ✓ %L2 = (L2d + L2i) / (B2 + B4) * 100
  - ✓ %L3 = (L3d + L3i) / (B2 + B4) * 100
  - ✓ %L4L = (L4Ld + L4Li) / (B2 + B4) * 100
  - ✓ %L4R = (L4Rd + L4Li) / (B2 + B4) * 100
  - ✓ %MEM = (MEMLd + MEMLi + MEMRd + MEMRd) / (B2 + B4) * 100

SMF records – SMF 113

- RNI (Relative Nest Intensity) is an index measuring how much a workload stresses the nest (shared processor caches and memory)

$$z14\ \mathit{RNI} = \mathbf{2{,}4} \times (\mathbf{0{,}4} \times \%L3 + \mathbf{1{,}5} \times \%L4L + \mathbf{3{,}2} \times \%L4R + \mathbf{7{,}0} \times \%MEM) \,/\, 100$$

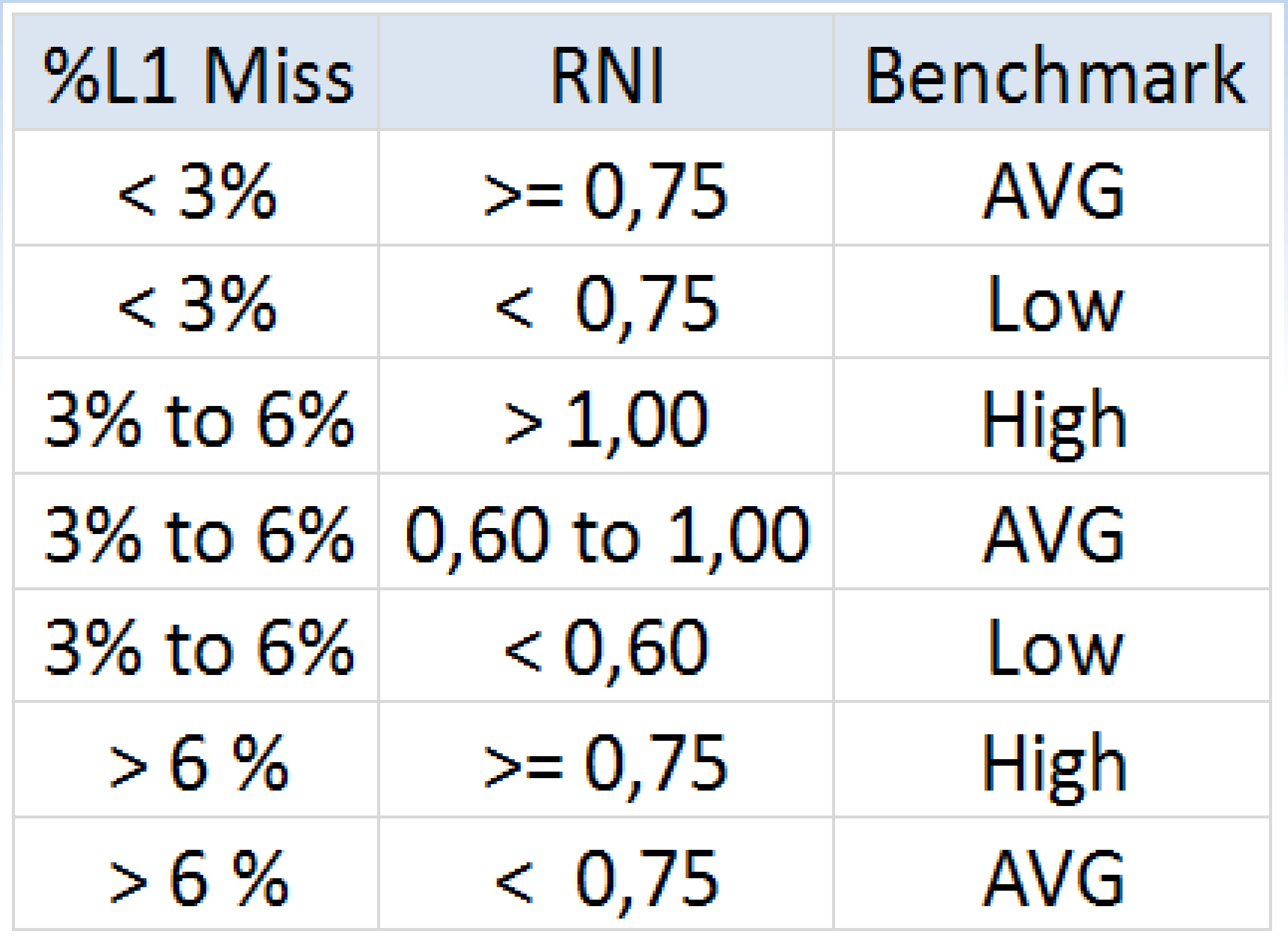
- The coefficients (in bold) are used to weight cache and memory accesses (IBM may change them)
- Also this formula is machine dependent

SMF records – SMF 113

Benchmarks

- LOW RNI (Relative Nest Intensity): it represents workloads lightly using the memory nest (shared processor caches and memory) hierarchy. It is similar to past high scaling primitives
- AVERAGE RNI (Relative Nest Intensity): it represents workloads with an average use of the memory nest. It is similar to the past LoIO-mix workload and is expected to represent the majority of production workloads
- HIGH RNI (Relative Nest Intensity): this category represents workloads heavily using the memory nest. It is similar to the past DI-mix workload

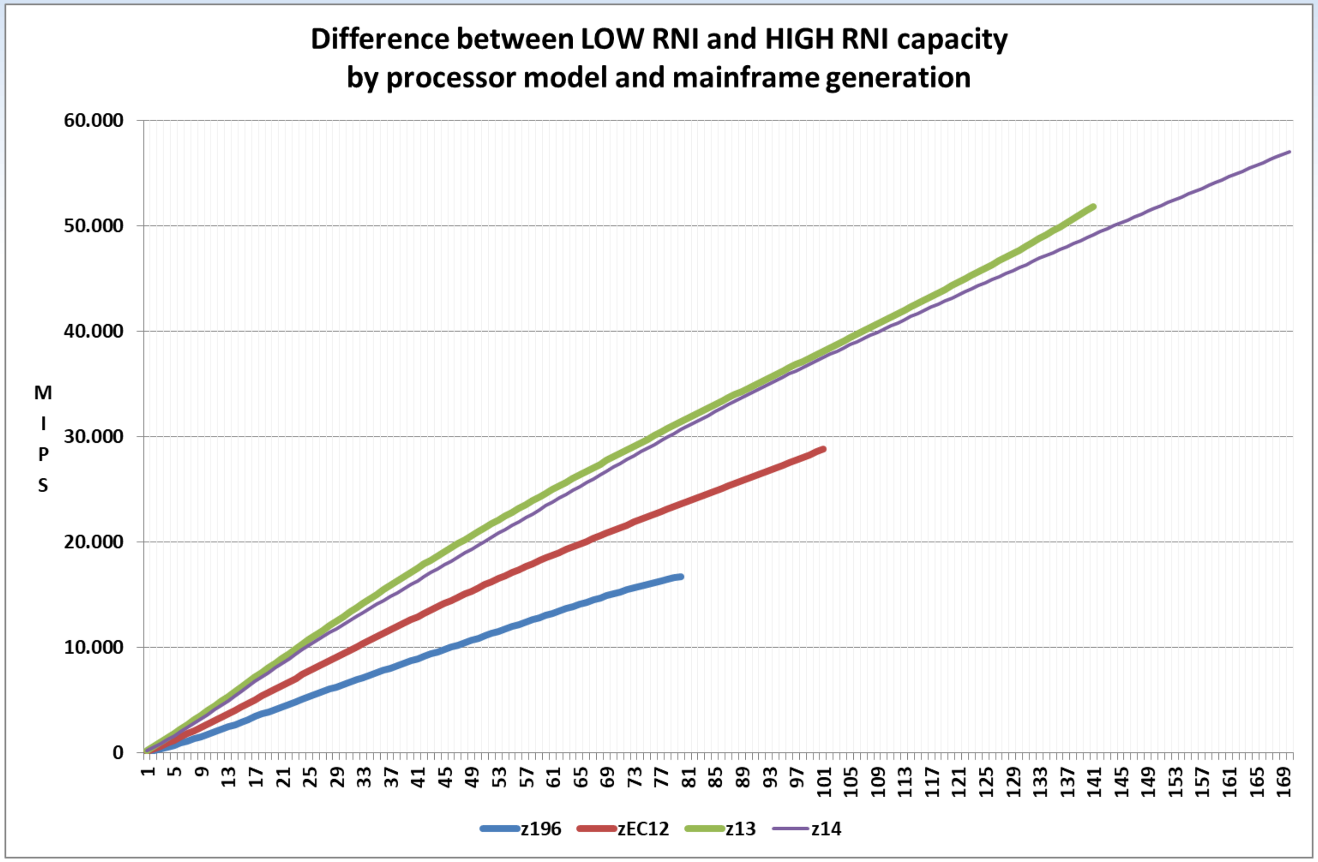
SMF records – SMF 113



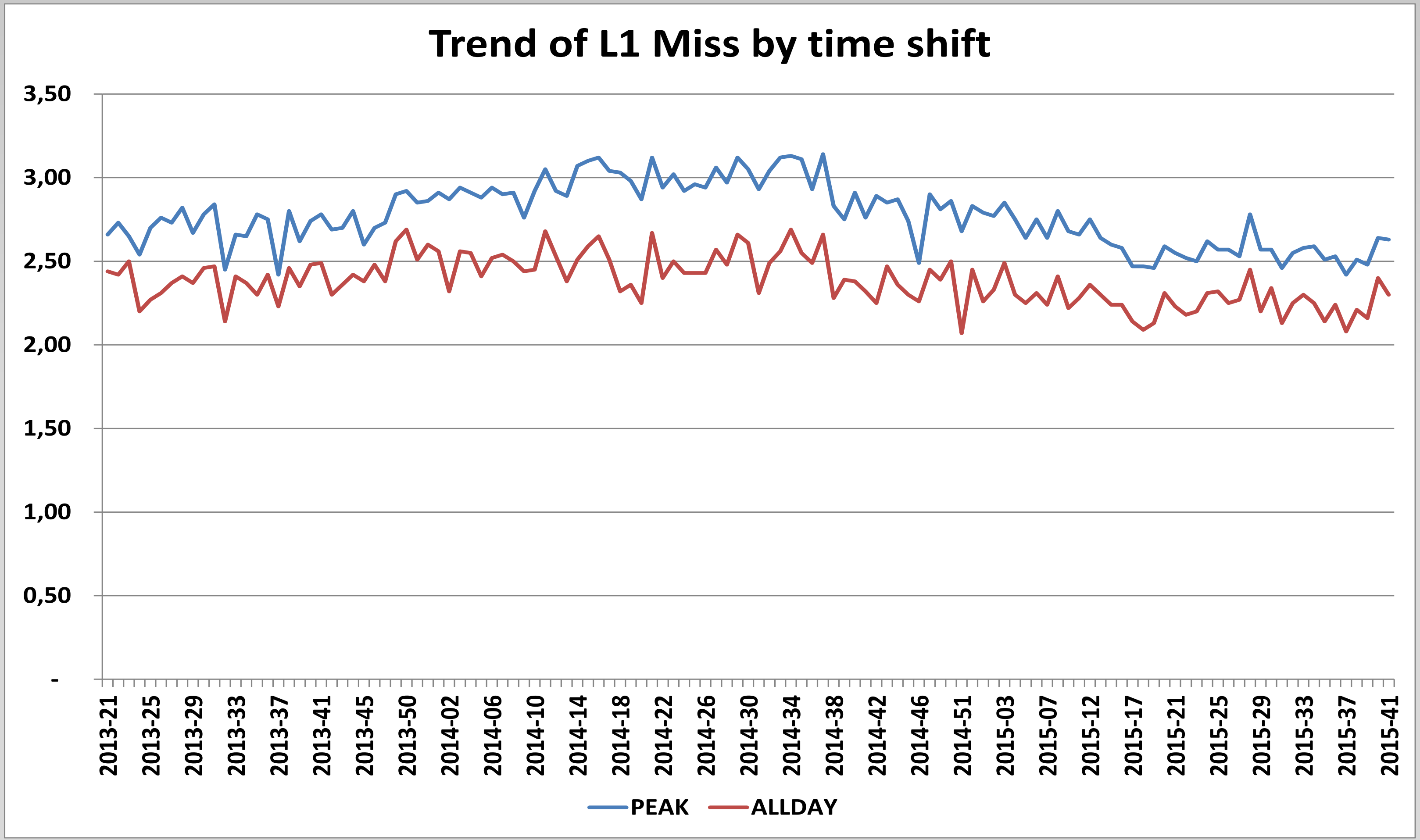
%L1 Miss	RNI	Benchmark
< 3%	>= 0,75	AVG
< 3%	< 0,75	Low
3% to 6%	> 1,00	High
3% to 6%	0,60 to 1,00	AVG
3% to 6%	< 0,60	Low
> 6 %	>= 0,75	High
> 6 %	< 0,75	AVG

SMF records – SMF 113

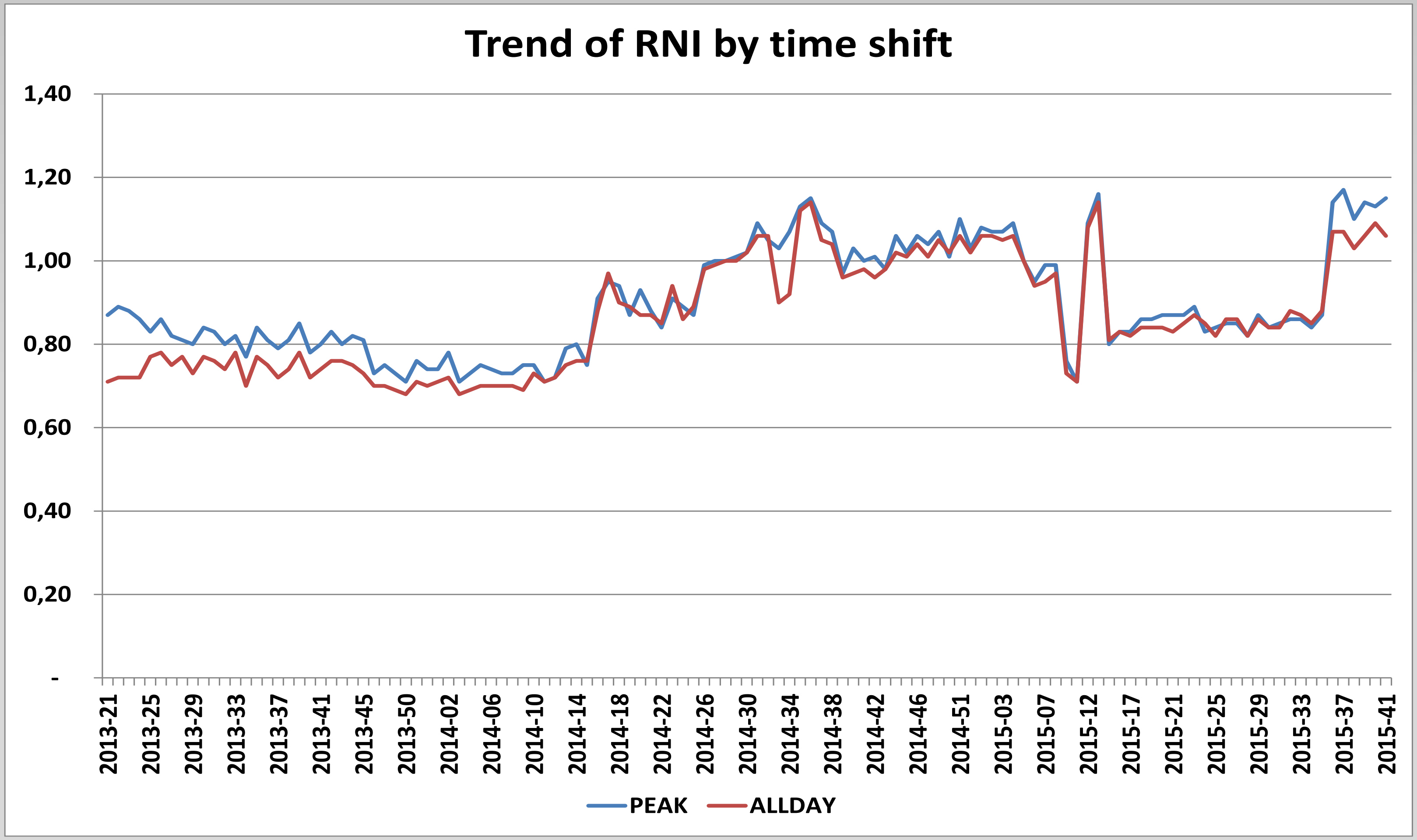
- Why is important choosing the right benchmark?



SMF records – SMF 113



SMF records – SMF 113



SMF records – SMF 113

- SMF 113 record layout has to be obtained by integrating the information provided in the following manuals
  - ✓ SMF
  - ✓ The Load-Program-Parameter and the CPU-Measurement Facilities – SA23-2260-05
  - ✓ The CPU-Measurement Facility Extended Counters Definition for z10, z196/z114, zEC12/zBC12, z13/z13s and z14– SA23-2261-04

SMF records – Db2

- The Db2 Instrumentation Facility Component (IFC) provides a powerful trace facility that you can use to record Db2 data and events
- These metrics are extremely useful to control and tune Db2 subsystems and application performance
- Unfortunately the volume of trace data can be quite large and the overhead to produce them can impact system performance
- Managing and processing this data can also require a lot of additional system resources

SMF records – Db2

- Each trace is broken down into classes. The type and amount of information produced depends on the activated classes
- Generally the activation of a class causes the production of one or more Instrumentation Facility Component Identifier (IFCID) record
- The description of the START TRACE command (in Db2 Command Reference) indicates which IFCIDs are activated for the different types of trace and the classes within those trace types

SMF records – Db2

- Statistics and Accounting traces can produce many IFCIDs but their mapping to SMF records is pretty simple:
  - ✓ most of the statistics trace IFCIDs are collected in SMF type 100
  - ✓ accounting trace IFCIDs are collected in SMF type 101
  - ✓ all the other IFCIDs are collected in SMF type 102

SMF records – Db2

- Statistics IFCIDs used by EPV for Db2

IFCID	TRACE	CLASS	SMF TYPE	SMF SUBTYPE	DESCRIPTION
001	STATISTICS	1	100	0	SYSTEM SERVICES STATISTICS
002	STATISTICS	1	100	1	DATABASE STATISTICS
202	STATISTICS	1	100	2	BUFFER POOL PARAMETERS
230	STATISTICS	5	100	3	DATA SHARING GLOBAL STATISTICS
225	STATISTICS	1	100	4	STORAGE STATISTICS
172	STATISTICS	3	102		DEADLOCK STATISTICS
196					TIMEOUT STATISTICS
105					DB TS MAPPING
199	STATISTICS	7	102		DATA SET I/O STATISTICS

SMF records – Db2

- Accounting IFCIDs used by EPV for Db2
- IFCID 003 is used by EPV for z/OS to report DDF throughput only

IFCID	TRACE	CLASS	SMF TYPE	SMF SUBTYPE	DESCRIPTION
003	ACCOUNTING	1	101	0	PLAN ACCOUNTING
003	ACCOUNTING	2	101	0	PLAN IN DB2 TIME
003	ACCOUNTING	3	101	0	PLAN WAIT TIME
239	ACCOUNTING	7	101	1	PACKAGE ACCOUNTING
239	ACCOUNTING	8	101	1	PACKAGE WAIT TIME
239	ACCOUNTING	10	101	1	PACKAGE ACCOUNTING DETAILS

SMF records – MQ

- Specific SMF records, allowing you to measure MQ infrastructure and workload, in detail, are available:
  - ✓ SMF 115 providing statistics information
  - ✓ SMF 116 providing accounting information
- SMF data collection is not active by default
- You have to request it by setting to YES the SMFSTAT and SMFACCT parameters provided in the CSQ6SYSP macro

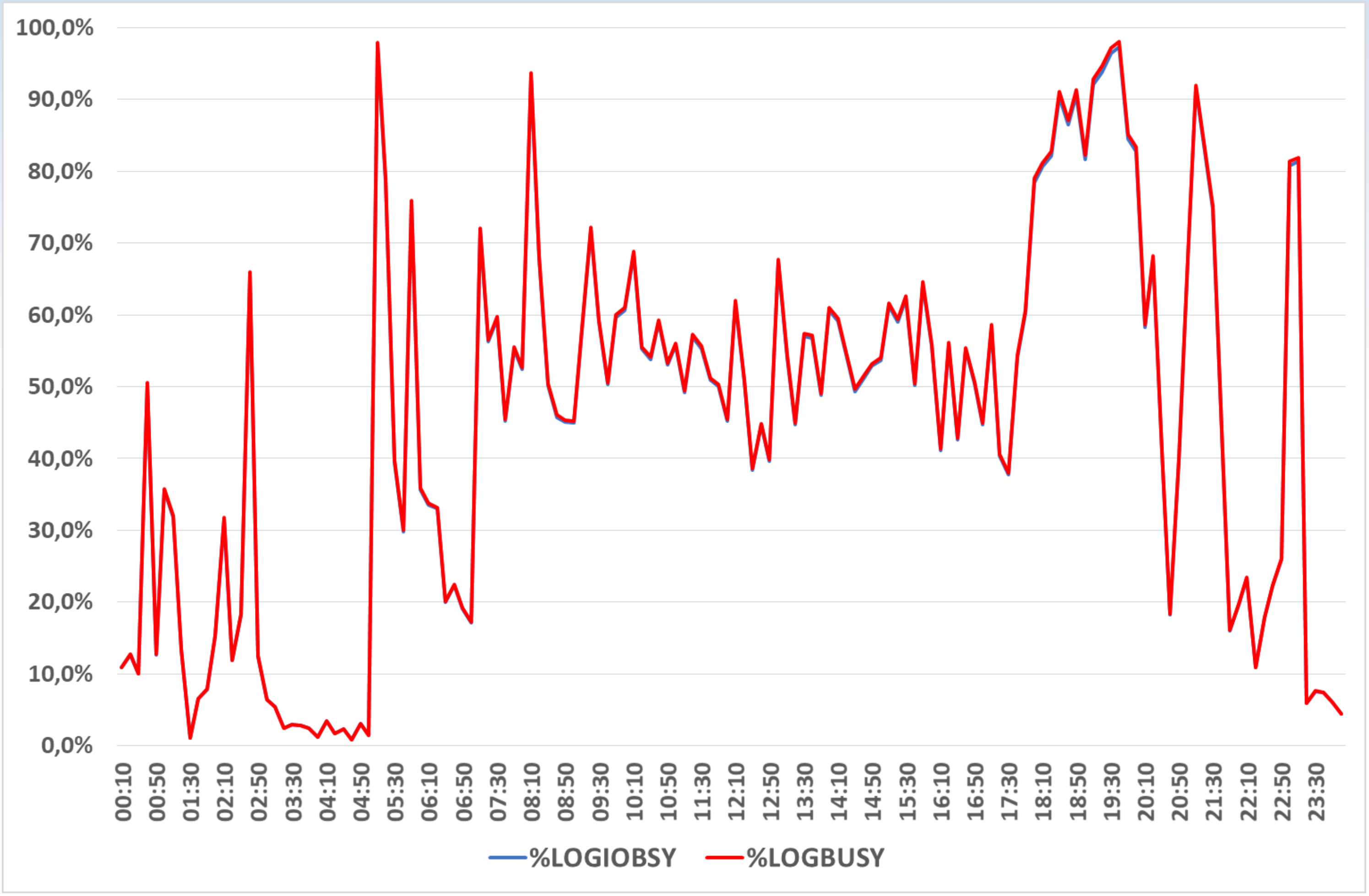
SMF records – MQ

- Setting SMFSTAT to YES will cause the activation of the Class 1 statistics trace and the production of two SMF 115 records:
  - ✓ one subtype 1 record, including log manager and storage manager statistics
  - ✓ one subtype 2 record, including message manager, data manager, buffer manager, lock manage, Db2 manager and CF manager statistics
- Setting STATIME to zero will align statistics record production to the SMF global accounting interval

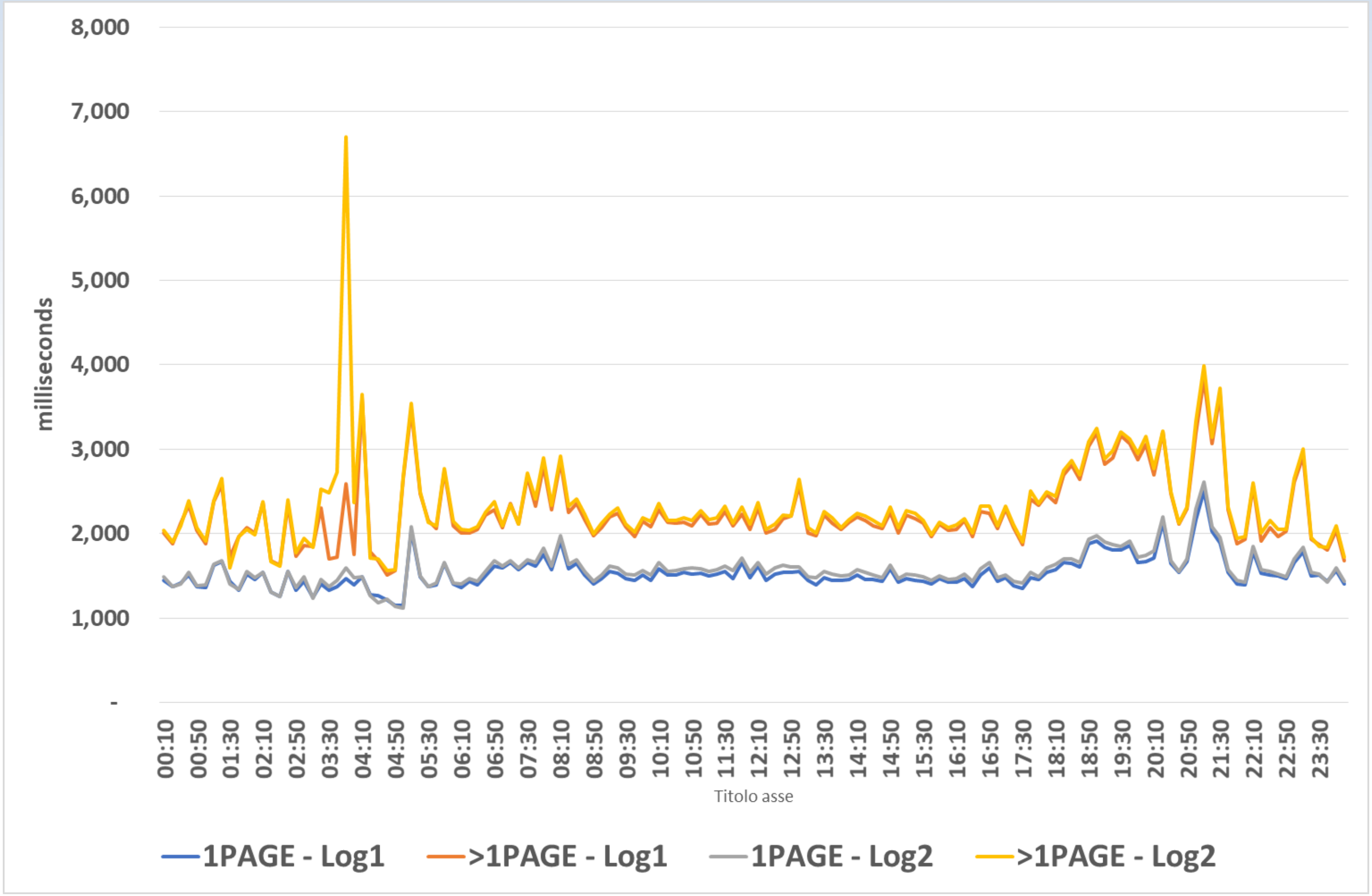
SMF records – MQ

- With MQ 7.1 statistics trace class 3 provides information about MQ memory used by MSTR address space in SMF 115 subtype 7 ; already available in EPV for MQ V13
- With MQ 8 and 9 new statistics provided in SMF 115:
  - extended buffer pools in subtype 215; they are available in EPV for MQ V13
  - log task activity in subtype 1; available in EPV for MQ V14
  - channel initiators in subtype 231; available in EPV for MQ V14
  - page sets in subtype 201; available in EPV for MQ V14

SMF records – MQ

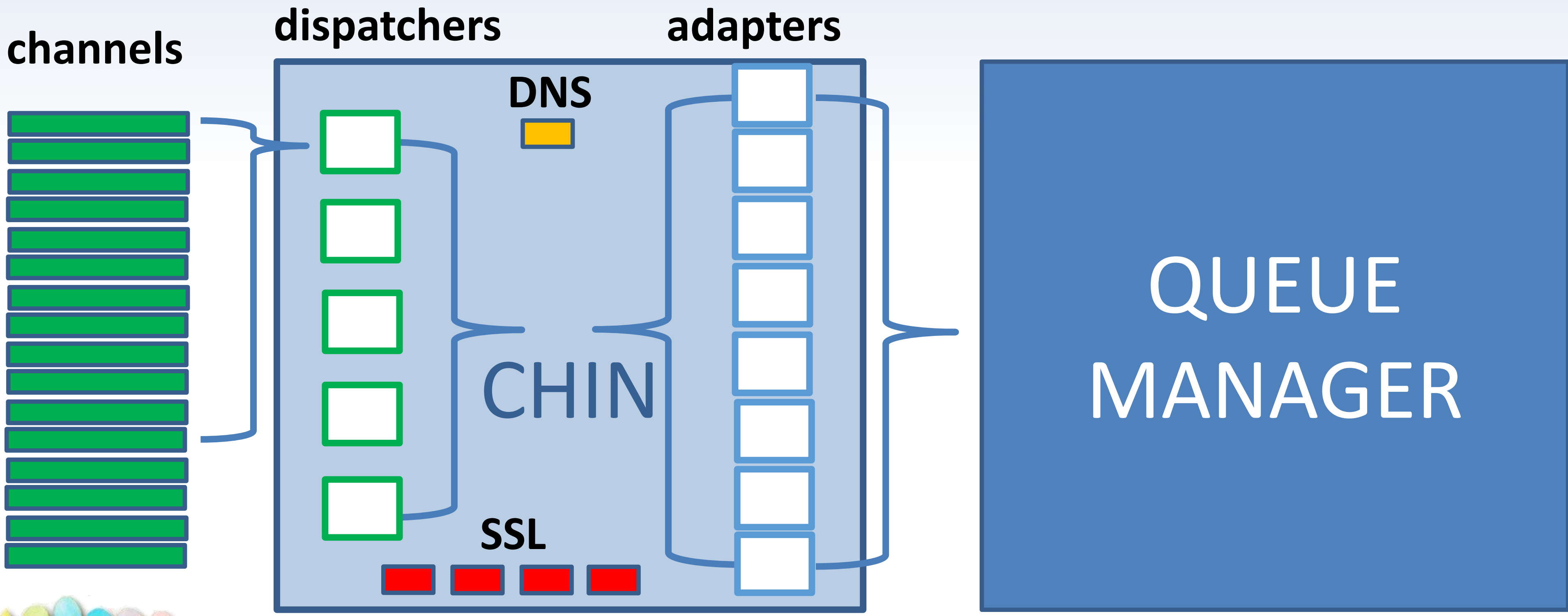


SMF records – MQ



SMF records – MQ

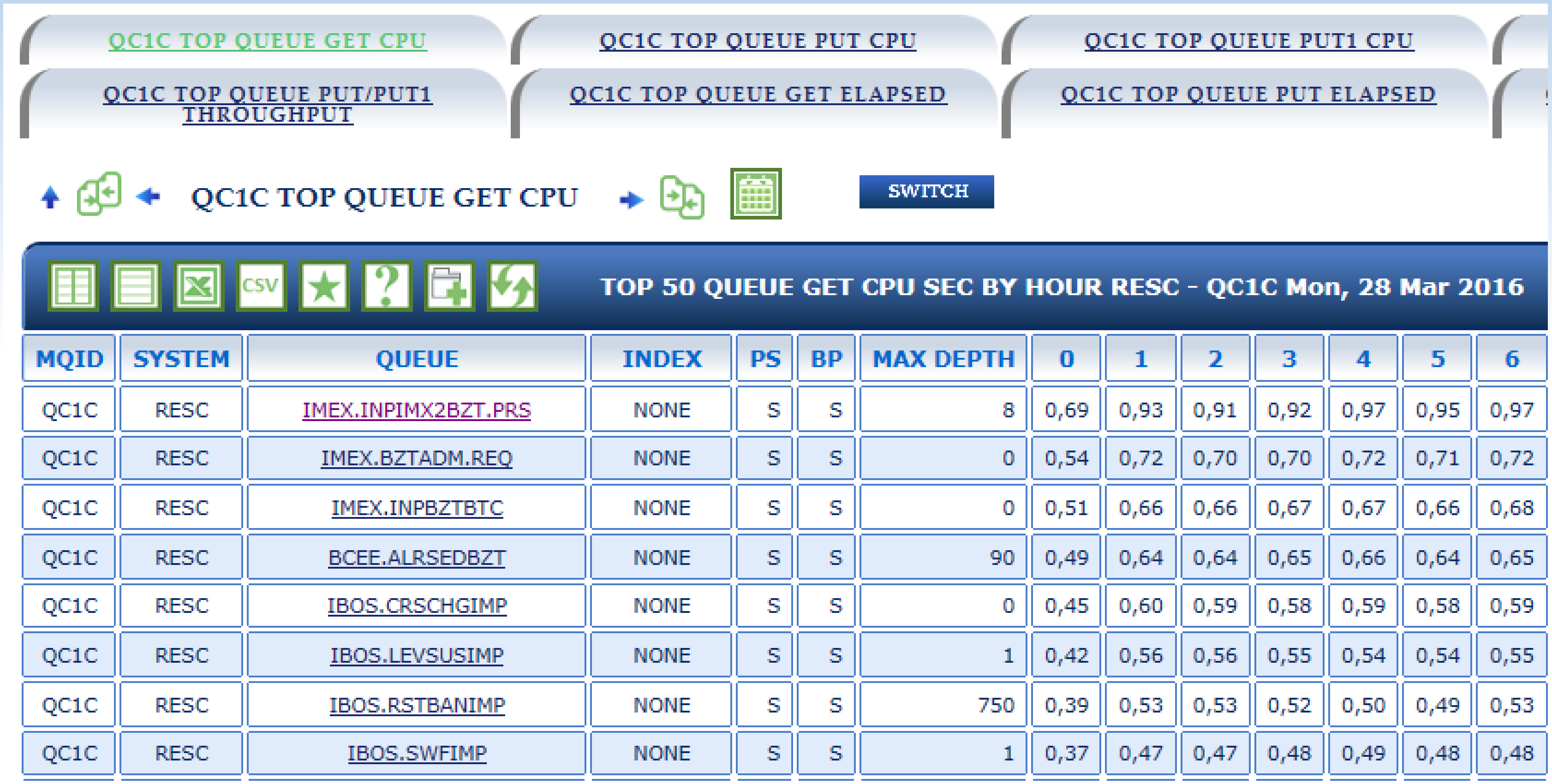
- Lot of information from inside the channel initiator AS



SMF records – MQ

- Setting SMFACCT to YES will activate the class 1 accounting trace causing the production of one SMF 116 subtype 0 record at every thread termination; used in EPV for z/OS
- More detailed information about threads and queues can be collected by activating the class 3 accounting trace causing the production of one subtype 1 and one or more subtype 2 records at thread termination and, for long running threads, at the time interval specified in STATIME; used in EPV for MQ

SMF records – MQ



QC1C TOP QUEUE GET CPU | QC1C TOP QUEUE PUT CPU | QC1C TOP QUEUE PUT1 CPU
QC1C TOP QUEUE PUT/PUT1 THROUGHPUT | QC1C TOP QUEUE GET ELAPSED | QC1C TOP QUEUE PUT ELAPSED

QC1C TOP QUEUE GET CPU

SWITCH

TOP 50 QUEUE GET CPU SEC BY HOUR RESC - QC1C Mon, 28 Mar 2016

MQID	SYSTEM	QUEUE	INDEX	PS	BP	MAX DEPTH	0	1	2	3	4	5	6
QC1C	RESC	IMEX.INPIMX2BZT.PRS	NONE	S	S	8	0,69	0,93	0,91	0,92	0,97	0,95	0,97
QC1C	RESC	IMEX.BZTADM.REQ	NONE	S	S	0	0,54	0,72	0,70	0,70	0,72	0,71	0,72
QC1C	RESC	IMEX.INPBZTBTC	NONE	S	S	0	0,51	0,66	0,66	0,67	0,67	0,66	0,68
QC1C	RESC	BCEE.ALRSEDBZT	NONE	S	S	90	0,49	0,64	0,64	0,65	0,66	0,64	0,65
QC1C	RESC	IBOS.CRSCHGIMP	NONE	S	S	0	0,45	0,60	0,59	0,58	0,59	0,58	0,59
QC1C	RESC	IBOS.LEVSUSIMP	NONE	S	S	1	0,42	0,56	0,56	0,55	0,54	0,54	0,55
QC1C	RESC	IBOS.RSTBANIMP	NONE	S	S	750	0,39	0,53	0,53	0,52	0,50	0,49	0,53
QC1C	RESC	IBOS.SWFIMP	NONE	S	S	1	0,37	0,47	0,47	0,48	0,49	0,48	0,48

SMF records – MQ

- With MQ 8 accounting information about channel initiator will be available in SMF 116 subtype 10; they will be available in EPV for MQ V14
- EPV for MQ V14 MA is expected by the end of March

SMF records – WebSphere

- WAS performance information provided in SMF 120
- EJB and WEB application measurements aggregated by object and interval in SMF 120 subtype 6 and 8
- Low overhead and few records
- They are used in EPV for z/OS Throughput views

SMF records – WebSphere

- Record type 120 provides the following subtypes:
  - ✓ Subtype 1: Server activity record,
  - ✓ Subtype 3: Server interval record
  - ✓ Subtype 5: J2EE container activity record
  - ✓ Subtype 6: J2EE container interval record
  - ✓ Subtype 7: WebContainer activity record
  - ✓ Subtype 8: WebContainer interval record
  - ✓ Subtype 9: Request Activity record
  - ✓ Subtype 10: Outbound Request record
  - ✓ Subtype 11: Liberty activity

SMF records – WebSphere

- Go to the Environment entries page of the administrative console
- click **Servers > Server Types > WebSphere application servers > server_name > Java and Process Management > Process definition > Environment entries**
- To enable SMF type 120 records, click New, and specify one or more of the following properties

SMF records – WebSphere

- To produce SMF 120 from subtype 1 to 8 set:
  - ✓ name = server_SMF_interval_length, value=n, where n is the interval, in seconds; set this value to 0 to use the default SMF recording interval
  - ✓ name = server_SMF_server_activity_enabled = 1 (subtype 1)
  - ✓ name = server_SMF_server_interval_enabled = 1 (subtype 3)
  - ✓ name = server_SMF_container_activity_enabled = 1 (subtype 5 and 7)
  - ✓ name = server_SMF_container_interval_enabled = 1 (subtype 6 and 8)

WEBSPHERE records – SMF 120

- To produce SMF 120 subtype 9 set:
  - ✓ name = server_SMF_request_activity_enabled = 1
  - ✓ name = server_SMF_request_activity_CPU_detail = 1
  - ✓ name = server_SMF_request_activity_timestamps = 1
  - ✓ name = server_SMF_request_activity_security = 1
  - ✓ name = server_SMF_request_activity_async = 1
- To produce SMF 120 subtype 10 set:
  - ✓ name = server_SMF_outbound_enabled = 1
- To produce SMF 120 subtype 11 see:

  WebSphere Liberty: Understanding the SMF 120 Subtype 11 Record

SMF records – WebSphere

- SMF 120 subtype 6 and 8 used in EPV for z/OS
- Main issue: CPU time includes zIIP/zAAP time, no asynch beans information
- Possible solution is SMF 120 subtype 9
- We investigated the possibility to use it instead of subtype 6 and 8

SMF records – WebSphere

- WebSphere creates one subtype 9 record for every request that the server processes — for both external requests (application requests) and internal requests, such as when the controller "talks to" the servant regions.

from ‘Understanding SMF Record Type 120, Subtypes 9 and 10’ - WP101342

WEBSPHERE records – SMF 120

- Activating SMF 120 subtype 9 will greatly increase the number of SMF records produced compared to using subtype 6 and 8
- In our test with a 2 minutes SMF interval we got about 500 subtype 6 and 8
- The number of subtype 9 records exceeded 100.000

SMF records – WebSphere

- Activating SMF 120 subtype 9 may impact WAS performance
- We turned it on during a performance test and WebSphere was so slow that we should turn it off after few minutes
- In that test we used all the collection options
- We plan to check it better but the feeling is that subtype 9 should not be used in production or when performing extensive tests

SMF records – WebSphere

- If you don’t activate the CPU Usage Breakdown option you don’t get info at the object level (servlet, EJB method, etc)
- If you activate the CPU Usage Breakdown option you get info only for the first 30 objects involved (e.g. if a servlet calls 35 different EJBs, information about only the first 29 EJBs, plus 1 servlet is provided)
- Split of CPU and zAAP/zIIP time is not provided in the CPU Usage Breakdown section

SMF records – WebSphere

- We tried to match EJB measurements in subtype 6 and 9
- We only focused on number of method executions and CPU time
- We found many numbers not matching
- We've investigated it again when we started EPV for z/OS V14 development but we found the same issues

	SMF 120-6		SMF 120-9	
AMCName (Application::Module::Class)	**#**	**Total CPU**	**#**	**Total CPU**
IDM::1stWfmEjbBuilder.jar::ProcErrorHandlerBean	3532	0,08	7.064	0,33
IDM::1stWfmEjbBuilder.jar::ServiceCall	571	0,06	1.142	5,56
IDM::depEjbBuilder.jar::DEPDataHandler	666	1,05	666	1,09
IDM::depEjbBuilder.jar::DEPFaultHandler	666	0,09	666	0,04
IDM::depEjbBuilder.jar::SWIIncomingFileSNF01MDB	26	12,78	26	12,46
IDM::depEjbBuilder.jar::SWIIncomingMessageTechnicalAckSNF01MDB	315	27,97	315	27,56
IDM::depEjbBuilder.jar::SWIIncomingMessageTechnicalAckSNF02MDB	325	29,46	325	29,03
IDM::MMCommonServicesEjbBuilder.jar::J1.INFOCENTER.QUEUE	3467	116,43	3.467	115,20
IDM::MMDriversFromApplEjbBuilder.jar::J1.MESSAGE03.FROM.BOA.QUEUE	79	12,48	79	12,43
IDM::runtimeYavaEjb.jar::CounterBean	15	0,03	15	0,03
IDM_WEB::webgenServiceEntryPoint-ejb-1.0.0.jar::ServiceEntryPoint	585	12,26	1.328	22,43
IDM_WEB::webgenServiceEntryPoint-ejb-1.0.0.jar::SessionDerivedData	762	0,15	4.512	0,42
	11.009	212,85	19.605	226,59

SMF records - CICS

- CICS can produce the following SMF records:
  - ✓ SMF 110 subtype 1; monitoring records (used by EPV for z/OS)
  - ✓ SMF 110 subtype 2; statistics records
  - ✓ SMF 110 subtype 3; shared temporary storage queue server statistics
  - ✓ SMF 110 subtype 4; coupling facility data table server statistics
  - ✓ SMF 110 subtype 5; named counter sequence number server statistics
  - ✓ SMF 111 subtype 1 for CICS Transaction Gateway statistics

SMF records - CICS

- CICS can collect four types, or classes, of monitoring data:
  - ✓ Performance class data; detailed transaction-level information, such as the processor and elapsed time details for a transaction; at least one performance monitoring record for each transaction (used by EPV for z/OS)
  - ✓ Exception class monitoring data; information on CICS resource shortages suffered by a transaction, such as queuing for file strings, or waiting for temporary storage; one exception record for each exception condition occurred

SMF records - CICS

- ✓ Transaction resource class data; additional transaction-level information about individual resources accessed by a transaction
- ✓ Identity class data; enhanced audit information by capturing identity propagation data from a client system across a network for eligible transactions

SMF records - CICS

- You can enable performance class monitoring by coding MNPER=ON (together with MN=ON) as a system initialization parameter
- Alternatively, you can use the monitoring facility transaction CEMN, or the EXEC CICS SET MONITOR command, to enable performance class monitoring dynamically

SMF records - CICS

- Most of CICS users customize the MCT
- So the record layout of SMF 110 is extremely variable
- To allow to use SMF 110 information each CICS region at start up write what is called “dictionary record”
- A dictionary record holds definitional information about each data field in a performance class data record

SMF records - CICS

- A new dictionary record, which always precedes the monitoring performance class data it relates to, is written whenever the user:
  - ✓ Starts CICS with the performance class active, and CICS monitoring on
  - ✓ Changes the status of the monitoring performance class from inactive to active, with CICS monitoring on; if monitoring is off and the monitoring performance class is switched from inactive to active, a dictionary record is scheduled to be written the next time monitoring is activated

SMF records - CICS

- If SMF switches data sets during the period when CICS monitoring is writing performance class data, CICS does not write a new dictionary record
- So it could not be possible to analyse SMF 110 records collected in that data set
- To overcome this issue you can use the CICS-supplied monitoring dictionary utility program, DFHMNDUP, and write dictionary records to a sequential data set
- Then you can read it before the SMF data set

SMF records - CICS

```
//STEP1 EXEC PGM=DFHMNDUP
//STEPLIB   DD DSN=CICS.XXX.SDFHLOAD,DISP=SHR
//          DD DSN=CICS.XXX.CAT.LOADLIB,DISP=SHR
//SYSUT4    DD DSN=YOUR.OUTPUT.DICT,UNIT=SYSDA,
//          SPACE=(CYL,(1,1)),DISP=(,CATLG)
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 MCT=MCT suffix
 SYSID=sysid
 GAPPLID=generic applid
 SAPPLID=specific applid
/*
```

SMF records - CICS

- CICS transaction CPU time always included time spent on zIIP/zAAP
- zIIP/zAAP time NORMALIZED to the CPU speed
- Starting from CICS TS 5.1 new metrics are available to separate CPU and zAAP/zIIP time
- Detailed information about CICS transaction is provided in SMF 110 subtype 1

SMF records - CICS

- USRCPUT: total TCB time
- RLSCPUT: SRB CPU time spent processing RLS file requests

- CPUTONCP: total CPU TCB time
- OFFLCPUT: CPU TCB time eligible for offload to zIIP/zAAP
- Normalized (CPU speed) zIIP/zAAP TCB time

USRCPUT – CPUTONCP

SMF records - CICS

- USRCPUT can exceed elapsed time when using knee-capped processors
- To avoid that and correctly decompose CICS transactions elapsed time you have to de-normalized zIIP/zAAP TCB time:

  *(USRCPUT – CPUTONCP) * 256 / R723NFFS*

- R723NFFS is the normalization factor for zIIP available in SMF 72

SMF records - CICS

- Exercise 2

CICS transaction running on a knee capped processor; zIIP speed 2 times CPU (R723NFFS=512)

- Elapsed time is 1 sec
- USRCPUT=1,4 sec
- CPUTONCP=0,2 sec

Calculate:

- de-normalized zIIP time
- not CPU not zIIP time

SMF records - CICS

- Exercise 2

CICS transaction running on a knee capped processor; zIIP speed 2 times CPU (R723NFFS=512)

- Elapsed time is 1 sec
- USRCPUT=1,4 sec
- CPUTONCP=0,2 sec

Calculate:

- de-normalized zIIP time = (1, 4 sec – 0,2 sec) * 256 / 512 = 0,6 sec
- not CPU not zIIP time = 1 sec – 0,2 sec – 0,6 sec = 0,2 sec

IMS log records

IMS log records - Overview

- IMS doesn’t write any SMF record
- All the relevant events are mapped to a specific log record number and written to the IMS log
- For many years BMC Mainview for IMS produced “Transaction log record”, identified by the log record number x’FA’ – BMC FA in the following
- Since IMS V10, IBM finally decided to provide an IMS log record specifically designed to collect performance information: the x’56FA’ log record (56FA in the following)

IMS log records – BMC FA records

- A specific Mainview for IMS component is responsible for data collection and BMC FA recording: the Event Collector; you can customise it by setting data collection parameters
- Discussion of these parameters is beyond the scope of this presentation so we will only focus on two of them :
- BMP = (YES | NO | NOCPU);
- CPU = (DEP | DEPPGM | DEPDb2 | ALL | NONE)

IMS log records – BMC FA records

- If you want to collect BMP (and JBP) transaction and program activity data and the corresponding CPU consumption you have to specify BMP=YES
- If you want to know how much of the transaction CPU has been used in Db2 then you have to choose CPU=DEP, DEPDb2 or ALL

IMS log records – IBM 56FA records

- You can enable or disable 56FA logging globally during system definition by specifying a new parameter, TRANSTAT (Y/N, where N is the default), in the Diagnostics Statistics section of the new DFSDFxxx PROCLIB member
- You can enable or disable 56FA logging on a program basis for non-message driven applications, and on a tran-by-tran basis for message-driven applications by using IMS UPDATE and CREATE commands

IMS log records – Selection

- Whatever input you choose to select only the needed records from the IMS log you can use the DFSERA10 utility

IMS log records – Selection

```
//SELIMS   EXEC PGM=DFSERA10
//STEPLIB  DD DSN=RESLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DSN=SLDS,
//         DISP=(OLD,PASS),UNIT=TAPE
//SYSUT4   DD DSN=OUTPUT_FILE,DISP=(,CATLG,DELETE),
//         UNIT=SYSDA,
//         SPACE=(CYL,(100,100),RLSE),
//         DCB=(RECFM=VB,LRECL=30970,BLKSIZE=30974)
//SYSIN    DD *
CONTROL  CNTL STOPAFT=EOF
OPTION   COPY  OFFSET=5,FLDLEN=2,VALUE=56FA,COND=E
OPTION   COPY  OFFSET=5,FLDLEN=1,VALUE=FA,COND=E
END
```

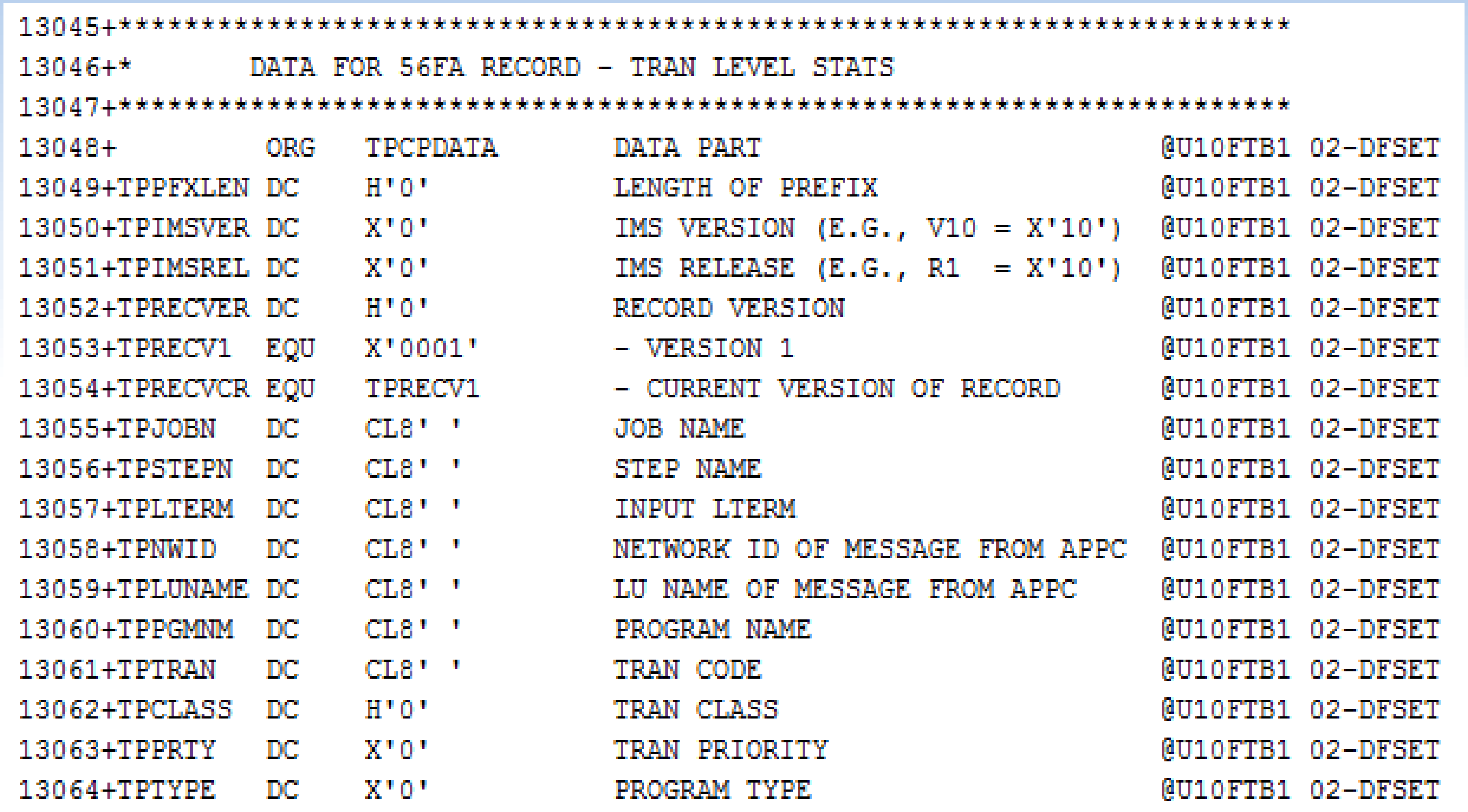
IMS log records - Layout

- BMC FA record layout is described in the Appendix A of “Mainview for IMS Offline – Customization and Utilities Guide”
- BMC FA record DSECTs are also provided in the IMETRN and IMETRNX members of the BBSAMP library
- All the IBM log records layout, including 56FA, can be obtained by assembling the ILOGREC macro (see JCL example in the next slide)

IMS log records - Layout

```
//DSECT    PROC
//HASM     EXEC
PGM=IEV90,PARM='NODECK,NOXREF,LIST,NORLD,NOOBJECT'
//SYSLIB   DD  DISP=SHR,DSN=IMS_prefix.SDFSMAC
//SYSUT1   DD  UNIT=SYSDA,SPACE=(CYL,(2,1))
//SYSUT2   DD  UNIT=SYSDA,SPACE=(CYL,(2,1))
//SYSUT3   DD  UNIT=SYSDA,SPACE=(CYL,(2,1))
//SYSPRINT DD  SYSOUT=*
//         PEND
//D1       EXEC DSECT
//SYSIN    DD  *
 ILOGREC  RECID=56
         END
```

IMS log records - Layout



```
13045+************************************************************************
13046+*       DATA FOR 56FA RECORD - TRAN LEVEL STATS
13047+************************************************************************
13048+         ORG   TPCPDATA       DATA PART                         @U10FTB1 02-DFSET
13049+TPPFXLEN DC    H'0'           LENGTH OF PREFIX                  @U10FTB1 02-DFSET
13050+TPIMSVER DC    X'0'           IMS VERSION (E.G., V10 = X'10')   @U10FTB1 02-DFSET
13051+TPIMSREL DC    X'0'           IMS RELEASE (E.G., R1  = X'10')   @U10FTB1 02-DFSET
13052+TPRECVER DC    H'0'           RECORD VERSION                    @U10FTB1 02-DFSET
13053+TPRECV1  EQU   X'0001'        - VERSION 1                       @U10FTB1 02-DFSET
13054+TPRECVCR EQU   TPRECV1        - CURRENT VERSION OF RECORD       @U10FTB1 02-DFSET
13055+TPJOBN   DC    CL8' '         JOB NAME                          @U10FTB1 02-DFSET
13056+TPSTEPN  DC    CL8' '         STEP NAME                         @U10FTB1 02-DFSET
13057+TPLTERM  DC    CL8' '         INPUT LTERM                       @U10FTB1 02-DFSET
13058+TPNWID   DC    CL8' '         NETWORK ID OF MESSAGE FROM APPC   @U10FTB1 02-DFSET
13059+TPLUNAME DC    CL8' '         LU NAME OF MESSAGE FROM APPC      @U10FTB1 02-DFSET
13060+TPPGMNM  DC    CL8' '         PROGRAM NAME                      @U10FTB1 02-DFSET
13061+TPTRAN   DC    CL8' '         TRAN CODE                         @U10FTB1 02-DFSET
13062+TPCLASS  DC    H'0'           TRAN CLASS                        @U10FTB1 02-DFSET
13063+TPPRTY   DC    X'0'           TRAN PRIORITY                     @U10FTB1 02-DFSET
13064+TPTYPE   DC    X'0'           PROGRAM TYPE                      @U10FTB1 02-DFSET
```

IMS log records – Old IBM records

- Many customers still use IMS log records 1, 7, 8
- Their processing is more complex and resource consuming because you have to merge these records
- Not all the needed information are provided because these records have not been designed for performance analysis
- Most important missing fields:
  - ✓ IMSID
  - ✓ IMS version
  - ✓ Transaction input queue time

IMS log records – New CPU Metrics

- In BMC FA, total CPU time, including zAAP/zIIP time, is provided in the TRXZTCPU field while the standard CPU time is provided in the TRXZONCP
- To get the zAAP/zIIP time:

  zAAP/zIIP time = TRXZTCPU - TRXZONCP
- As in CICS the zAAP/zIIP time is normalized to the CPU speed
- zAAP and zIIP eligible time on standard CPU is provided in TRXZAOCP and TRXZIOCP
- CPU time in Db2 is provided (DEP, DEPDb2, ALL)

IMS log records – New CPU Metrics

- In 56FA, from IMS V12, zAAP/zIIP time is provided in the TPEZAAP field while the standard CPU time is collected in TPEXTIME
- As in CICS the zAAP/zIIP time is normalized to the CPU speed
- zAAP and zIIP eligible time on standard CPU is not provided in 56FA records
- CPU time in Db2 still missing in 56FA (you need to look at SMF 101)

DCOLLECT records

DCOLLECT records - Overview

- The DFSMS Data Collection Facility (DCOLLECT) is a function of Access Method Services
- DCOLLECT may provides a lot of information about storage usage in many different record types
- The most commonly used are:
  - Type V, for Volume information (used by EPV for z/OS)
  - Type D, for Data set information

DCOLLECT records - Overview

- This information can be used for disk space tuning, capacity planning and cost accounting
- The DCOLLECT record layout is provided in the Appendix of the *DFSMS Access Method Services – Commands* manual

DCOLLECT records – Volume info

- Volume info can be collected with this simple JCL

```
//SELDCOL  EXEC  PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//DCOUT    DD DSN=dcolpref.sysid,DISP=(,CATLG,DELETE),
//         UNIT=SYSDA, SPACE=(CYL,(10,1),RLSE),
//         DCB=(RECFM=VB,LRECL=644,BLKSIZE=0)
//SYSIN    DD    *
    DCOLLECT -
    OUTFILE(DCOUT) -
    VOLUMES( * ) -
    NODATAINFO
/*
```

DCOLLECT records – Dataset info

- Dataset info can be collected with this simple JCL

```
//SELDCOL  EXEC  PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//DCOUT    DD DSN=dcolpref.sysid,DISP=(,CATLG,DELETE),
//         UNIT=SYSDA, SPACE=(CYL,(10,1),RLSE),
//         DCB=(RECFM=VB,LRECL=644,BLKSIZE=0)
//SYSIN    DD    *
    DCOLLECT -
    OUTFILE(DCOUT) -
    VOLUMES( * )
/*
```

VTS BVIR records

VTS BVIR records - Overview

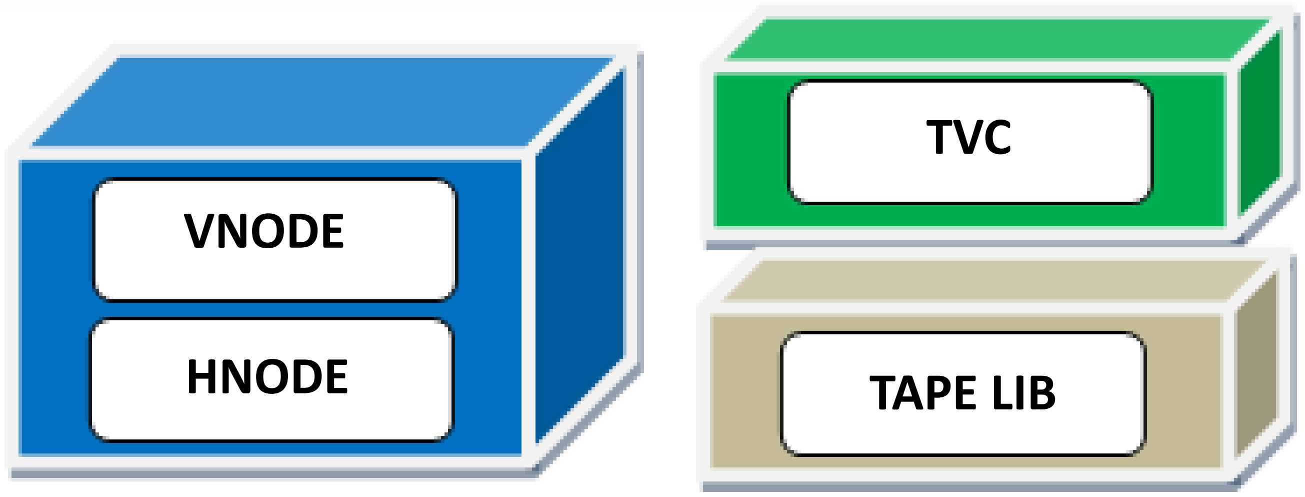
- The IBM Virtualization Engine TS7700, also named Hydra, is the current generation of tape virtualization solution for mainframe systems
- The new architecture has been completely redesigned to make it more modular and scalable and to allow easy implementation of advanced Disaster Recovery and Business Continuity solutions
- We will call it simply VTS in the following

VTS BVIR records - Overview

- VTS is built on a distributed node architecture; the virtualization engine includes:
  - ✓ a VNODE; it is the virtualization node and has to present the image of virtual drives to host systems; it receives tape mounts requests, translates them to virtual tape drive requests and uses files in a disk subsystem , the so called Tape Volume Cache (TVC), to represent the logical tape volume image
  - ✓ a HNODE; it is the hierarchical data storage management node and has to perform all management of a logical tape volume residing in the TVC or in a physical tape after it has been created or altered by the host system through a VNODE; it is also responsible for any replication of the logical volumes and their attributes across site boundaries

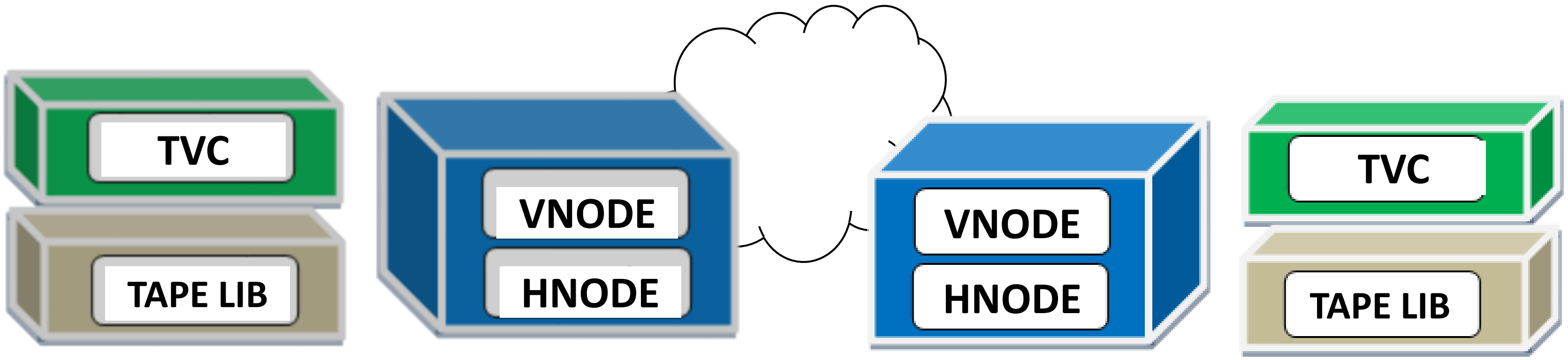
VTS BVIR records - Overview

- A Cluster is the combination of the Virtualization Engine with a disk subsystem providing the TVC and (optionally) that part of a Tape Library which is assigned to the VTS



VTS BVIR records - Overview

- A Cluster always belongs to a grid; to identify a cluster (a VTS) you need:
  - ✓ the Grid Library Sequence Number, 5 characters
  - ✓ the Cluster ID, 1 byte hexadecimal value to identify clusters inside a grid (from x'00' to x'07')



VTS BVIR records - Statistics

- The VTS statistics production system has been also redesigned to provide a lot of very useful metrics
- The bad news is that no SMF records are written natively; all the statistics are collected inside the machine
- These statistics are available to users via the Bulk Volume Information Retrieval (BVIR) facility

VTS BVIR records - Statistics

- Two types of statistics are provided:
  - ✓ Point-in-time statistics to understand what’s happening in the VTS at this moment; data provided by this type of statistic is a snapshot of the activity over the last 15-second interval; each new 15-second interval data overlays the previous interval’s data
  - ✓ Historical statistics to understand how you are using VTS resources; the data provided by this type of statistic is captured over a 15-minute interval; 90 rolling days of historical statistics are kept in the VTS subsystem database (used by EPV for z/OS)

VTS BVIR records - Statistics

Data Type	Description	Number of Records	EPV for z/OS
x20	Vnode Virtual Device Historical Record	1 per Vnode	YES
x21	Vnode Adapter Historical Record	1 per Vnode	YES
x30	Hnode HSM Historical Record	1 per Hnode	YES
x31	Hnode Export/Import Historical Record	1 per Hnode	NO
x32	Hnode Library Historical Record	1 per library/cluster	YES
x33	Hnode Grid Historical Record	1 per Hnode	YES

VTS BVIR records – Producing statistics

- To get statistical records from BVIR you may use a two step JCL running the standard IEBGENER utility
- The result is a file in Undefined format containing binary data which has to be interpreted by using the appropriate record formats (see IBM Virtualization Engine TS7700 Series - Statistical Data Format - White Paper)

VTS BVIR records – Producing statistics

- In the first step a single data set with the information request is written to a logical volume
- The logical volume can be any logical volume in the subsystem the information is to be obtained from
- The data set contains a minimum of two records and a maximum of three records that specifies the type of data being requested
- The records are in human readable form, i.e. lines of character data
- On close of the volume, the virtualization engine server will recognize it as a request volume and 'prime' the subsystem for the next step

VTS BVIR records – Producing statistics

```
//VTSREQ   EXEC PGM=IEBGENER
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  DUMMY
//SYSUT2   DD  DSN=userid.VTSREQ,
//         LABEL=(1,SL),DISP=(,CATLG),UNIT=vtsunit,
//         DCB=(RECFM=F,BLKSIZE=80,LRECL=80,TRTCH=NOCOMP)
//SYSUT1   DD   *
VTS BULK VOLUME DATA REQUEST
HISTORICAL STATISTICS FOR 334-334
```

VTS BVIR records – Producing statistics

- In the second step, the request volume is again mounted, this time as a specific mount
- Seeing that the volume was ‘primed’ for a data request, the virtualization engine appends the requested information to the data set
- Once the VTS has completed appending to the data set, the host is notified that the mount has completed
- The requested data can then be accessed like any other tape data set

VTS BVIR records – Producing statistics

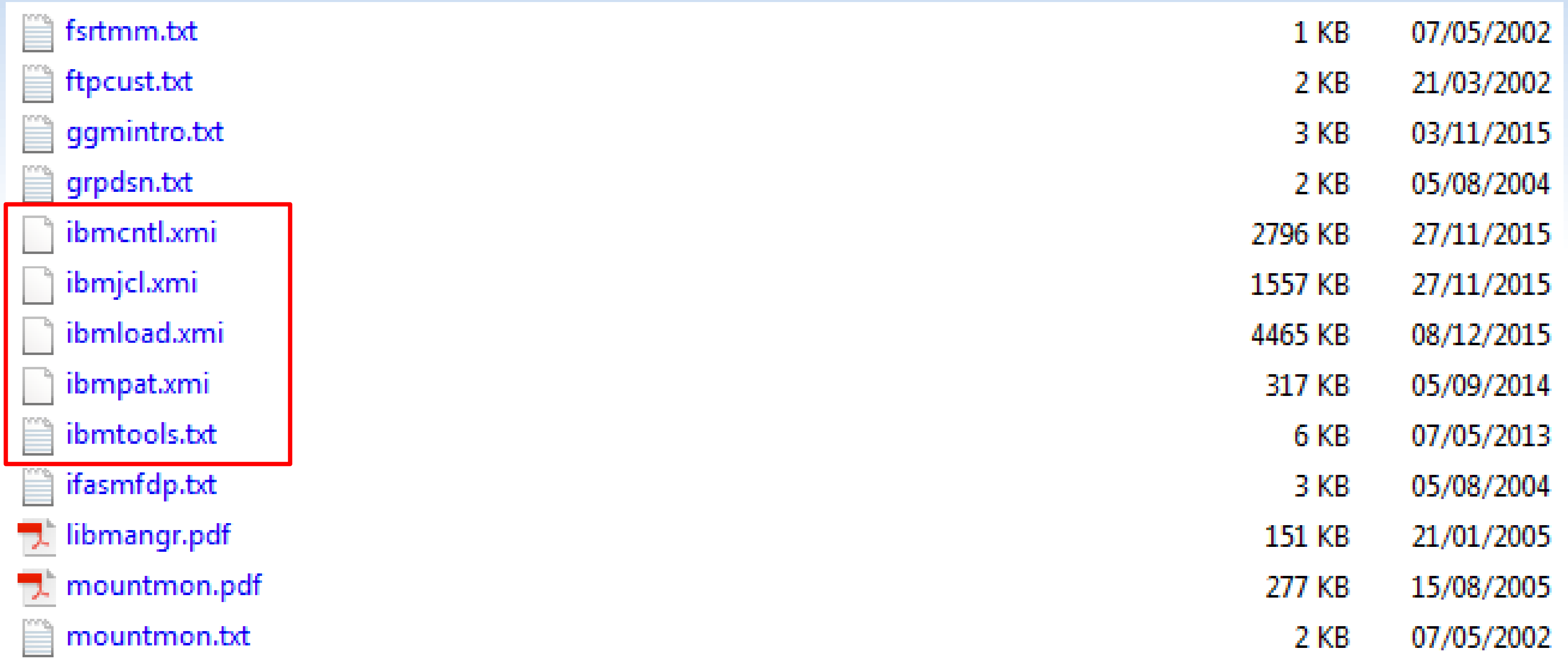
```
//VTSDATA   EXEC PGM=IEBGENER
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD DSN=userid.VTSREQ,DISP=SHR,
//         DCB=(RECFM=U,BLKSIZE=24000)
//SYSUT2   DD DSN=userid.VTSDATA,
//         SPACE=(CYL,(10,5)),DISP=(,CATLG),
//         UNIT=SYSDA,DCB=(RECFM=U,BLKSIZE=24000)
//SYSIN    DD DUMMY
```

VTS BVIR records – Producing statistics

- Many customers disagree with the IBM decision to not provide VTS statistics in SMF because it requires ad hoc processing separated from the standard SMF flow
- So IBM decided to provide additional IBM tools which allow for the writing of VTS Historical statistics records in SMF
- You have to download programs and JCL from the IBM tapetool ftp site

VTS BVIR records – Producing statistics

ftp://ftp.software.ibm.com/storage/tapetool/



fsrtmm.txt	1 KB	07/05/2002
ftpcust.txt	2 KB	21/03/2002
ggmintro.txt	3 KB	03/11/2015
grpdsn.txt	2 KB	05/08/2004
ibmcntl.xmi	2796 KB	27/11/2015
ibmjcl.xmi	1557 KB	27/11/2015
ibmload.xmi	4465 KB	08/12/2015
ibmpat.xmi	317 KB	05/09/2014
ibmtools.txt	6 KB	07/05/2013
ifasmfdp.txt	3 KB	05/08/2004
libmangr.pdf	151 KB	21/01/2005
mountmon.pdf	277 KB	15/08/2005
mountmon.txt	2 KB	07/05/2002

VTS BVIR records – Producing statistics

- Writing VTS Historical statistics records in SMF is not straightforward; two main issues have to be addressed:
  - ✓ the CPYHIST program, writing the VTS SMF records, has to reside in an APF library
  - ✓ you need to download the IBM programs and JCL periodically because they have an expiration date, set by IBM, to ensure that customers don’t use obsolete objects

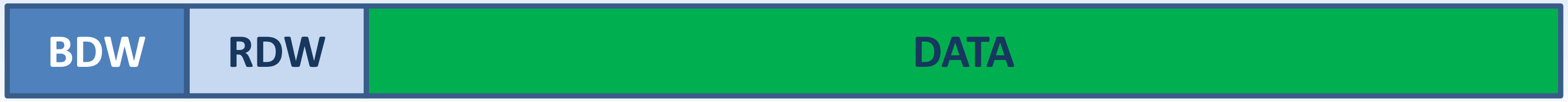
Moving data to another platform

Moving data to another platform

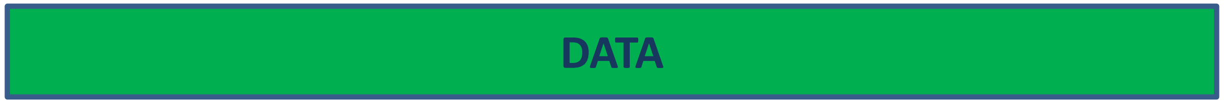
- All V, VB and VBS records have a 4-byte BDW and a 4-byte RDW
- V uses only the BDW, VB uses BDW and RDW, and VBS uses both and the low two-bytes of RDW for the spanning bits)
- Block Descriptor Word (BDW)
  - ✓ The block descriptor word is a 4-byte field that describes the block; it specifies the 4 byte block length for the BDW plus the total length of all records or segments within the block
- Record Descriptor Word (RDW)
  - ✓ The record descriptor word is a 4 byte field describing the record; the first 2 bytes contain the length (LL) of the logical record (including the 4 byte RDW); the length can be from 4 to 32 760

Moving data to another platform

- Tools based on ftp strip away both the BDW and RDW blocks to transfer only the data



- Without these blocks providing information about start and end of physical and logical records the records arriving to the external platform are not readable anymore



Moving data to another platform

- All the input data normally used for z/OS Performance Analysis are in VBS or VB format
- When sending this data to another platform the following actions are needed in order to assure they will be readable:
  - ✓ Protect the block and record headers
  - ✓ Transfer the data in binary mode

Moving data to another platform

- To protect BDW and RDW you can:
  - ✓ Convert the VB or VBS data set to Undefined format; it can be done with a JCL step; it requires to perform a copy of the file
  - ✓ Mask data as Undefined format; it can be done during the ftp or with a JCL step which changes the data set DSCB to RECFM=U without copying it; this second option doesn't work with multi-volume data sets
  - ✓ Compress data; data can be compressed with many expensive tools; the best option is EPVzip, a JAVA tool using only zIIP time free for EPV customers

Moving data to another platform

- When converted or masked as Undefined, all the record, included BDW and RDW, is considered as data

- When compressed, all the record is consider as data

Moving data to another platform

Converting a VBS data set to Undefined

```
//UNDSMF   EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DSN=smfpref.VBS,DISP=SHR,
//         DCB=(RECFM=U,BLKSIZE=32760)
//SYSUT2   DD DSN=smfpref.UND,DISP=(,CATLG),
//         DCB=(RECFM=U,BLKSIZE=32760),
//         UNIT=SYSDA,SPACE=(CYL,(100,100),RLSE)
//SYSIN    DD DUMMY
/*
```

Moving data to another platform

Masking a VBS data set as Undefined

```
//UNDSMF EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSUT2   DD DSN=smfpref.VBS,DISP=MOD,
//            DCB=RECFM=U
//SYSUT1   DD DSN=NULLFILE,DCB=*.SYSUT2
//SYSIN    DD DUMMY
/*
```

Moving data to another platform

Masking a VBS data set as Undefined during ftp

```
//FTPSMF   EXEC PGM=FTP,PARM='(EXIT'
//SYSPRINT DD SYSOUT=*
//OUTPUT   DD SYSOUT=*
//DDSMF    DD DSN=smfpref.VBS,RECFM=U,BLKSIZE=32760,DISP=SHR
//INPUT DD *
your.ftp.host.address
user password
quote PASV
bin
put //DD:DDSMF /smfpref.smf
close
quit
```

Moving data to another platform

Converting BVIR data to VB and then to Undefined

```
//FROMU2VB EXEC PG=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DSN=userid.VTSDATA,DISP=(OLD,PASS)
//SYSUT2   DD DSN=userid.VTSVB,DISP=(,PASS),
//         DCB=(RECFM=VB,LRECL=32756,BLKSIZE=32760),UNIT=SYSDA,SPACE=(CYL,(10,5))
//SYSIN    DD DUMMY
//*
//FINAL   EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DSN=userid.VTSVB,DISP=(OLD,PASS), DCB=(RECFM=U,BLKSIZE=32760)
//SYSUT2   DD DSN=userid.VTSUND,DISP=(,CATLG),
//         DCB=(RECFM=U,BLKSIZE=32760),UNIT=SYSDA,SPACE=(CYL,(10,5))
//SYSIN    DD DUMMY
```

Moving data to another platform

Compressing a VBS data set with EPVzip

```
//EPV001A  JOB MSGCLASS=A,CLASS=A,NOTIFY=&SYSUID
// SET JAVA='/usr/lpp/java/J8.0/bin'
// SET INSTDIR='/u/epv/v2/'
//EPVZIP  EXEC PGM=BPXBATCH,
// PARM=('pgm &JAVA./java -jar &INSTDIR./EPVzip.jar')
//STDENV   DD *
LC_ALL=en_us.IBM-1047
_BPX_SHAREAS=YES
_BPX_BATCH_SPAWN=YES
_BPX_SPAWN_SCRIPT=YES
//CEEDUMP  DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//INPUT    DD DISP=SHR,DSN=EPV001.SMF,DCB=RECFM=U
//OUTPUT   DD DISP=(,CATLG),DSN=EPV001.SMFZIP,
//         DCB=(BLKSIZE=27998,RECFM=U),UNIT=SYSDA,SPACE=(CYL,(500,200))
```

Moving data to another platform

- Whatever method you use to protect record integrity, remember to ftp the data set in binary mode
- Other methods not based on ftp can be used to send data to other platforms:
  - ✓ z/OS SMB
  - ✓ z/OS NFS
  - ✓ Storage processor features (e.g. EMC InfoMover)

zParser data collection architecture

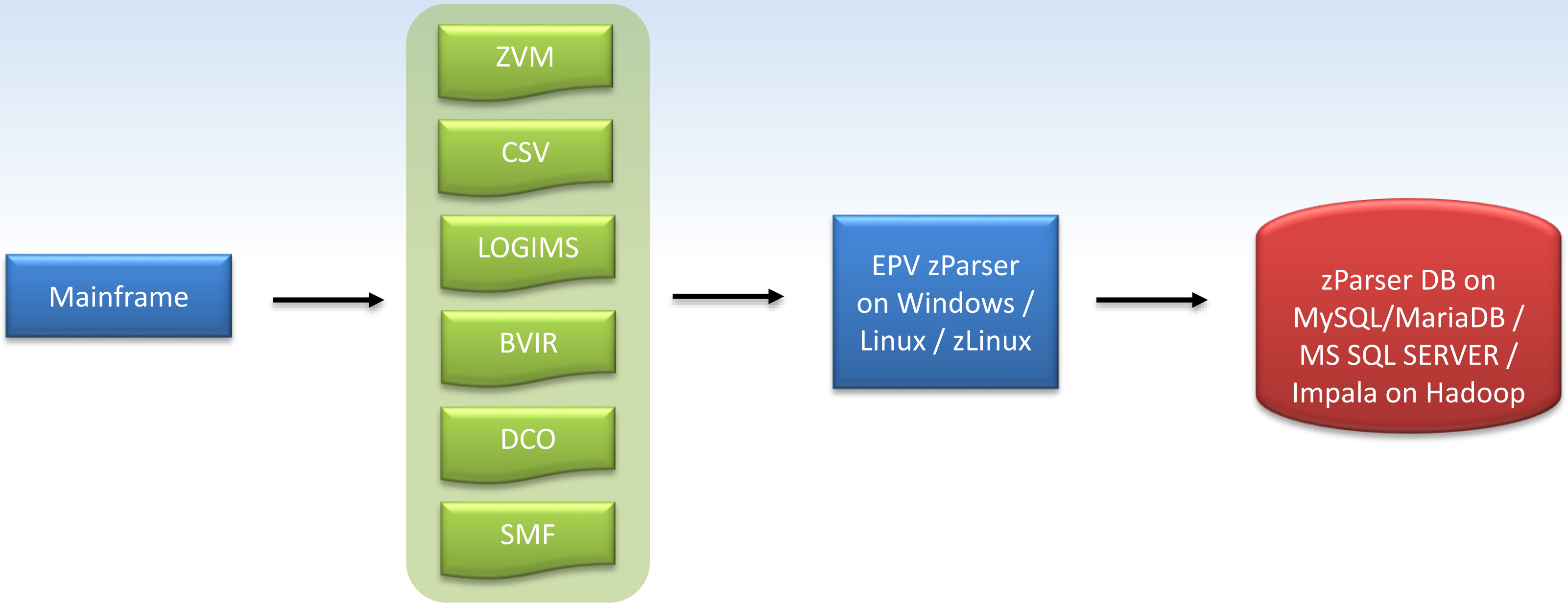
zParser data collection architecture - Overview

- EPV zParser is a revolutionary alternative tool designed to free customers from the need of using old, expensive and complex-to-manage mainframe tools to interpret and collect SMF and other data needed to support Capacity Management and other related activities
- It interprets and collects SMF and RMF standard records, SMF user records, IMS log records, IDCAMS/DCOLLECT records, z/VM monitor records, VTS records produced by the IBM TS7700 VTS, CSV files and creates TXT files which can be loaded in a SQL database

zParser data collection architecture - Overview

- The product architecture is modular and very flexible. It can be installed on most of the hardware and software platforms on the market
- It can be customized in different ways depending on the customer needs

zParser data collection architecture - Overview


Mainframe
ZVM
CSV
LOGIMS
BVIR
DCO
SMF
EPV zParser on Windows / Linux / zLinux
zParser DB on MySQL/MariaDB / MS SQL SERVER / Impala on Hadoop

zParser data collection architecture - Collection modes

- Depending on the customer needs and amount of SMF data produced on a daily basis, zParser can work in two different ways:

  ✓ Once a day

  ✓ Continuous mode

zParser data collection architecture - Collection modes

- Once a day
  - ✓ Send the input files all together (normally just after midnight) or as soon as they are produced during the day
  - ✓ The parsing process, the data aggregation and the HTML production will start only once a day at a specific point in time set in your scheduler
  - ✓ Just one day of raw SMF data is kept in the zParser's DB
  - ✓ This processing mode is ideal for small and medium size customers (e.g. less than 20 GB of input data per day)

zParser data collection architecture - Collection modes

- Continuous mode
  - ✓ Send the input files as soon as they are produced during the day
  - ✓ The parsing process will start as soon as each file arrives
  - ✓ The data aggregation and the HTML production is triggered through the STARTBTC flag file from MF
  - ✓ Multiple days of raw SMF data can be kept in the zParser DBs in a round robin fashion
  - ✓ This processing mode is ideal for large size customers (e.g. more than 20 GB of input data per day).

zParser data collection architecture - Licenses

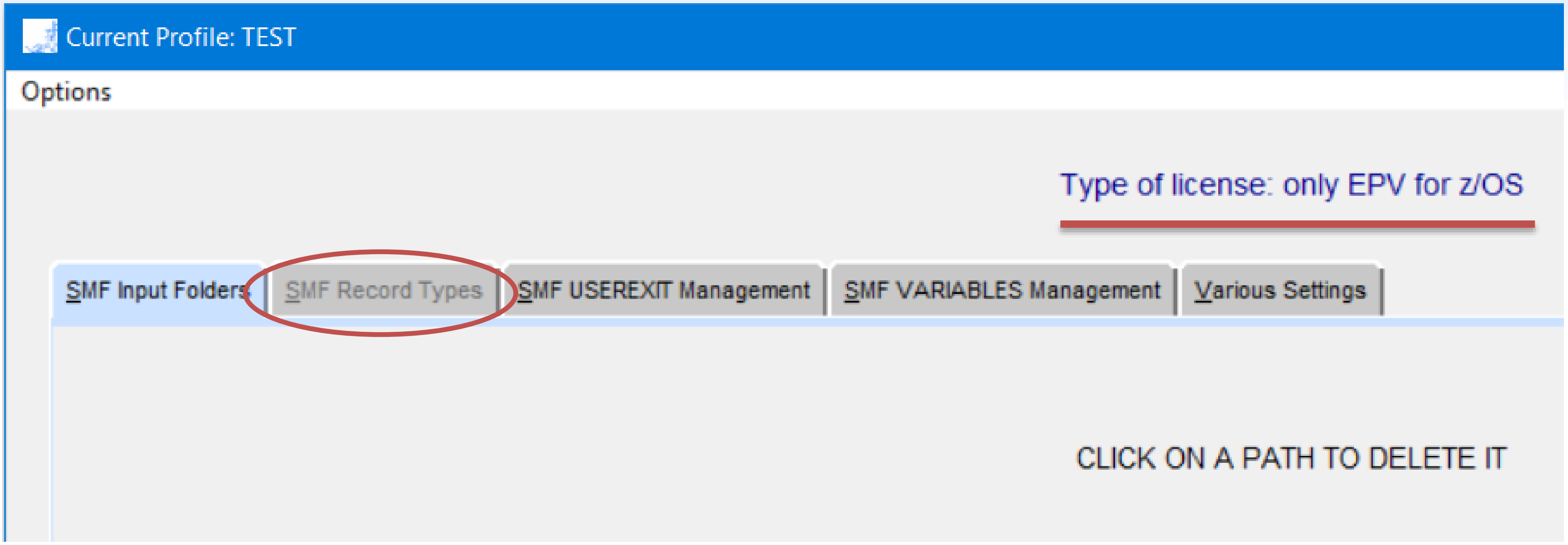
- Lite: available only the records and fields needed by the licensed EPV products (EPV for z/OS, EPV for Db2, EPV for zLINUX and EPV for MQ)

- Full: all the desired records and fields; it's up to the customer to select what has to be processed

- Big Data: same as Full but with more possibilities to configure the architecture (ideal for large size customers)

zParser data collection architecture - Licenses

- Lite: no possibility to choose between record types/subtypes and fields


Current Profile: TEST

Options

Type of license: only EPV for z/OS

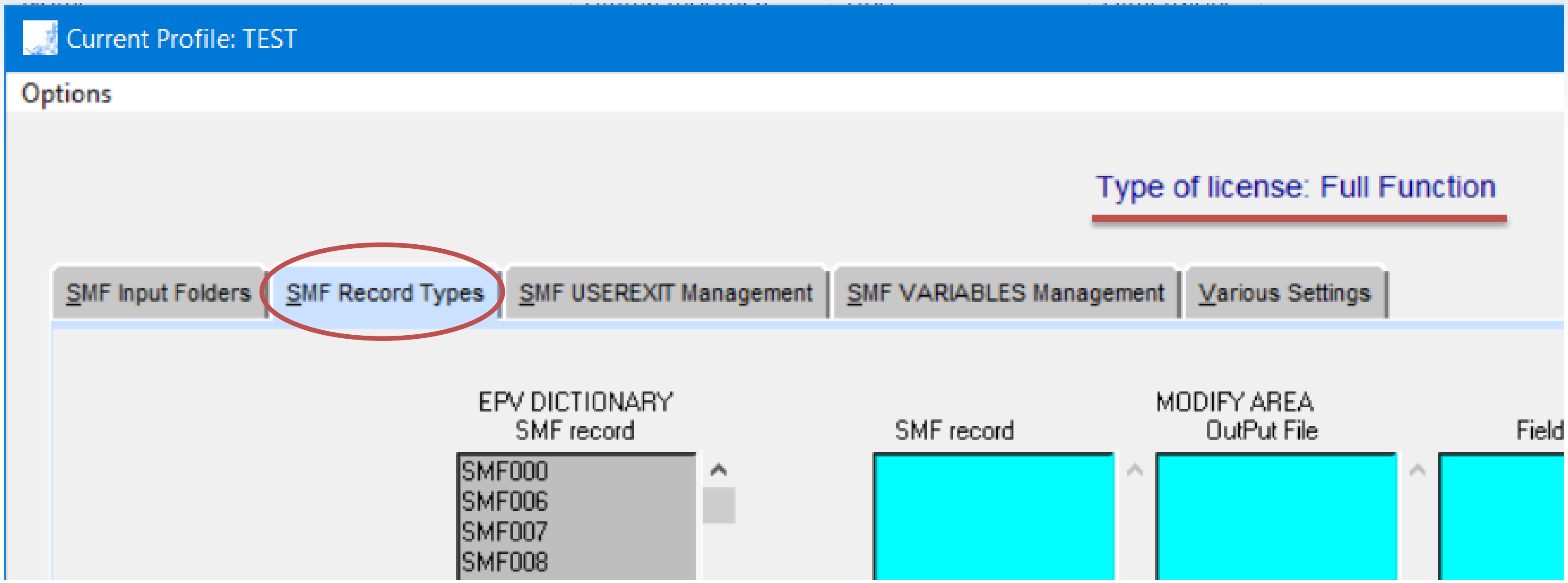
CLICK ON A PATH TO DELETE IT

zParser data collection architecture - Licenses

- Lite licenses are always related to other EPV products installed by the customer. So, these are the available lite licenses with different combination of EPV products:
  - ✓ only EPV for z/OS
  - ✓ only EPV for Db2
  - ✓ only EPV for zLINUX
  - ✓ only EPV for MQ
  - ✓ only EPV for z/OS & EPV for Db2
  - ✓ only EPV for z/OS & EPV for zLINUX
  - ✓ only EPV for Db2 & EPV for zLINUX
  - ✓ only EPV for z/OS & EPV for MQ
  - ✓ only EPV for Db2 & EPV for MQ
  - ✓ only EPV for MQ & EPV for zLINUX
  - ✓ EPV for: z/OS - Db2 - zLINUX
  - ✓ EPV for: z/OS - Db2 - MQ
  - ✓ EPV for: z/OS - zLINUX - MQ
  - ✓ EPV for: Db2 - zLINUX - MQ
  - ✓ EPV for: z/OS - Db2 - zLINUX - MQ

zParser data collection architecture - Licenses

- Full: possibility to choose between record types/subtypes and fields



Current Profile: TEST

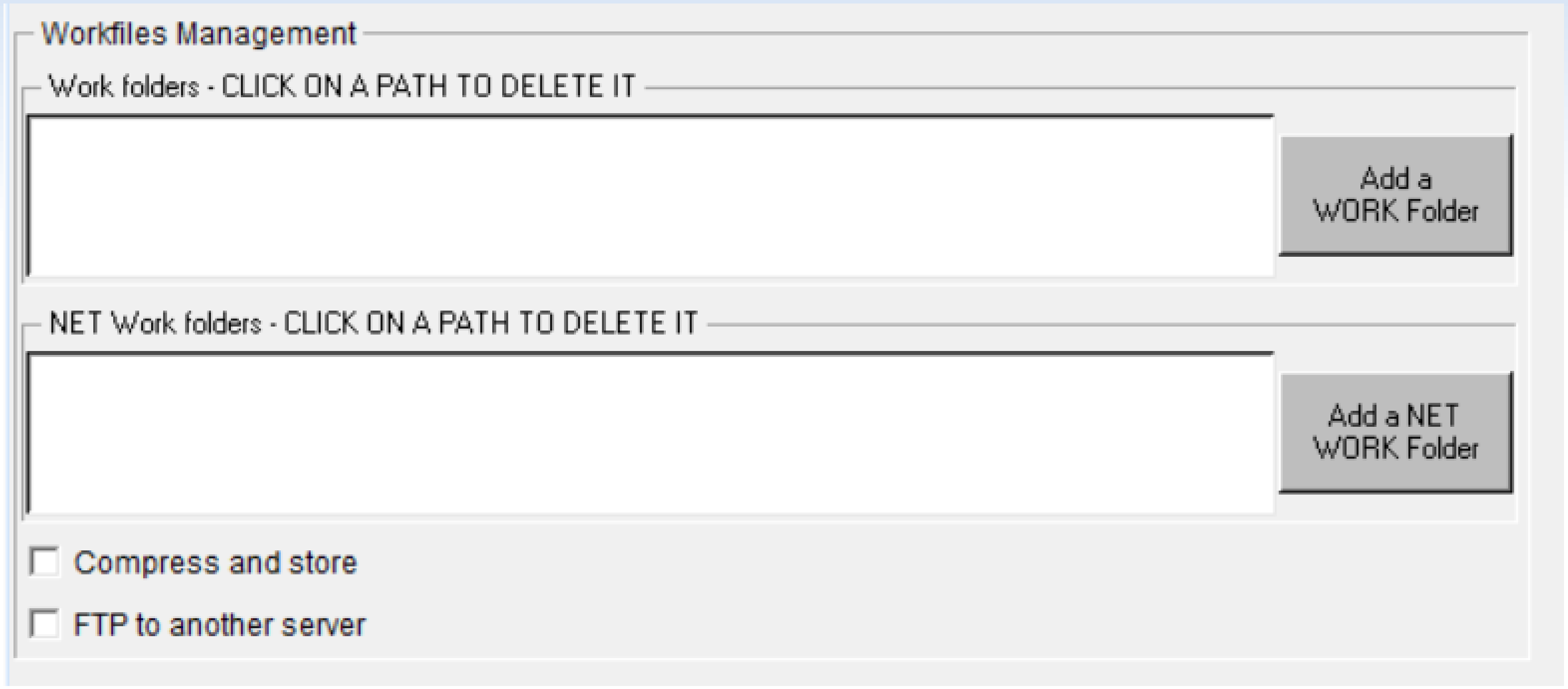
Options

Type of license: Full Function

SMF000
SMF006
SMF007
SMF008

zParser data collection architecture - Licenses

- Big Data: completely customizable process


Workfiles Management

Work folders - CLICK ON A PATH TO DELETE IT

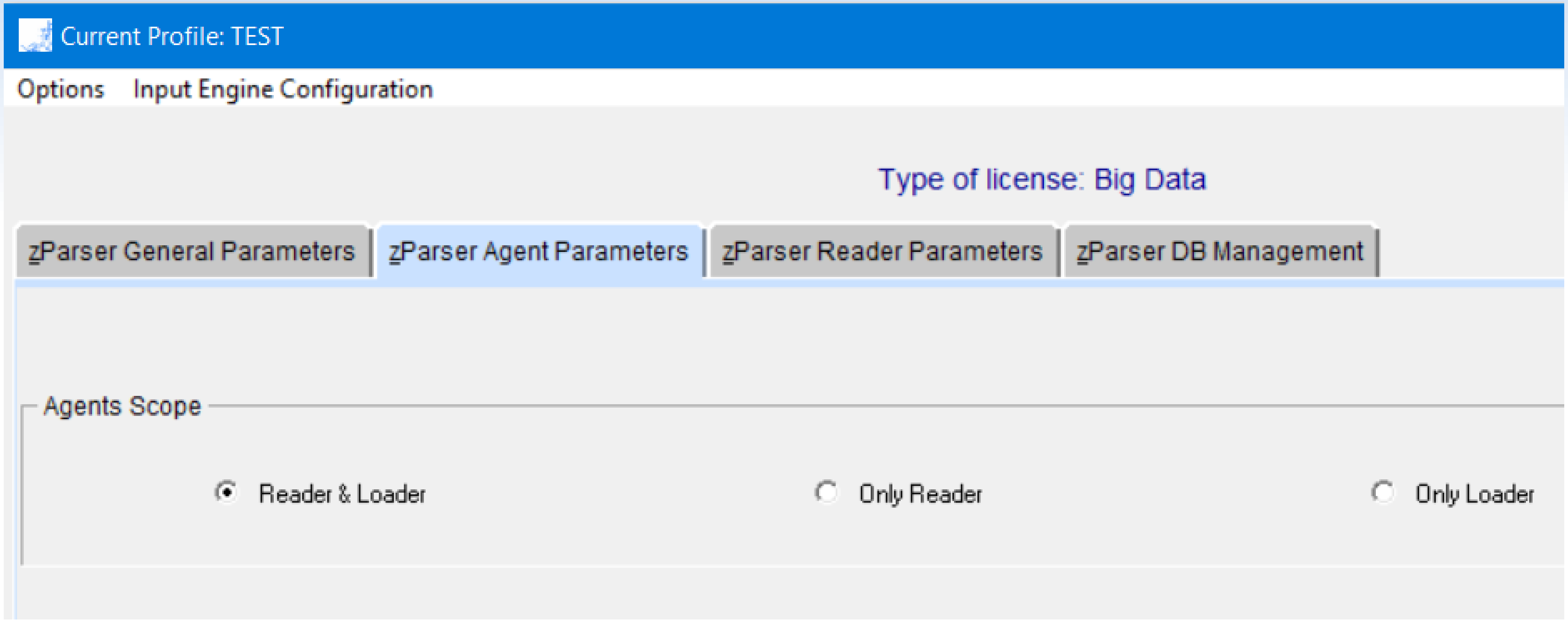
NET Work folders - CLICK ON A PATH TO DELETE IT

☐ Compress and store

☐ FTP to another server

zParser data collection architecture - Licenses

- Big Data: completely customizable process


Current Profile: TEST

Options   Input Engine Configuration

Type of license: Big Data

Agents Scope

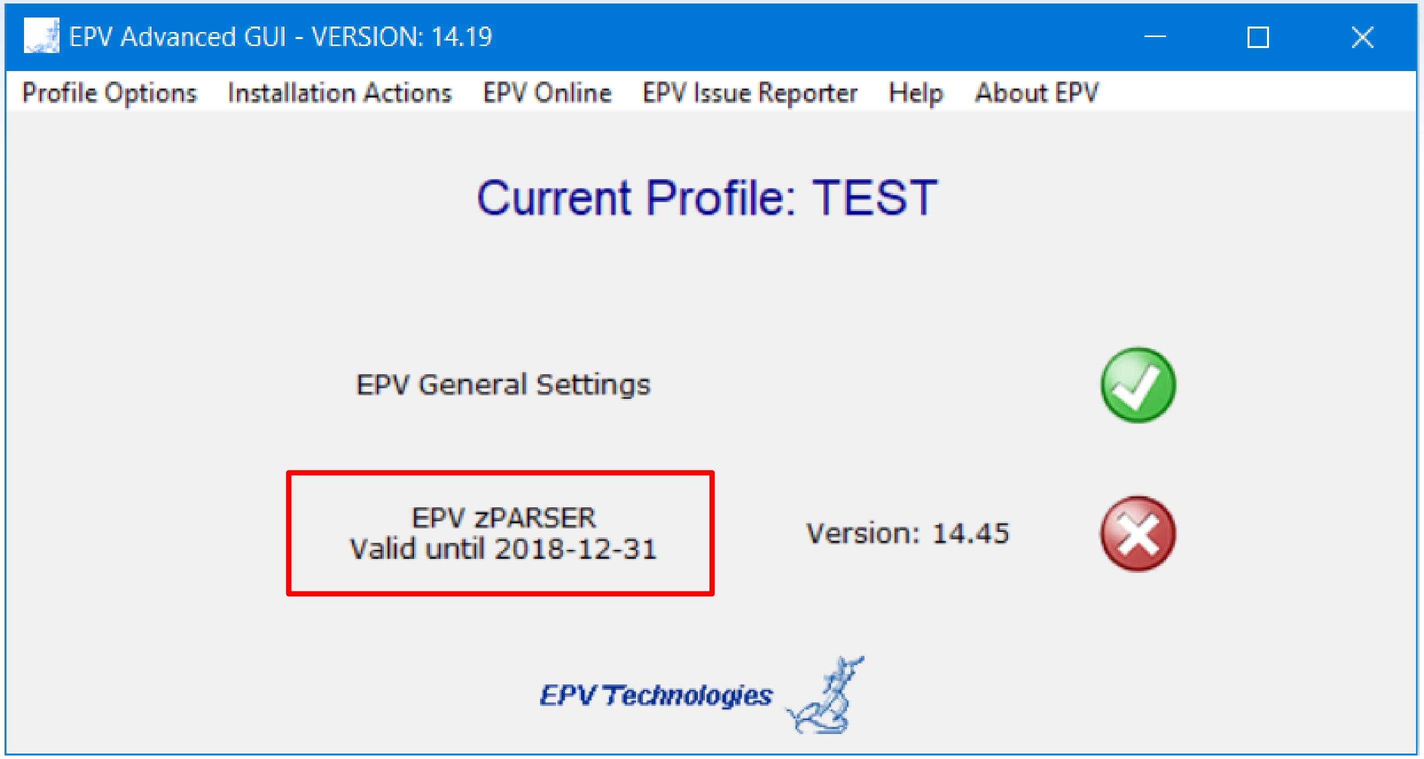
[✓ Reader & Loader]   Only Reader   Only Loader

zParser data collection architecture - Fields selection

- Available with Full or Big Data licenses
- Graphical user interface to easily make the selection
- Also possible when running in Continuous mode, with no impact on the daily processing, through the ‘Promote Profile’
- Available for both Windows and Linux user profiles

zParser data collection architecture - Fields selection

- Run EPVAdvancedGUI.exe in EPVROOT/SETUP
- Press the EPV zParser button



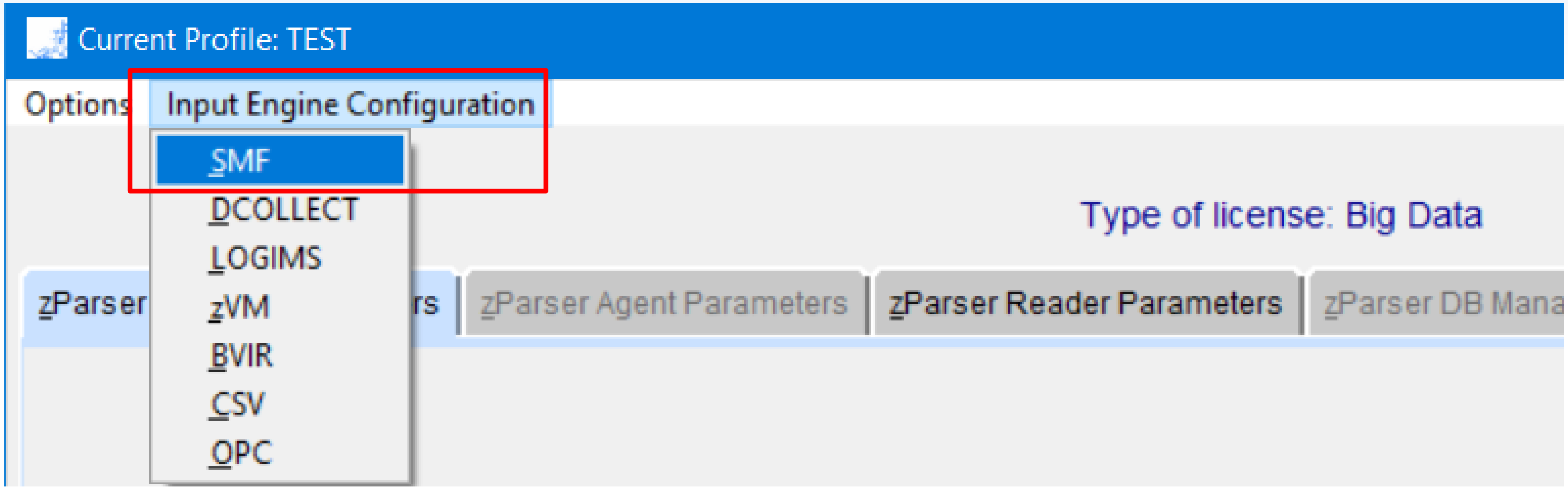
Current Profile: TEST

EPV General Settings ✓

EPV zPARSER
Valid until 2018-12-31
Version: 14.45 ✗

zParser data collection architecture - Fields selection

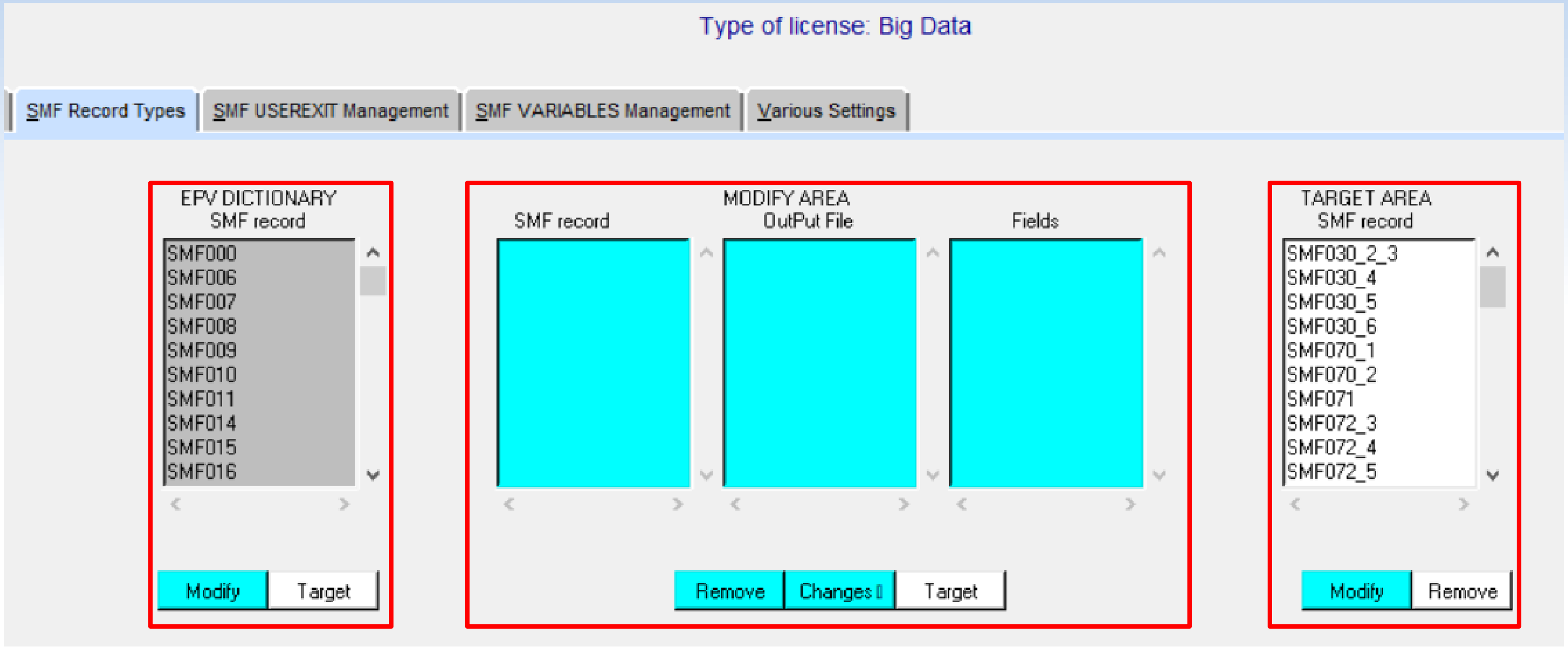
- Select one Input Engine of your interest (for example SMF)



zParser data collection architecture - Fields selection

- The ‘SMF Record Types’ tab is divided in 3 areas:
  - ✓ EPV Dictionary: it contains ALL the available SMF records
  - ✓ Modify area: it contains the record type in which you want to make a selection on specific subtypes or fields
  - ✓ Target Area: it contains the SMF records selected by the user

zParser data collection architecture - Fields selection



SMF000
SMF006
SMF007
SMF008
SMF009
SMF010
SMF011
SMF014
SMF015
SMF016

SMF030_2_3
SMF030_4
SMF030_5
SMF030_6
SMF070_1
SMF070_2
SMF071
SMF072_3
SMF072_4
SMF072_5

zParser data collection architecture - Fields selection

- To add one SMF record type to the user's selection:

  ✓Select the SMF record from the EPV Dictionary area

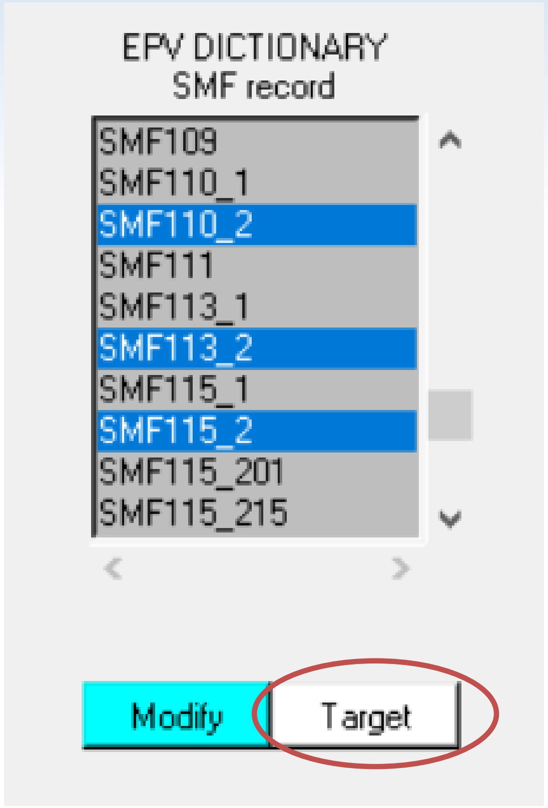
  (Also a multiple selection can be performed by using the Ctrl key on the keyboard)

  ✓Press the Target button below the EPV Dictionary area

  ✓Once added all the desired records, select Options, then Save

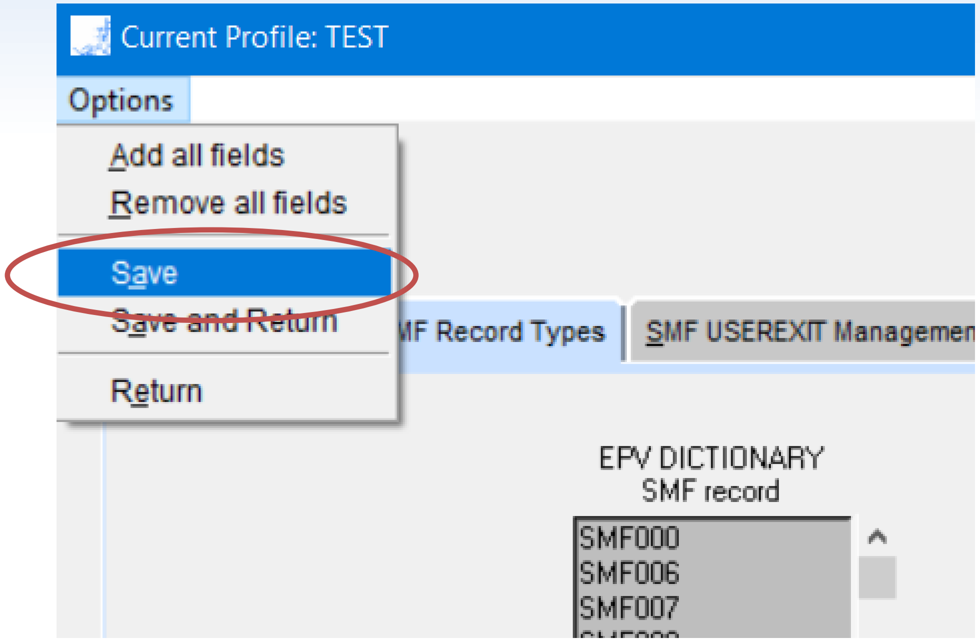
zParser data collection architecture - Fields selection

- ✓Select the SMF record from the EPV Dictionary area
- ✓Press the Target button below the EPV Dictionary area



zParser data collection architecture - Fields selection

✓Once added all the desired records, select Options, then Save

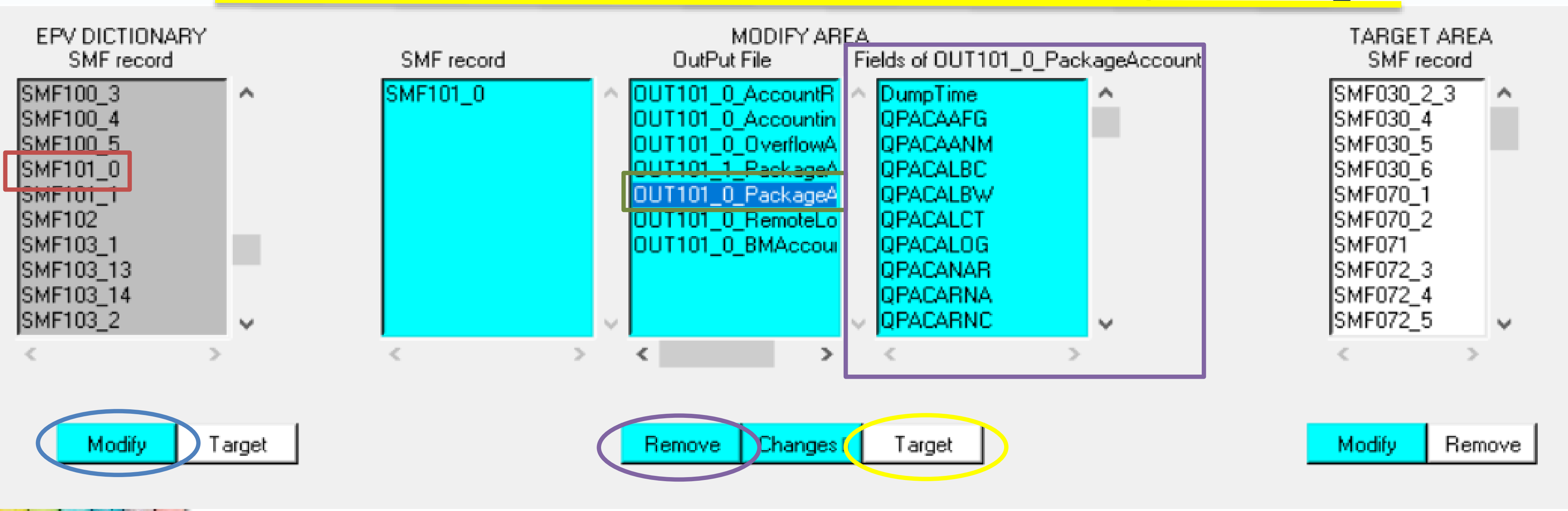


zParser data collection architecture - Fields selection

- To select only some specific SMF fields inside an SMF record:
  - ✓ Select the SMF record from the EPV Dictionary area
  - ✓ Press the Modify button below the EPV Dictionary area
  - ✓ Select the table from the middle area
  - ✓ Select the fields to remove and press the Remove button
  - ✓ Once removed all the non necessary fields, press Target
  - ✓ Once finished, select Options, then Save

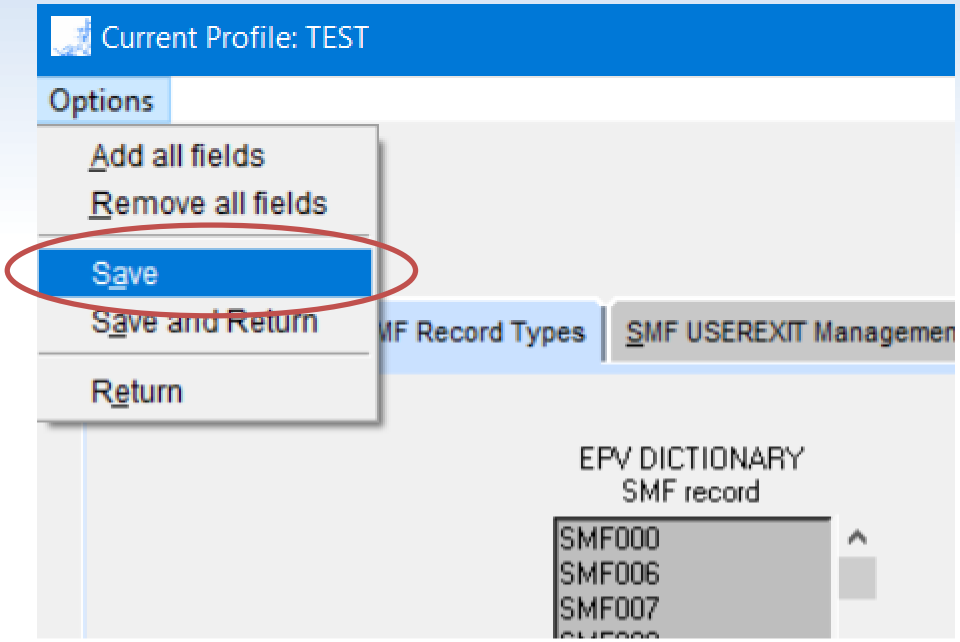
zParser data collection architecture - Fields selection

- ✓ Select the SMF record from the EPV Dictionary area
- ✓ Press the Modify button below the EPV Dictionary area
- ✓ Select the table from the middle area
- ✓ Select the fields to remove and press the Remove button
- ✓ Once removed all the non necessary fields, press Target



zParser data collection architecture - Fields selection

- ✓ Once finished, select Options, then Save

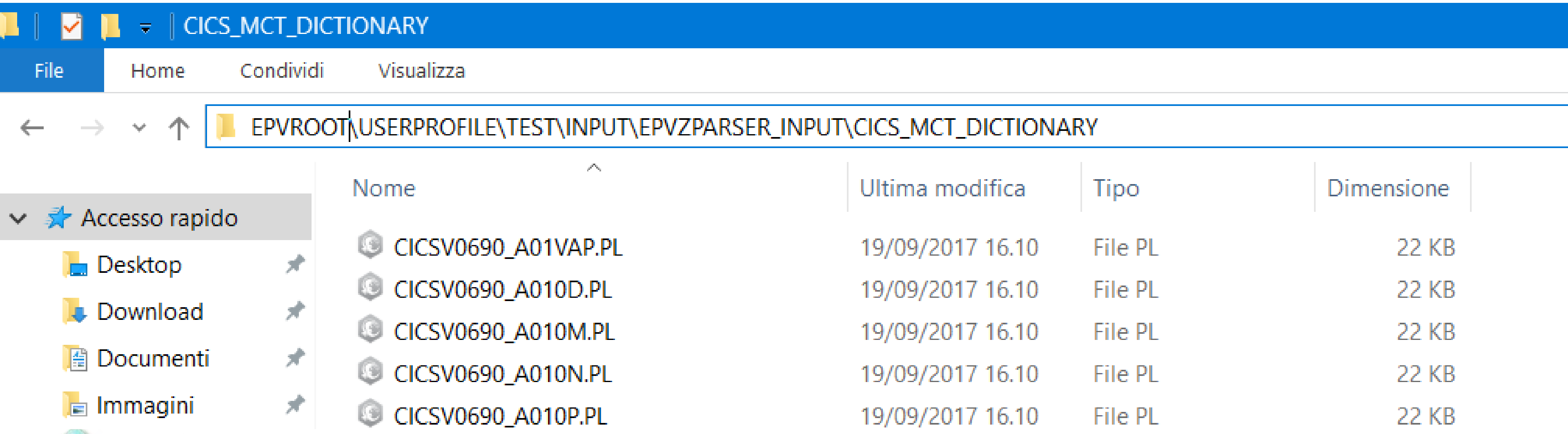


zParser data collection architecture - CICS dictionary

- The monitoring control table (MCT) defines the user data fields in CICS monitoring performance class records and describes how they are manipulated at event monitoring points (EMPs)
- It also controls which system-defined performance class data fields are recorded
- It is put inside the SMF 110 records at every CICS restart
- EPV zParser automatically adapts its reading routines accordingly to the MCT customizations the users make

zParser data collection architecture - CICS dictionary

- The EPV zParser's reading routines about MCT are stored by default in EPVROOT/USERPROFILE/*PROFILE_NAME*/INPUT/EPV ZPARSER_INPUT/CICS_MCT_DICTIONARY

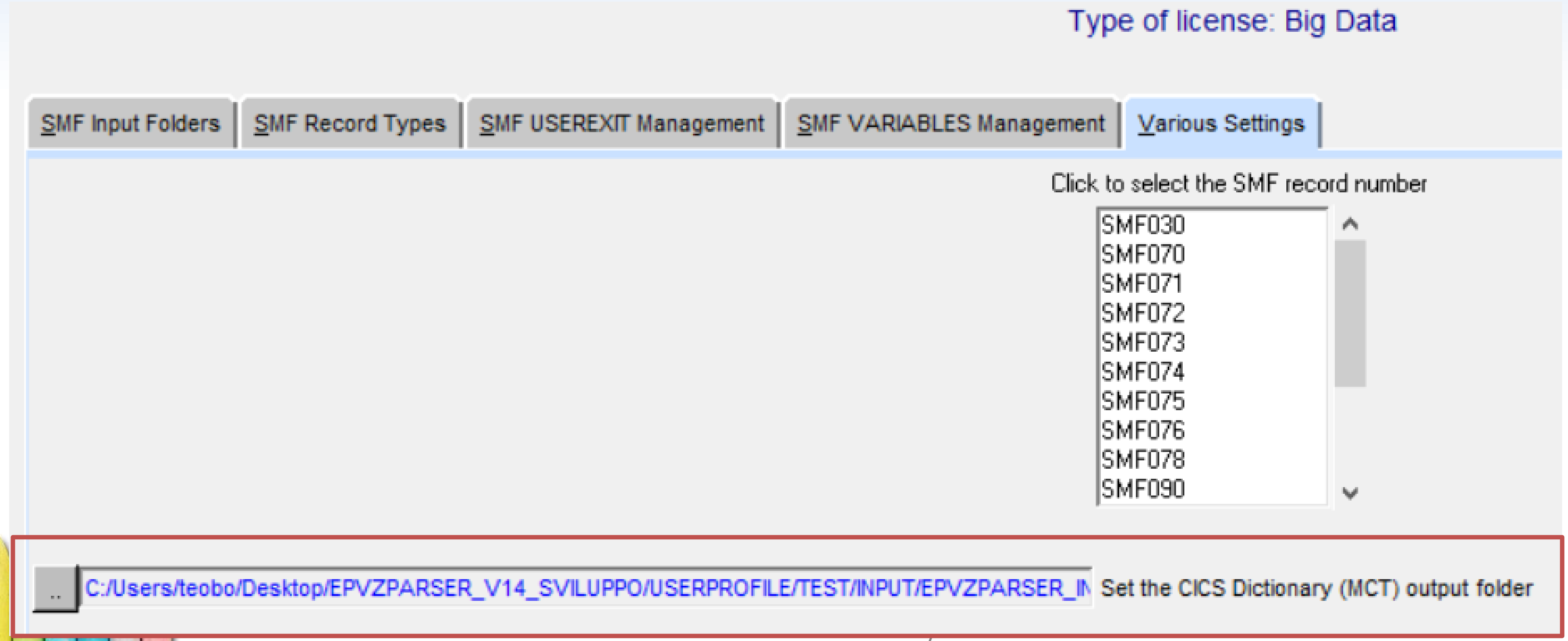


EPVROOT\USERPROFILE\TEST\INPUT\EPVZPARSER_INPUT\CICS_MCT_DICTIONARY

Nome	Ultima modifica	Tipo	Dimensione
CICSV0690_A01VAP.PL	19/09/2017 16.10	File PL	22 KB
CICSV0690_A010D.PL	19/09/2017 16.10	File PL	22 KB
CICSV0690_A010M.PL	19/09/2017 16.10	File PL	22 KB
CICSV0690_A010N.PL	19/09/2017 16.10	File PL	22 KB
CICSV0690_A010P.PL	19/09/2017 16.10	File PL	22 KB

zParser data collection architecture - CICS dictionary

- It is also possible to choose where to store these reading routines in the Various Settings tab


Type of license: Big Data

Click to select the SMF record number

SMF030
SMF070
SMF071
SMF072
SMF073
SMF074
SMF075
SMF076
SMF078
SMF090

C:/Users/teobo/Desktop/EPVZPARSER_V14_SVILUPPO/USERPROFILE/TEST/INPUT/EPVZPARSER_IN Set the CICS Dictionary (MCT) output folder

zParser data collection architecture - Input Engines

- As well as SMF, other Input Engines are available:
  - ✓ DCOLLECT
  - ✓ IMS
  - ✓ z/VM
  - ✓ BVIR
  - ✓ CSV
  - ✓ OPC

zParser data collection architecture - DCOLLECT

- zParser is able to read the following IDCAMS DCOLLECT records
  - ✓ DCOLLECT_A -> “VSAM base cluster association name information“
  - ✓ DCOLLECT_AG -> "Aggregate Group definition “
  - ✓ DCOLLECT_AI -> "Accounting Information “
  - ✓ DCOLLECT_B -> “Data Set Backup Version Information”
  - ✓ DCOLLECT_BC -> “Base Configuration Information”
  - ✓ DCOLLECT_C -> “Dasd capacity planning information”
  - ✓ DCOLLECT_CN -> “Sms cache names definition”
  - ✓ **DCOLLECT_D -> “Active data set information”**
  - ✓ DCOLLECT_DC -> “Data class construct information”

zParser data collection architecture - DCOLLECT

- ✓DCOLLECT_DR -> “Sms optical drive definition”
- ✓DCOLLECT_LB -> “Sms optical library definition”
- ✓DCOLLECT_M -> “Migrated data set information”
- ✓DCOLLECT_MC -> “Management class construct information”
- ✓DCOLLECT_SC -> “Storage class construct information”
- ✓DCOLLECT_SG -> “Storage group construct information”
- ✓DCOLLECT_T -> “Tape capacity planning information”
- ✓**DCOLLECT_V -> “Volume information”**
- ✓DCOLLECT_VL -> “Sms volume definition”

zParser data collection architecture - DCOLLECT

- Only the following 2 types of DCOLLECT are provided in LITE license:
  - ✓**DCOLLECT_D**
  - ✓**DCOLLECT_V**

- With Big Data and Full licenses, also a selection on the fields can be done as described in the previous slides

zParser data collection architecture - IMS

- zParser is able to read differenent kind of performance data generated by Information Management System (IMS)
  - ✓ LOGIMS_0A -> “CPI-CI Driven Program“
  - ✓ LOGIMS_01 -> "Message queue record (message received from a CNT).“
  - ✓ LOGIMS_03 -> "Message queue record (message received from a PSB or IMS).“
  - ✓ **LOGIMS_07 -> “Program Termination Accounting Record”**
  - ✓ **LOGIMS_08 -> “Program Schedule Record”**
  - ✓ LOGIMS_31 -> “Message queue GU record”
  - ✓ LOGIMS_32 -> “Message queue reject record”
  - ✓ LOGIMS_33 -> “Message queue DRRN free record”

zParser data collection architecture - IMS

- ✓ LOGIMS_34 -> “Message queue cancel record”
- ✓ LOGIMS_35 -> “Message queue enqueue record”
- ✓ LOGIMS_36 -> “Message queue dequeue record”
- ✓ LOGIMS_37 -> “Message queue syncpoint transfer record”
- ✓ LOGIMS_38 -> “Message queue syncpoint fail record”
- ✓ **LOGIMS_56FA -> “Transaction Level Statistics”**
- ✓ LOGIMS_59 -> “Fast path records (input message, output message, dequeue message, syncpoint, syncpoint fail”
- ✓ LOGIMS_CHKPTSTATS -> “Checkpoint Statistics”
- ✓ LOGIMS_CHKPTVSAM -> “Statistics VSAM subpool”
- ✓ LOGIMS_F9 -> “Program log record”
- ✓ **LOGIMS_FA -> “Transaction log record”**

zParser data collection architecture - IMS

- Only the following 3 types of IMS logs are provided in LITE license:
  - ✓ **LOGIMS_56FA (DEFAULT)**
  - ✓ **LOGIMS_FA**
  - ✓ **LOGIMS_07/8**

- With Big Data and Full licenses, also a selection on the fields can be done as described in the previous slides

zParser data collection architecture - z/VM

- zParser is able to read the following z/VM records produced by the z/VM Monitor
  - ✓ZVM_6_4 -> “Cache Activity Data“
  - ✓ZVM_0_1 -> "System Data“
  - ✓ZVM_0_2 -> "Processor Data“
  - ✓ZVM_0_3 -> “Real storage allocation and use”
  - ✓ZVM_0_5 -> “Expanded Storage Data Per Processor”
  - ✓ZVM_0_14 -> “Assess use and benefits derived from expanded storage”
  - ✓ZVM_0_16 -> “CPU utilization data for logical partition”
  - ✓ZVM_0_17 -> “CPU Utilization data for LPAR management”
  - ✓ZVM_0_20 -> “Extended Channel Measurement Data”

zParser data collection architecture - z/VM

- ✓ZVM_1_4 -> “System Configuration”
- ✓ZVM_1_5 -> “Processor Configuration”
- ✓ZVM_1_6 -> “Device Configuration Data”
- ✓ZVM_1_7 -> “Memory Configuration Data”
- ✓ZVM_1_8 -> “Paging Configuration Data”
- ✓ZVM_1_9 -> “Sample Profile”
- ✓ZVM_1_17 -> “Expanded Store Data”
- ✓ZVM_3_4 -> “Auxiliary Storage Management”
- ✓ZVM_4_3 -> “User activity data”
- ✓ZVM_4_4 -> “User Interaction Data”
- ✓ZVM_6_3 -> “Device Activity”

zParser data collection architecture - z/VM

- All the record types are provided in LITE licenses that include EPV for zLinux product:
  - ✓ only EPV for zLINUX
  - ✓ only EPV for z/OS & EPV for zLINUX
  - ✓ only EPV for Db2 & EPV for zLINUX
  - ✓ only EPV for MQ & EPV for zLINUX
  - ✓ EPV for: z/OS - Db2 - zLINUX
  - ✓ EPV for: z/OS - zLINUX - MQ
  - ✓ EPV for: Db2 - zLINUX - MQ
  - ✓ EPV for: z/OS - Db2 - zLINUX - MQ
- With Big Data and Full licenses, also a selection on the fields can be done as described in the previous slides

zParser data collection architecture - BVIR

- zParser is able to read the following records produced by IBM TS7700 VTS

✓BVIR020 -> “Vnode Virtual Device Container“
✓BVIR021 -> “Vnode Adapter Historical Record“
✓BVIR030 -> “Hnode HSM Historical Record“
✓BVIR031 -> “Hnode Export/Import Container”
✓BVIR032 -> “Hnode Library Historical Record”
✓BVIR033 -> “Hnode Grid Historical Record”

zParser data collection architecture - BVIR

- All the record types are provided in LITE licenses that include EPV for z/OS product:
  - ✓ only EPV for z/OS
  - ✓ only EPV for z/OS & EPV for Db2
  - ✓ only EPV for z/OS & EPV for zLINUX
  - ✓ only EPV for z/OS & EPV for MQ
  - ✓ EPV for: z/OS - Db2 - zLINUX
  - ✓ EPV for: z/OS - Db2 - MQ
  - ✓ EPV for: z/OS - zLINUX - MQ
  - ✓ EPV for: z/OS - Db2 - zLINUX - MQ
- With Big Data and Full licenses, also a selection on the fields can be done as described in the previous slides

zParser data collection architecture - OPC

- With Big Data and Full licenses, Tivoli Operations Planning and Control (OPC) data can be read

- Information related to “OPC CURRENT PLAN” and “OPC JOBS END”

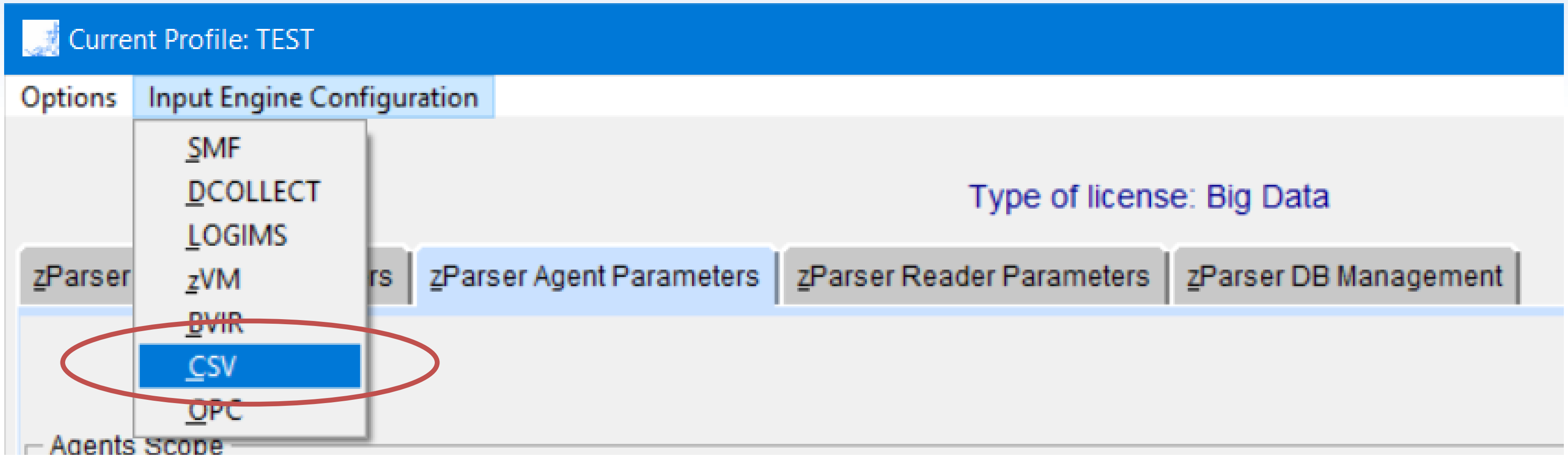
zParser data collection architecture - CSV input files

- In case of Big Data or Full license, there is also the possibility to load CSV files inside the zParser's database

- In order to do that you have to:
  - ✓ Configure the CSV Input Engine (define input folder and fields separator)
  - ✓ Create the CSV files by following a few rules
  - ✓ Put the CSV files inside the zParser's CSV input folder

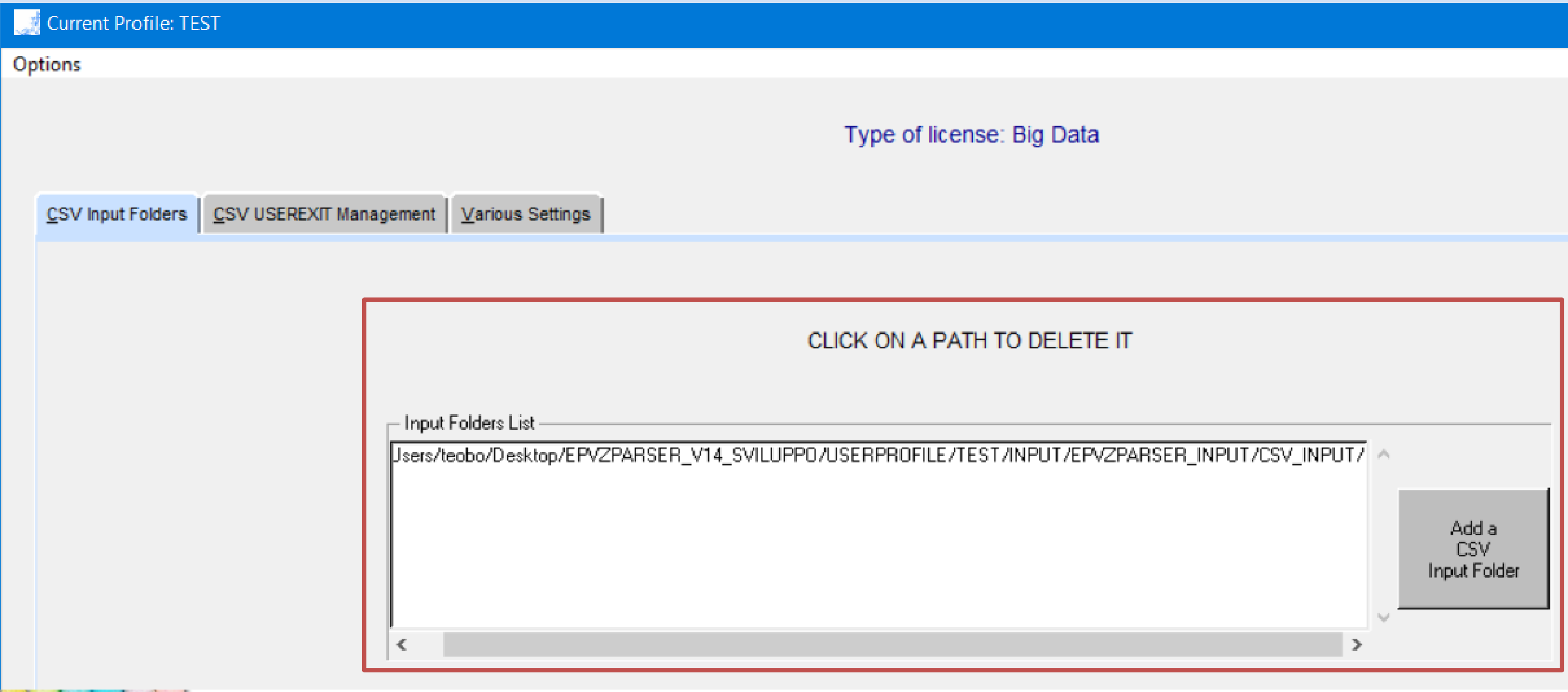
zParser data collection architecture - CSV input files

- Configure the CSV Input Engine



zParser data collection architecture - CSV input files

- Define one or more input folders


Current Profile: TEST

Options

Type of license: Big Data

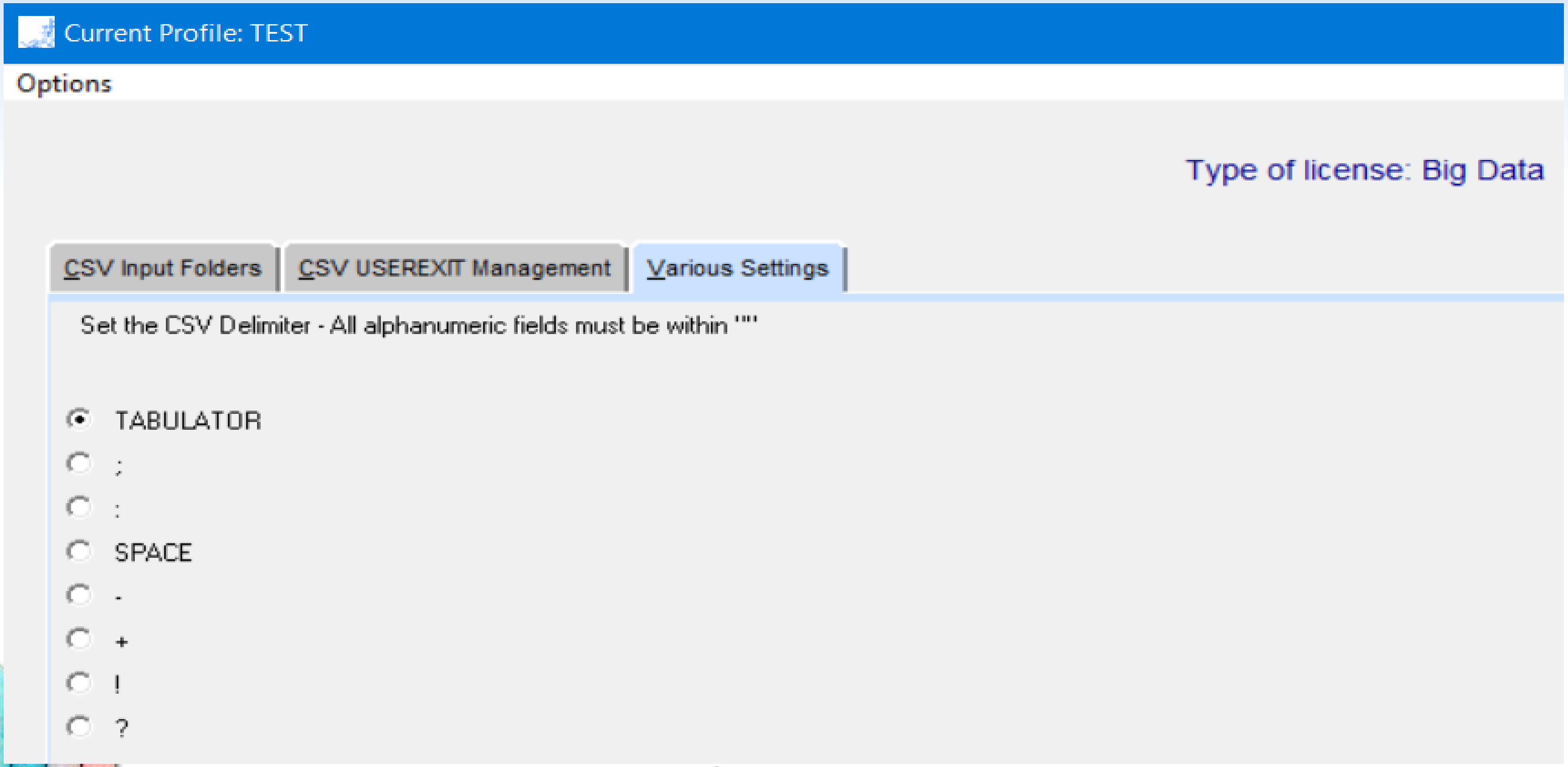
CLICK ON A PATH TO DELETE IT

Input Folders List

Jsers/teobo/Desktop/EPVZPARSER_V14_SVILUPPO/USERPROFILE/TEST/INPUT/EPVZPARSER_INPUT/CSV_INPUT/

zParser data collection architecture - CSV input files

- Choose the fields separator in the Various Settings tab



Current Profile: TEST

Options

Type of license: Big Data

Set the CSV Delimiter - All alphanumeric fields must be within ""

[✓ TABULATOR]

;

:

SPACE

-

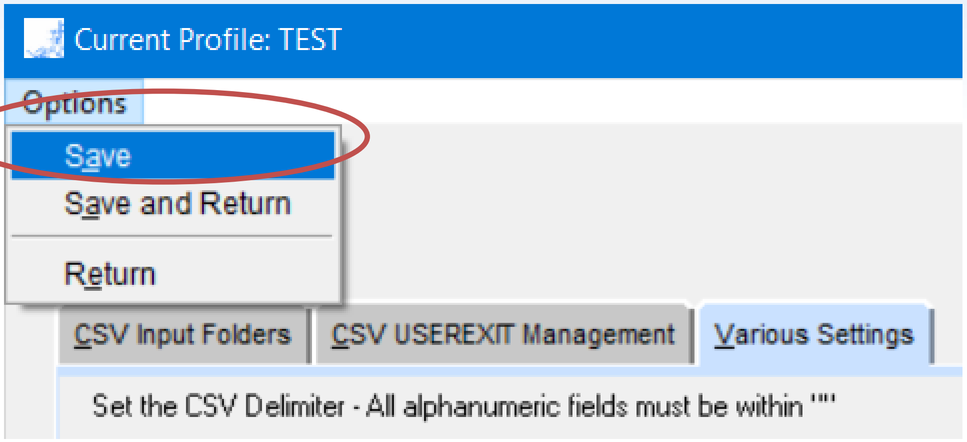
+

!

?

zParser data collection architecture - CSV input files

- Once finished, select Options, then Save



Set the CSV Delimiter - All alphanumeric fields must be within ""

zParser data collection architecture - CSV input files

- Create the CSV files by following a few rules
  - ✓ The first four rows of the file have to contain the names and the description of the variables as follows:
    - ➢ 00 Variable name
    - ➢ 01 Variable format
    - ➢ 02 Variable description (It is not mandatory but we suggest to use it; in case the row is missing the variable name is used as description)
    - ➢ 03 Flag of the end of the description part (e.g." 03;USR")
  - ✓ Use the field separator that you’ve previously configured
  - ✓ The file name is used to create the table in the zParser database

zParser data collection architecture - CSV input files

- Example of a correctly-formatted CSV file with semicolon as separator

```
00;USER;APPLICATION;DEPARTMENT;CITY;CONNECT_TIME
01;CHAR(8);VARCHAR(250);VARCHAR(64);DECIMAL(12.3)
02;User Name;Application Name;office department;City;duration of connection
03;USR
U129907;sales;account;London;8:20
U129955;sales;account;London;6:00
U449955;sales;marketing;Paris;9:00
U992255;payroll;organic;Rome;9:00
```

zParser data collection architecture - SMF user records

- zParser supports the following SMF user records:
  - ✓ IBM Tivoli A/F Operator
  - ✓ BMC CONTROL-D
  - ✓ IBM Hierarchical Storage Management (HSM)
  - ✓ Tape Activity Monitor
  - ✓ Syncsort MFX
  - ✓ CA-TPX
  - ✓ Oracle Virtual Tape Control System (VTCS)
  - ✓ IBM Virtual Tape Server (VTS)
  - ✓ zCost

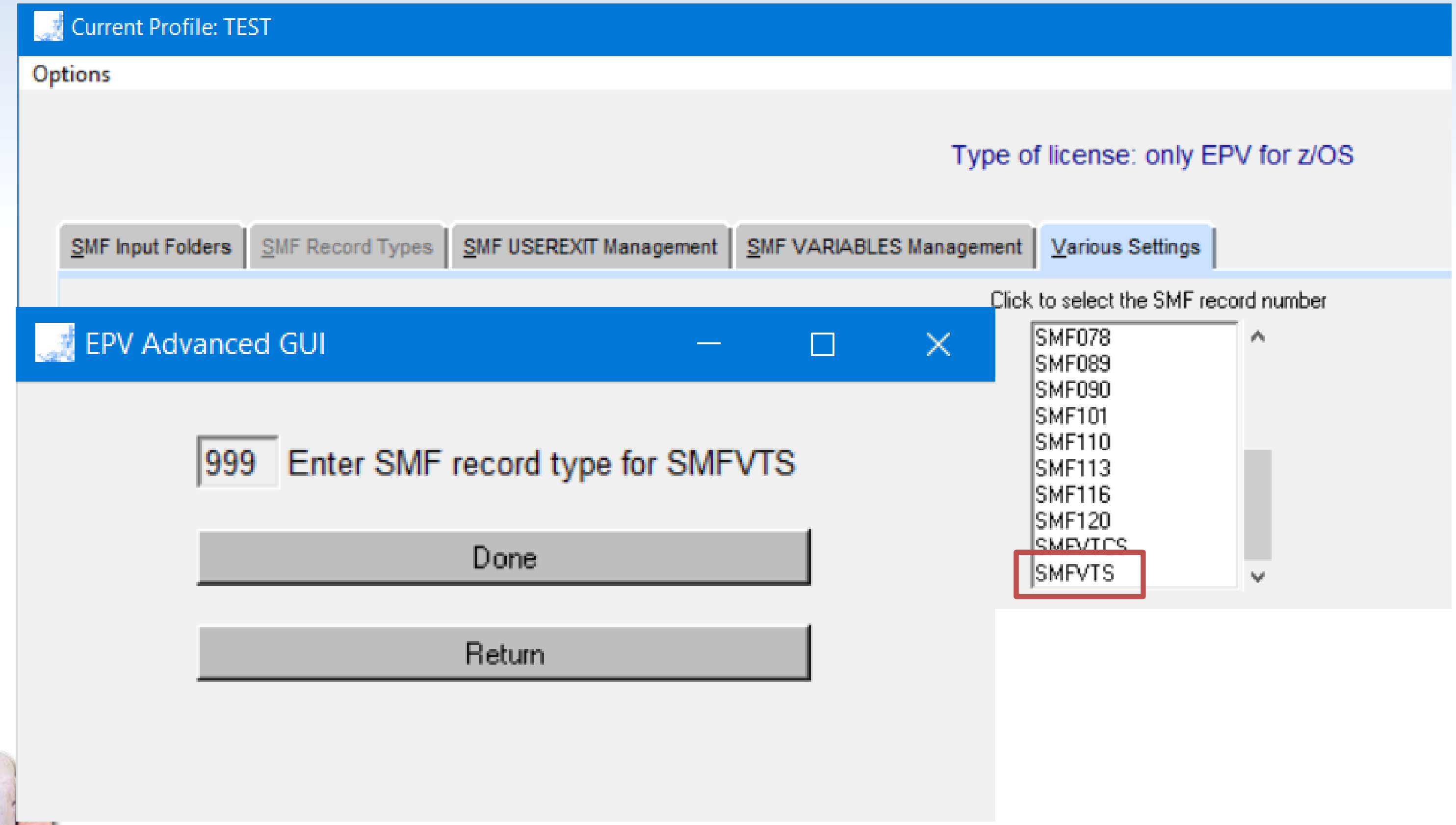
zParser data collection architecture - SMF user records

- In case of LITE license, only the following are provided:
  - ✓ Oracle Virtual Tape Control System (VTCS)
  - ✓ IBM Virtual Tape Server (VTS)

- It is possible to configure the SMF record number for each SMF record through the EPVAdvancedGUI.exe program

zParser data collection architecture - SMF user records

- To configure the SMF record number, go inside the SMF Input Engine section, click on the Various Settings tab



Current Profile: TEST

Options

Type of license: only EPV for z/OS

SMF Input Folders | SMF Record Types | SMF USEREXIT Management | SMF VARIABLES Management | Various Settings

Click to select the SMF record number

SMF078
SMF089
SMF090
SMF101
SMF110
SMF113
SMF116
SMF120
SMFVTCS
SMFVTS

EPV Advanced GUI

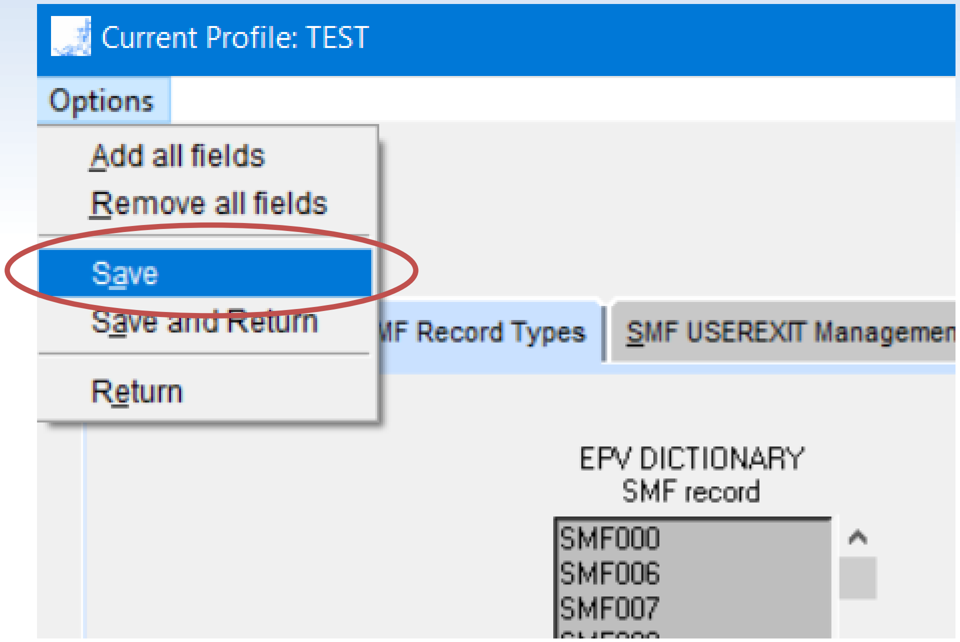
999 Enter SMF record type for SMFVTS

Done

Return

zParser data collection architecture - SMF user records

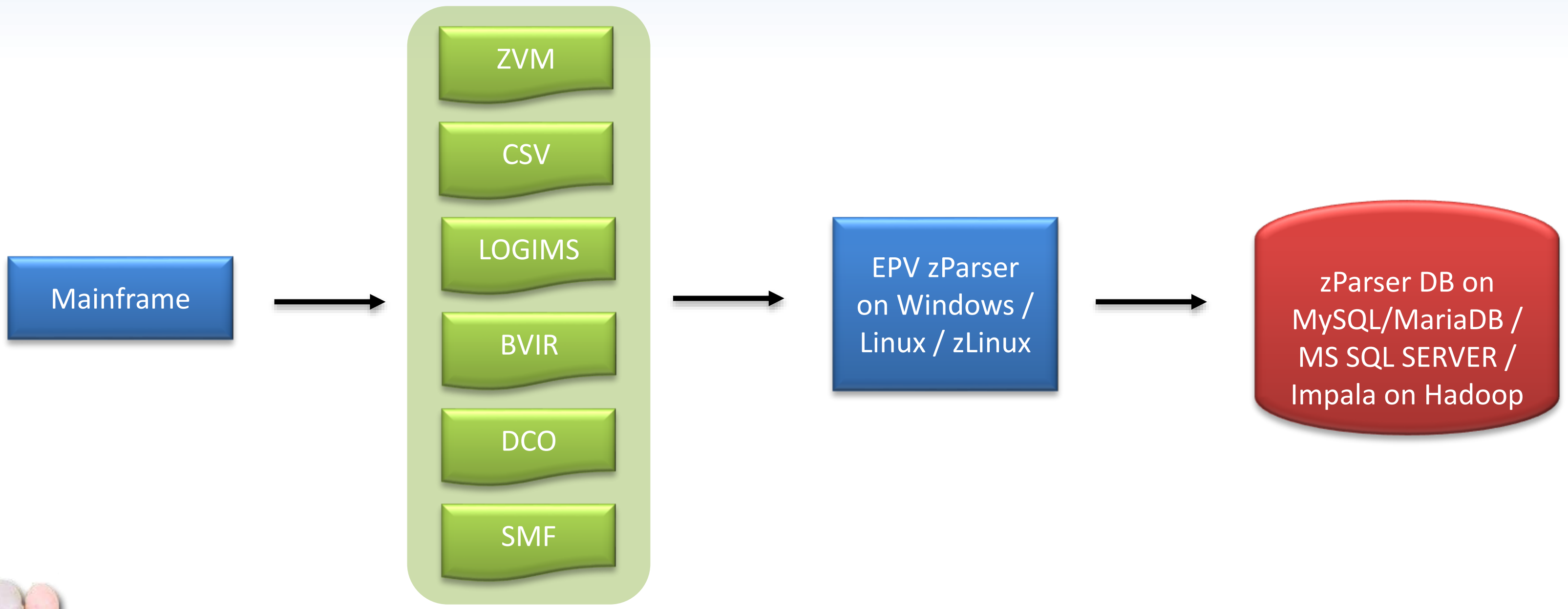
- Once finished, select Options, then Save



zParser basic processing

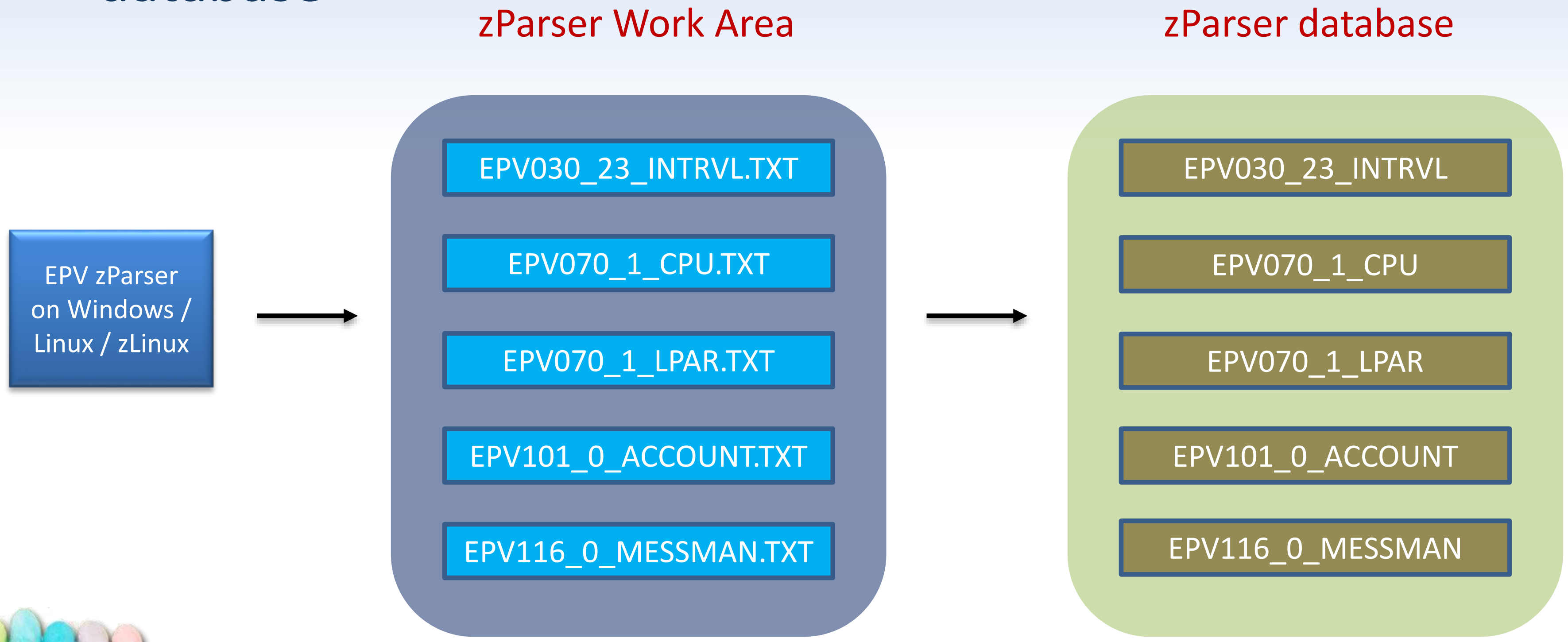
zParser basic processing - Daily flow

- As seen in the previous slides, the data collection can be run once a day or multiple times during the day


Mainframe
ZVM
CSV
LOGIMS
BVIR
DCO
SMF
EPV zParser on Windows / Linux / zLinux
zParser DB on MySQL/MariaDB / MS SQL SERVER / Impala on Hadoop

zParser basic processing - Daily flow

- SMF data is being transformed into multiple TXT files, one for each record type / subtype, and then loaded into the database



zParser basic processing - Daily flow

- In ‘Once a day’, the daily flow is very simple:
  - ✓ Send the data once
  - ✓ Start EPV (zParser and the other products) once

- In Continuous mode, the daily flow is a little bit more complex:
  - ✓ Send the data as soon as it is available or every x minutes
  - ✓ zParser automatically parse each file as soon as it arrives on the EPV server and loads data inside the database
  - ✓ Start the other EPV products once

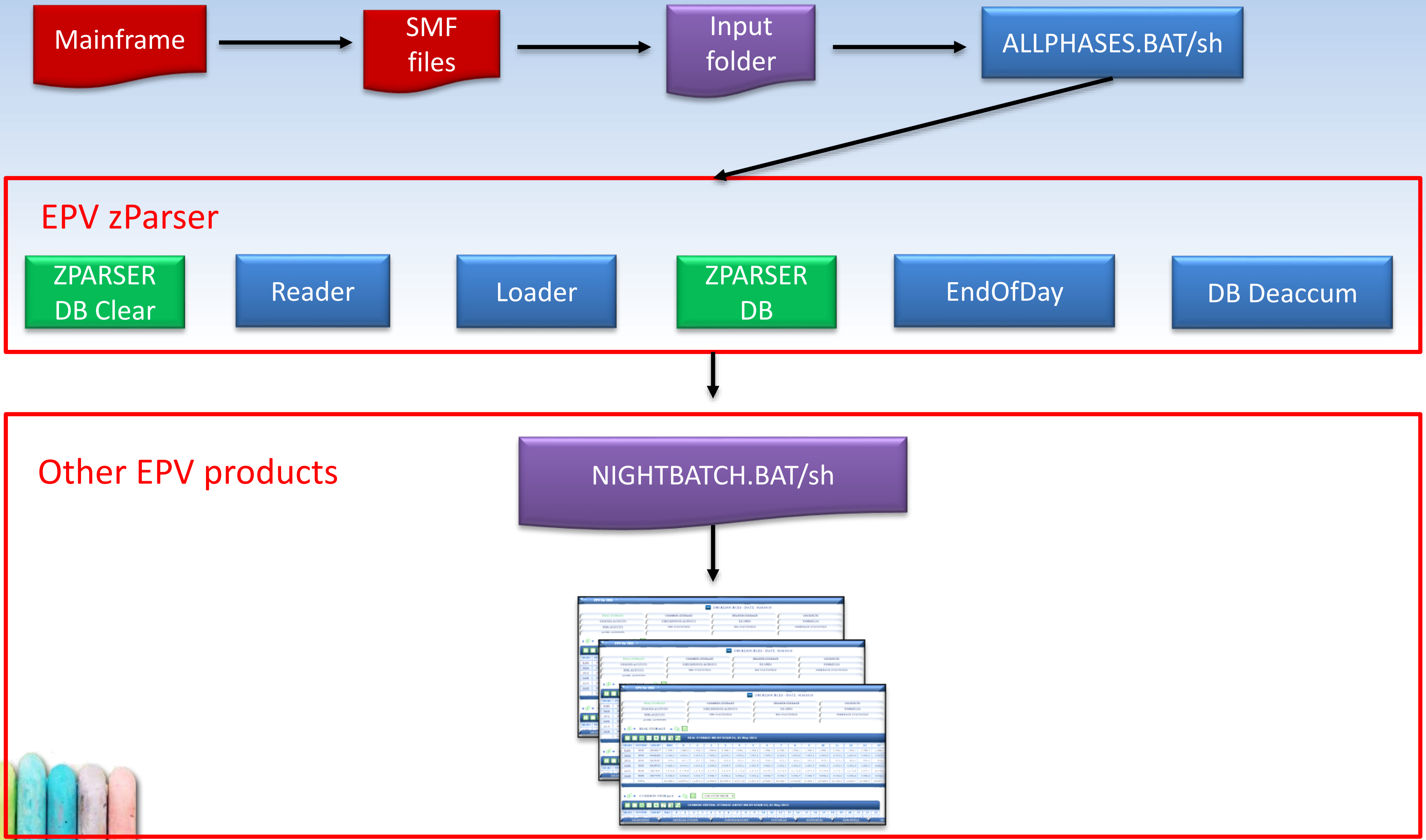
zParser basic processing - Once a day

- How to use it

  ✓ Prepare and send the input files to the EPV server once a day

  ✓ Run the whole EPV process by scheduling, for example at 2 a.m., the ALLPHASES.BAT/sh program in the task scheduler

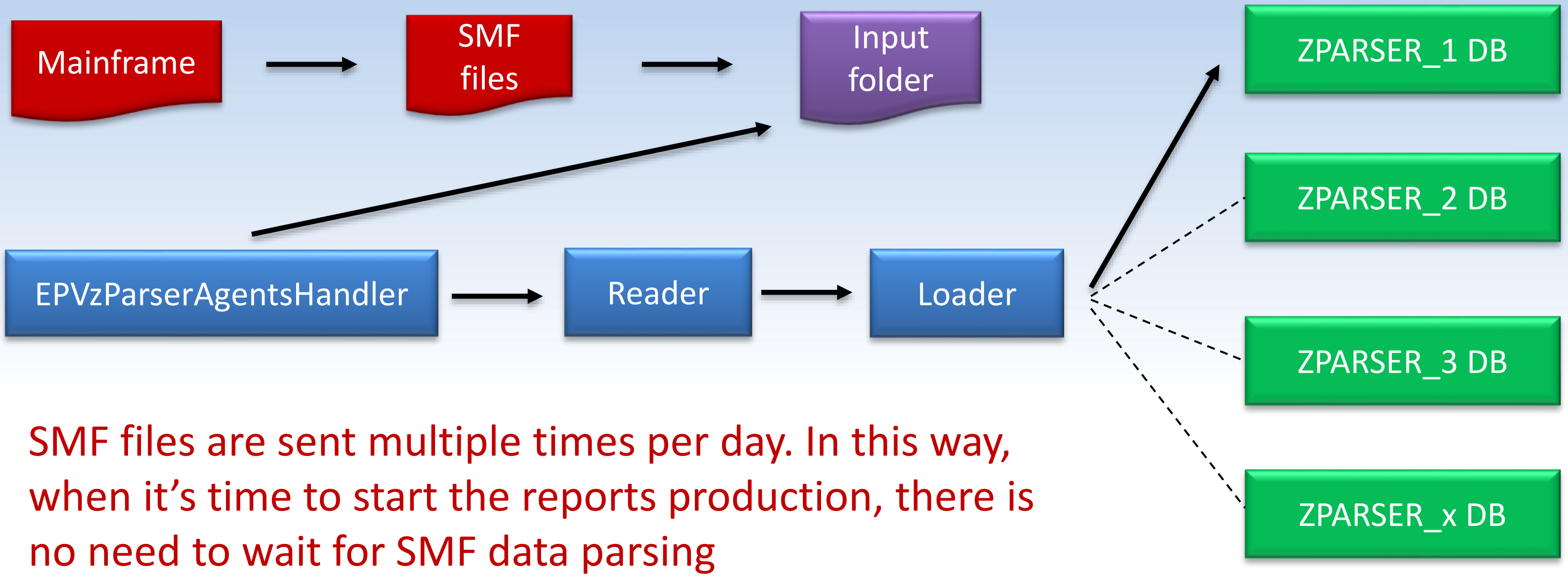
zParser basic processing - Once a day



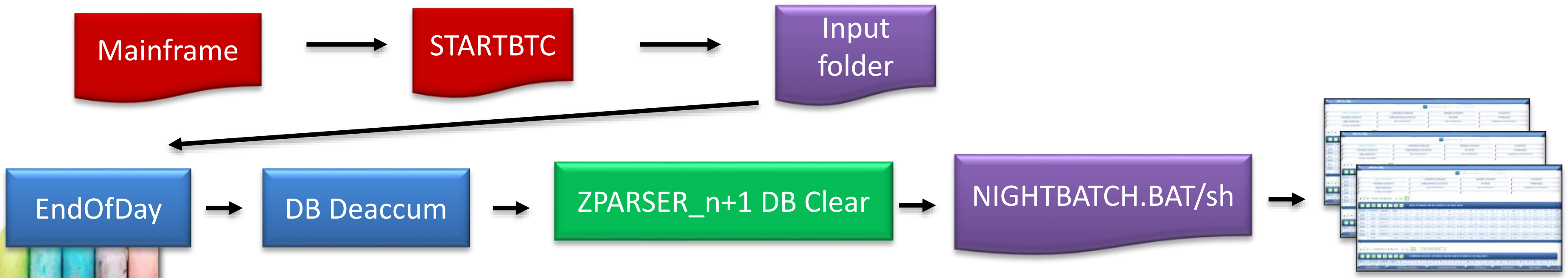
zParser basic processing - Continuous mode

- How to use it
  - ✓ Set the EPVzParserAgentsHandler program to start at EPV server’s boot
  - ✓ Prepare and send SMF data to the EPV server as soon as they’re available during the day
  - ✓ After each file transfer from MF to EPV server, send also an empty file named like the file, plus the _END suffix; this is to tell EPV that the file transfer is complete. Example: filename SMF.TODAY, end-file SMF.TODAY_END
  - ✓ Once all the SMF files of the day are sent, put the STARTBTC flag file in one of the input folders

zParser basic processing - Continuous mode



SMF files are sent multiple times per day. In this way, when it’s time to start the reports production, there is no need to wait for SMF data parsing



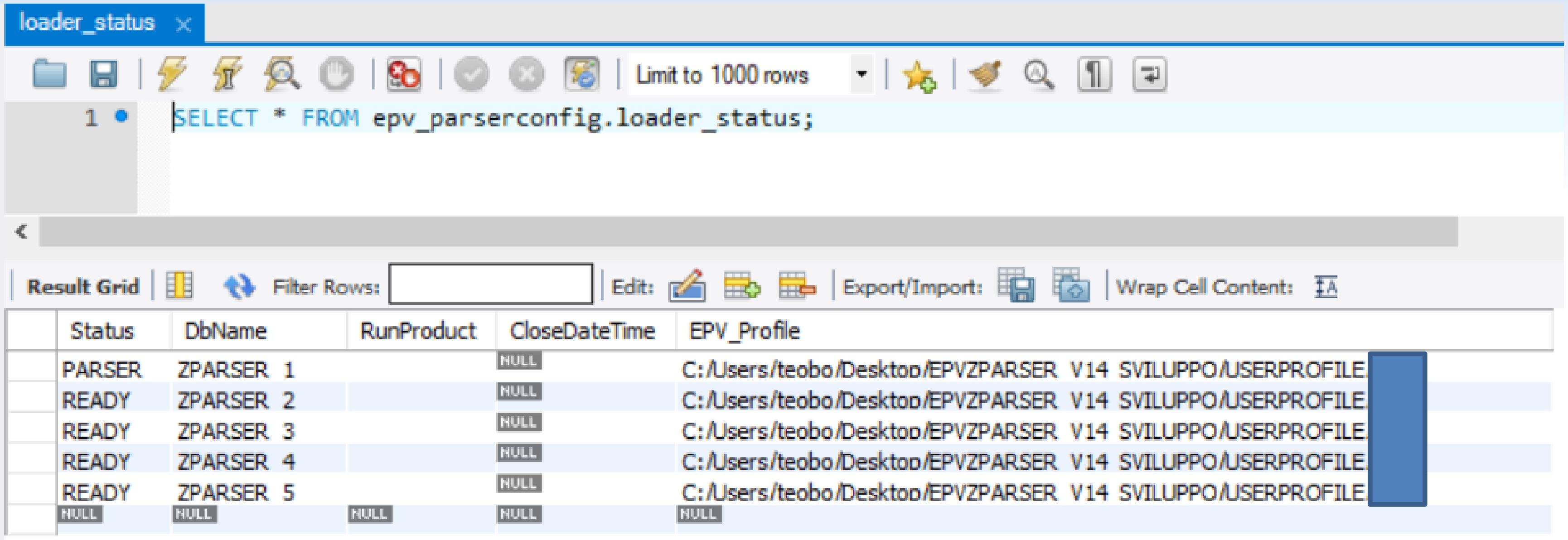
zParser basic processing - Continuous mode

- zParser uses a number of pre allocated DB versions (from a minimum of 2 to a maximum of 99) also called ‘Staging DBs’
- The main purpose of using more than one database, is to permit the continuity of processing by using these DB versions in a cyclical way so that when the last version has been written the cycle restarts from the first one
- The EPV_ParserConfig is a management DB. Its main purpose it to store information about the status of the ‘Staging DBs’
- In specific, these information are contained in the Loader_Status table

zParser basic processing - Continuous mode

- A DB can be in one of the following status:
  - ✓PARSER: it indicates that zParser is writing into this DB
  - ✓CLOSE: it indicates that it is impossible to use this DB because other products (EPV z/OS and/or EPV Db2 and/or EPV MQ and/or zLINUX) are reading it. The product that is using this DB is written in the “Run Product” column
  - ✓READY: it indicates that the data from this DB has already been used so it is ready to be used again by zParser; in this case the value in the “Run Product” column is null

zParser basic processing - Continuous mode



```
SELECT * FROM epv_parserconfig.loader_status;
```

Status	DbName	RunProduct	CloseDateTime	EPV_Profile
PARSER	ZPARSER_1		NULL	C:/Users/teobo/Desktop/EPVZPARSER_V14_SVILUPPO/USERPROFILE
READY	ZPARSER_2		NULL	C:/Users/teobo/Desktop/EPVZPARSER_V14_SVILUPPO/USERPROFILE
READY	ZPARSER_3		NULL	C:/Users/teobo/Desktop/EPVZPARSER_V14_SVILUPPO/USERPROFILE
READY	ZPARSER_4		NULL	C:/Users/teobo/Desktop/EPVZPARSER_V14_SVILUPPO/USERPROFILE
READY	ZPARSER_5		NULL	C:/Users/teobo/Desktop/EPVZPARSER_V14_SVILUPPO/USERPROFILE
NULL	NULL	NULL	NULL	NULL

zParser basic processing - Agents

- When running in Continuous Mode, the EPVzParserAgentsHandler process is the key
- EPVzParserAgentsHandler is a light weight task running in background that manages all the zParser related tasks

zParser basic processing - Agents

- Both in Windows and Linux, it can be installed as System Service. This allows the user to manage this process very easily
- It can also be managed through flag files
- Of course, flag files can also come from mainframe

zParser basic processing - Agents

- EPVzParserAgentsHandler main tasks
  - ✓ Create empty tables in the zParser's database at startup
  - ✓ Check for input data every 100 (default) seconds
  - ✓ Start one zParser pseudo-thread for each input file for a maximum of $Max_threads parallel processes (default is n.processors -1)
  - ✓ Start EPVzParserEndOfDay when the STARTBTC file is being put in one input folder

zParser basic processing - Agents

- Flag files accepted

  ✓ STARTBTC -> start the reports production
  ✓ STARTMAINT -> start the automatic update process
  ✓ STARTPAUSE -> start activities pause period
  ✓ ENDPAUSE -> end activities pause period
  ✓ SHUTDOWN -> close the process
  ✓ RESTART -> restart the process

zParser basic processing - Reader/Loader

- EPVzParser process is split into 2 phases: the reader and the loader
  - ✓ Reader, as the name suggests, reads the input file and transforms the information in a more understandable format inside TXT files
  - ✓ Loader, also as the name suggests, loads the TXT files created by the reader in the zParser's database

zParser basic processing - Reader

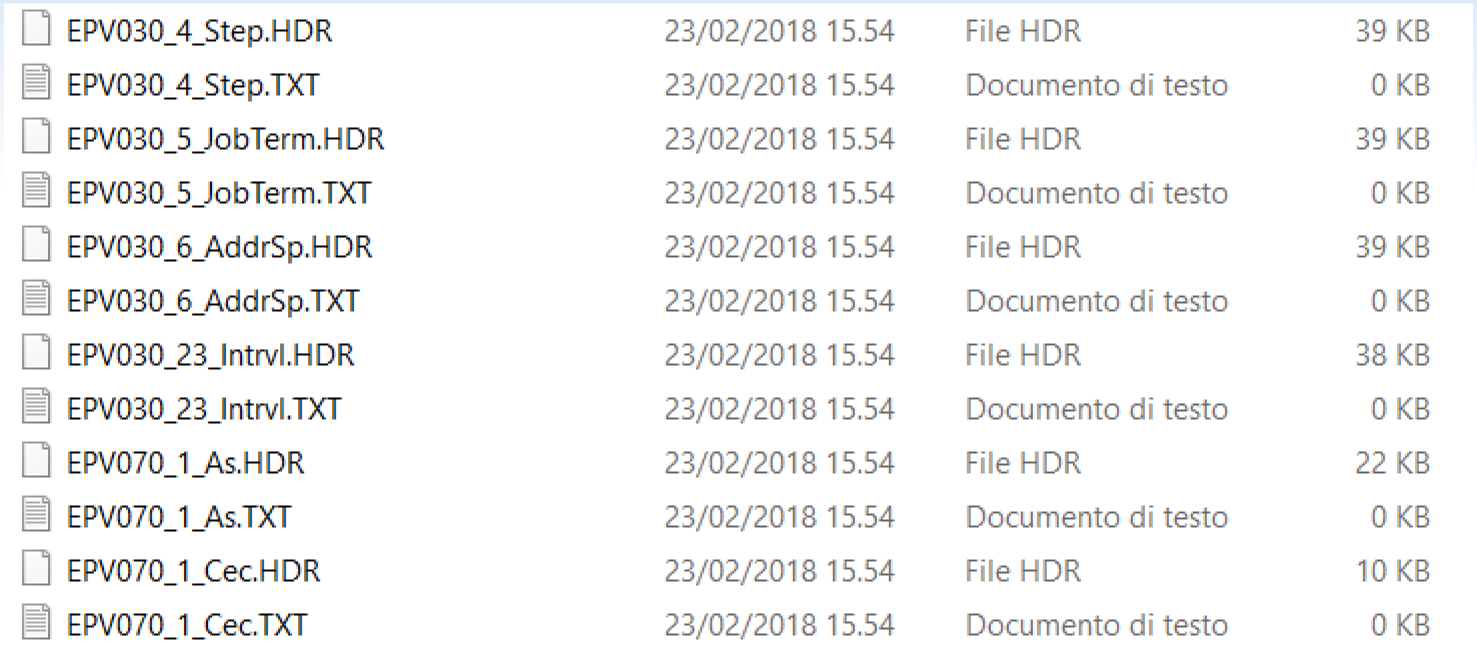
- zParser Reader main tasks
  - ✓ Rename the input file in .Parsing; in this way, it is clear when one file is being read by zParser
  - ✓ Extract information contained in the input file by using hard coded routines
  - ✓ Write in a specific WorkArea for each input file
  - ✓ Write only the information selected by the user and discarding the rest
  - ✓ Move the input file in a temporary folder (Recovery) in case of errors
  - ✓ Avoid to read duplicate files/records (see dedicated chapter later)

zParser basic processing - Reader

- INPUT: SMF/DCO/IMS/etc.
- OUTPUT: for each SMF/DCO/IMS/etc. record type/subtype
  - ✓ One TXT file containing data
  - ✓ One HDR file containing information about data structure (fields types and length)

zParser basic processing - Reader

- Reader's output



EPV030_4_Step.HDR	23/02/2018 15.54	File HDR	39 KB
EPV030_4_Step.TXT	23/02/2018 15.54	Documento di testo	0 KB
EPV030_5_JobTerm.HDR	23/02/2018 15.54	File HDR	39 KB
EPV030_5_JobTerm.TXT	23/02/2018 15.54	Documento di testo	0 KB
EPV030_6_AddrSp.HDR	23/02/2018 15.54	File HDR	39 KB
EPV030_6_AddrSp.TXT	23/02/2018 15.54	Documento di testo	0 KB
EPV030_23_Intrvl.HDR	23/02/2018 15.54	File HDR	38 KB
EPV030_23_Intrvl.TXT	23/02/2018 15.54	Documento di testo	0 KB
EPV070_1_As.HDR	23/02/2018 15.54	File HDR	22 KB
EPV070_1_As.TXT	23/02/2018 15.54	Documento di testo	0 KB
EPV070_1_Cec.HDR	23/02/2018 15.54	File HDR	10 KB
EPV070_1_Cec.TXT	23/02/2018 15.54	Documento di testo	0 KB

zParser basic processing - Loader

- zParser Loader main tasks
  - ✓ Check in which zParser database it has to load data by looking at the Loader_Status table from EPV_ParserConfig database
  - ✓ Create table in the zParser database in case it is missing; in order to do that, it uses the HDR files to generate the ‘create table’ scripts to execute
  - ✓ Load each TXT file in its specific table by using these commands
    - ✓ LOAD DATA INFILE for MySQL/MariaDB
    - ✓ BULK INSERT for MS SQL Server
  - ✓ Move the input file in a temporary folder (Recovery) in case of errors
  - ✓ Delete or move the input file when the loader ends

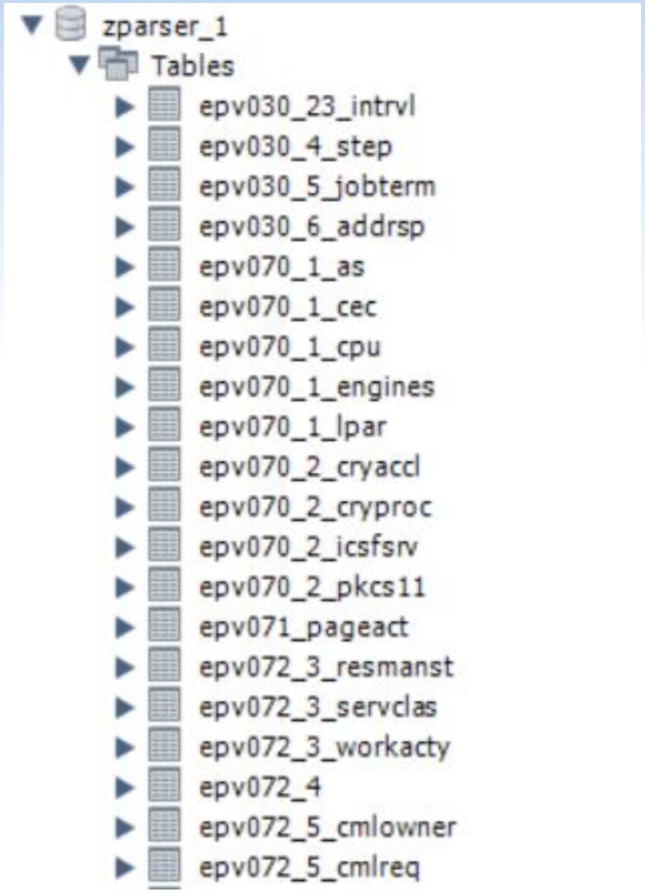
zParser basic processing - Loader

- INPUT: TXT/HDR files created by the Reader

- OUTPUT: one DB table for each TXT/HDR file

zParser basic processing - Loader

- Loader's output



```
▼ zparser_1
  ▼ Tables
    ▶ epv030_23_intrvl
    ▶ epv030_4_step
    ▶ epv030_5_jobterm
    ▶ epv030_6_addrsp
    ▶ epv070_1_as
    ▶ epv070_1_cec
    ▶ epv070_1_cpu
    ▶ epv070_1_engines
    ▶ epv070_1_lpar
    ▶ epv070_2_cryaccl
    ▶ epv070_2_cryproc
    ▶ epv070_2_icsfsrv
    ▶ epv070_2_pkcs11
    ▶ epv071_pageact
    ▶ epv072_3_resmanst
    ▶ epv072_3_servclas
    ▶ epv072_3_workacty
    ▶ epv072_4
    ▶ epv072_5_cmlowner
    ▶ epv072_5_cmlreq
```

zParser basic processing - EndOfDay

- EPVzParserEndOfDay manages all the activities that needs to be done once a day before reports production

- In Once a day it is started by the ALLPHASES script

- In Continuous mode it is started by EPVzParserAgentsHandler as soon as the STARTBTC flag file arrives in one of the input folders

zParser basic processing - EndOfDay

- EPVzParserEndOfDay main tasks
  - ✓ Check if all the tables needed by the other EPV products are there in the zParser's database
  - ✓ Check if there is data inside these tables
  - ✓ Execute EPVzParserDbDeaccum process
  - ✓ In case of Continuous mode, it makes the Staging DBs switch by putting the current zParser DB in CLOSE status and the next one in PARSER status
  - ✓ Drop and re-create tables inside the new zParser's DB that is just being put in PARSER status
  - ✓ Start the POSTZPARSER script (reports production)

zParser basic processing - Db Deaccum

- EPVzParserDbDeaccum process is meant to normalize data that is natively written accumulated. The obtain the new value, the process starts from the last record and then it subtracts the previous record; this behavior is applied to each record in the table
- It creates new tables with _deacc suffix
- It also detects duplicated records and it can automatically remove them (see dedicated chapter)
- It is run just once a day by EPVzParserEndOfDay program

zParser basic processing - Db Deaccum

- It normalizes data for the following tables

  ✓ SMF

    - EPV030_6_AddrSp
    - EPV113_2_HDCap
    - EPV100_0_Stat0
    - EPV100_0_RemLoc
    - EPV100_1_GBPools
    - EPV100_1_Stat1
    - EPV100_1_Bpools

zParser basic processing - Db Deaccum

- It normalizes data for the following tables
  - ✓ZVM

➢D00R01_MRSYTSYP	➢D00R20_MRSYTEPM
➢D00R14_MRSYTXSG	➢D03R04_MRSTOASP
➢D00R16_MRSYTCUP	➢D04R03_MRUSEACT
➢D00R17_MRSYTCUM	➢D04R04_MRUSEINT
➢D00R02_MRSYTPRP	➢D06R03_MRIODDEV
➢D00R05_MRSYTXSP	

zParser advanced processing

zParser advanced processing - Duplicated data controls

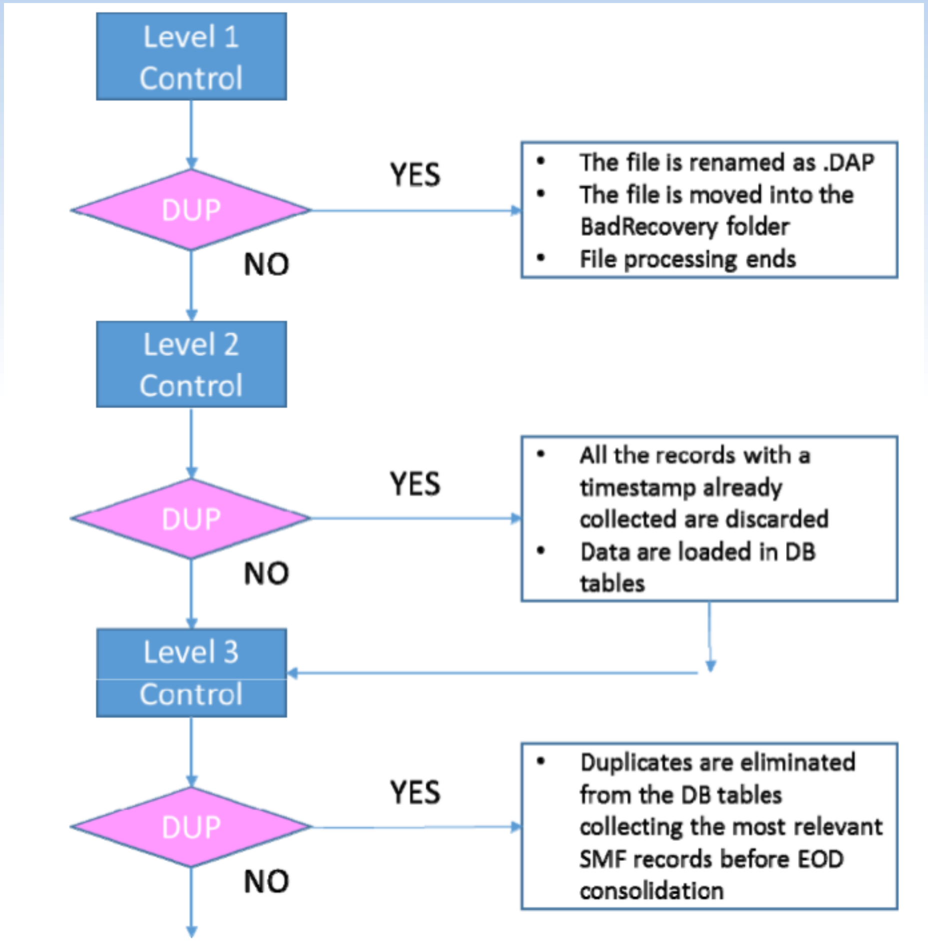
- A major issue with daily collection of large amount of data is the possibility that the same data are loaded more than one time. This is normally due to errors in procedure design or to anomalies in host batch processing so it should be fixed at that level.

- However, zParser provides some controls that the user can activate in order to avoid collecting duplicated data

zParser advanced processing - Duplicated data controls

- EPV zParser provides three levels of control to avoid duplicated data to be collected in the database:
  - ✓ Level 1: to avoid the same input file is loaded more times
  - ✓ Level 2: to avoid that the same input data in different files will be loaded more times
  - ✓ Level 3: to avoid that the same input data even in the same files will be loaded more times

zParser advanced processing - Duplicated data controls



- The file is renamed as .DAP
- The file is moved into the BadRecovery folder
- File processing ends

- All the records with a timestamp already collected are discarded
- Data are loaded in DB tables

- Duplicates are eliminated from the DB tables collecting the most relevant SMF records before EOD consolidation

zParser advanced processing - Duplicated data controls

- **Level 1** control can be activated/deactivated by setting the $CHKDUP_SAME_FILE parameter in the CONFIG.PL member. The default value is Y (control activated). You can set it to N to deactivate this control

- If the Level 1 control is activated, EPV zParser reads the first 112 bytes of each Dcollect, IMS or SMF file. This string is then compared with the contents of a perl DB (firstbytes.db); if that string is already there for the same system-id, the file is marked as DAP (Dump Already Parsed) and then moved to the BadRecovery folder

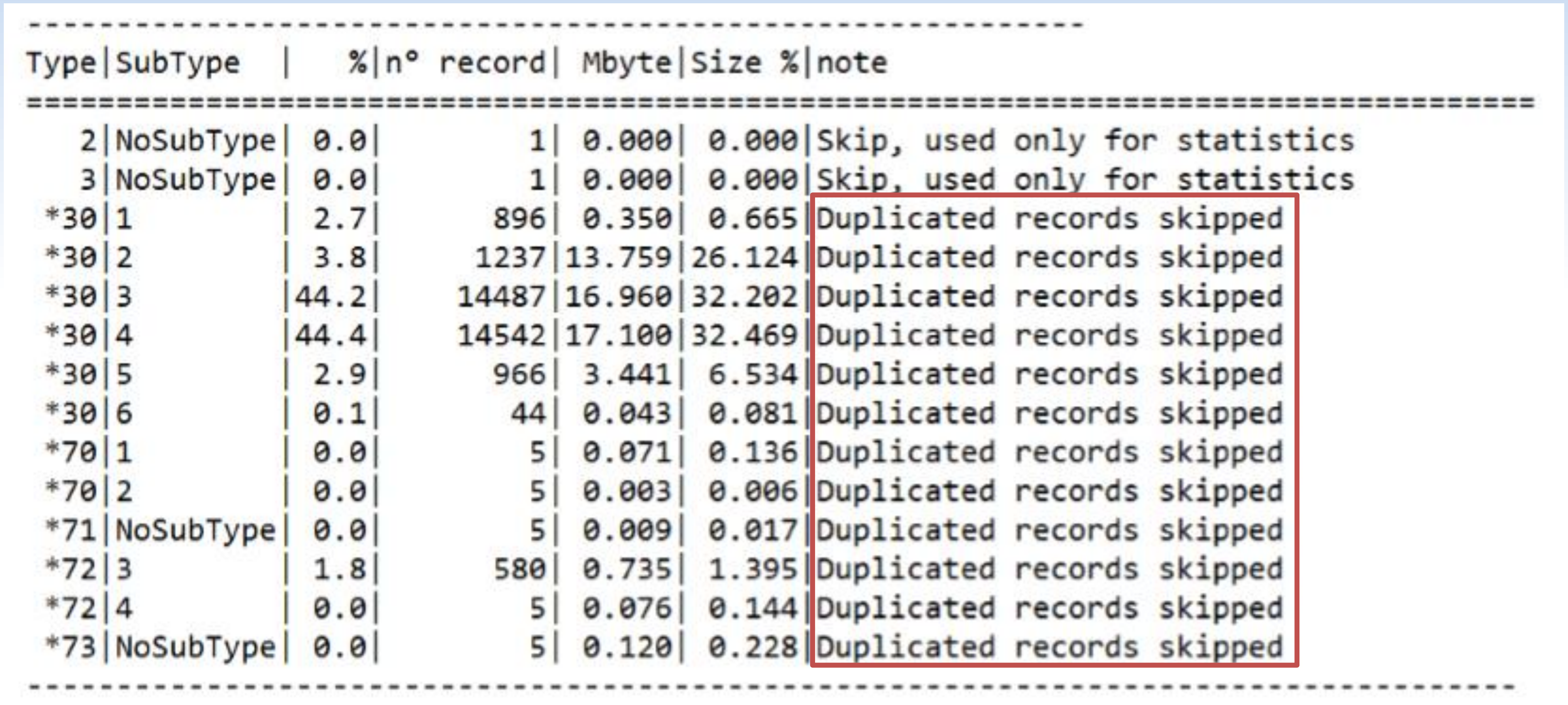
zParser advanced processing - Duplicated data controls

- **Level 2** control can be activated/deactivated by setting the $CHKDUP_MORE_FILES parameter in the CONFIG.PL member. The default value is N (control deactivated). You can set it to Y to activate this control.
- If the Level 2 control is activated, for each input file and for each system-id and record type inside the input file, the highest and the lowest timestamps are stored in a control file
- At the beginning of each file processing, this control file is being loaded into an indexed table on a memory resident DB and accessed through the SQL standard language

zParser advanced processing - Duplicated data controls

- For each record that is read, the following query is executed: SELECT RECTYPE FROM ARCHTIME WHERE SYS='$system' AND RECTYPE='$rectype' AND ENDINTRV >= '$intrvl' AND STARTINTRV <= '$intrvl’. $system, $rectype and $intrvl are the system, record type and timestamp of the record. If the query returns at least one record, zParser assumes that this record was already loaded and, so, it discards the record
- When duplicated records are discarded, “Duplicated records skipped" is written in the note column of the record summary report provided in the zParser log for each processed file

zParser advanced processing - Duplicated data controls



```
-----------------------------------------------------------
Type|SubType  |   %|n° record| Mbyte|Size %|note
=====================================================================================
   2|NoSubType| 0.0|         1| 0.000| 0.000|Skip, used only for statistics
   3|NoSubType| 0.0|         1| 0.000| 0.000|Skip, used only for statistics
 *30|1        | 2.7|       896| 0.350| 0.665|Duplicated records skipped
 *30|2        | 3.8|      1237|13.759|26.124|Duplicated records skipped
 *30|3        |44.2|     14487|16.960|32.202|Duplicated records skipped
 *30|4        |44.4|     14542|17.100|32.469|Duplicated records skipped
 *30|5        | 2.9|       966| 3.441| 6.534|Duplicated records skipped
 *30|6        | 0.1|        44| 0.043| 0.081|Duplicated records skipped
 *70|1        | 0.0|         5| 0.071| 0.136|Duplicated records skipped
 *70|2        | 0.0|         5| 0.003| 0.006|Duplicated records skipped
 *71|NoSubType| 0.0|         5| 0.009| 0.017|Duplicated records skipped
 *72|3        | 1.8|       580| 0.735| 1.395|Duplicated records skipped
 *72|4        | 0.0|         5| 0.076| 0.144|Duplicated records skipped
 *73|NoSubType| 0.0|         5| 0.120| 0.228|Duplicated records skipped
------------------------------------------------------------------------------------
```

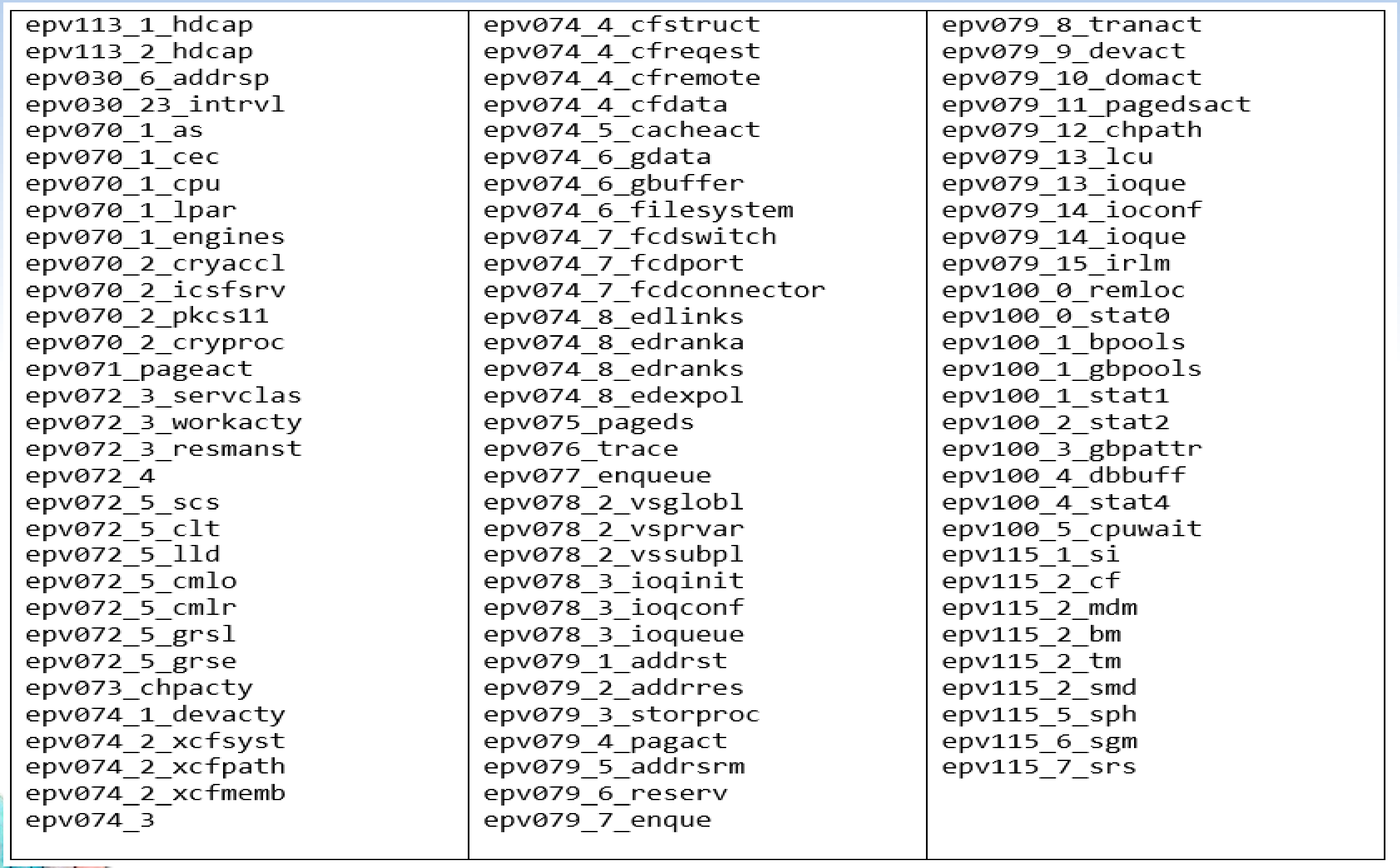
zParser advanced processing - Duplicated data controls

- **Level 3** control can be activated/deactivated by setting the $CHKDUP_INSIDE_ONEFILE parameter in the CONFIG.PL member. The default value is N (control deactivated). You can set it to S or Y to activate this control
- Whatever value you set in the above parameter, at EPVzParserDbDeaccum beginning, a check of duplicate rows is executed on the following tables:
  - ✓ epv070_1_cpu
  - ✓ epv100_0_stat0
  - ✓ epv115_1_si

zParser advanced processing - Duplicated data controls

- If duplicated records are found in the above tables, the behaviour depends on the $CHKDUP_INSIDE_ONEFILE setting:
  - ✓ with N a warning message is written in the log and process continues
  - ✓ with S an error message is written in the log and process stops
  - ✓ with Y duplicated data are automatically removed from all the following tables

zParser advanced processing - Duplicated data controls



epv113_1_hdcap	epv074_4_cfstruct	epv079_8_tranact
epv113_2_hdcap	epv074_4_cfreqest	epv079_9_devact
epv030_6_addrsp	epv074_4_cfremote	epv079_10_domact
epv030_23_intrvl	epv074_4_cfdata	epv079_11_pagedsact
epv070_1_as	epv074_5_cacheact	epv079_12_chpath
epv070_1_cec	epv074_6_gdata	epv079_13_lcu
epv070_1_cpu	epv074_6_gbuffer	epv079_13_ioque
epv070_1_lpar	epv074_6_filesystem	epv079_14_ioconf
epv070_1_engines	epv074_7_fcdswitch	epv079_14_ioque
epv070_2_cryaccl	epv074_7_fcdport	epv079_15_irlm
epv070_2_icsfsrv	epv074_7_fcdconnector	epv100_0_remloc
epv070_2_pkcs11	epv074_8_edlinks	epv100_0_stat0
epv070_2_cryproc	epv074_8_edranka	epv100_1_bpools
epv071_pageact	epv074_8_edranks	epv100_1_gbpools
epv072_3_servclas	epv074_8_edexpol	epv100_1_stat1
epv072_3_workacty	epv075_pageds	epv100_2_stat2
epv072_3_resmanst	epv076_trace	epv100_3_gbpattr
epv072_4	epv077_enqueue	epv100_4_dbbuff
epv072_5_scs	epv078_2_vsglobl	epv100_4_stat4
epv072_5_clt	epv078_2_vsprvar	epv100_5_cpuwait
epv072_5_lld	epv078_2_vssubpl	epv115_1_si
epv072_5_cmlo	epv078_3_ioqinit	epv115_2_cf
epv072_5_cmlr	epv078_3_ioqconf	epv115_2_mdm
epv072_5_grsl	epv078_3_ioqueue	epv115_2_bm
epv072_5_grse	epv079_1_addrst	epv115_2_tm
epv073_chpacty	epv079_2_addrres	epv115_2_smd
epv074_1_devacty	epv079_3_storproc	epv115_5_sph
epv074_2_xcfsyst	epv079_4_pagact	epv115_6_sgm
epv074_2_xcfpath	epv079_5_addrsrm	epv115_7_srs
epv074_2_xcfmemb	epv079_6_reserv	
epv074_3	epv079_7_enque	

zParser advanced processing - User Exits

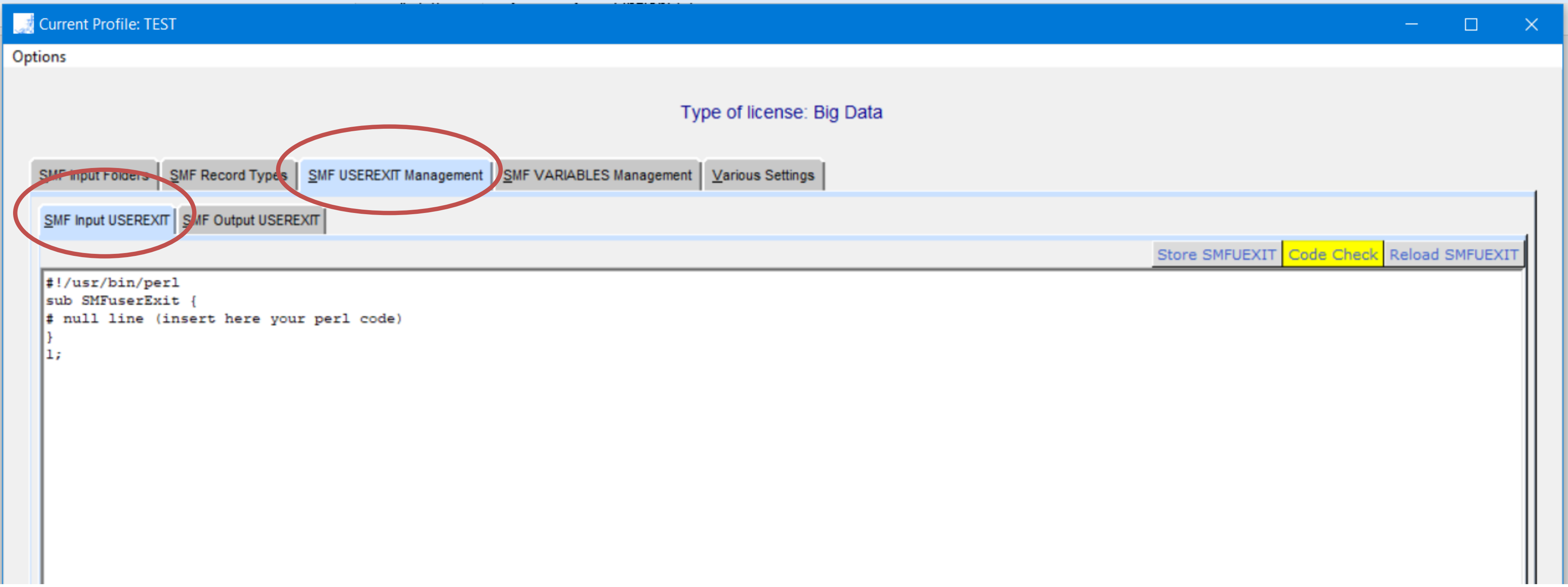
- For each Input Engine it is possible to define two levels of user exists:
  - ✓ Input, that are executed during the reading of input data
  - ✓ Output, that are executed during the production of the TXT files

zParser advanced processing - Input User Exit

- Users can write here their own Perl code to filter some data during the reading of the input files
- EPV Support can help them because they're familiar with Perl ☺
- This kind of user exit is applied to all record types/subtypes

zParser advanced processing - Input User Exit

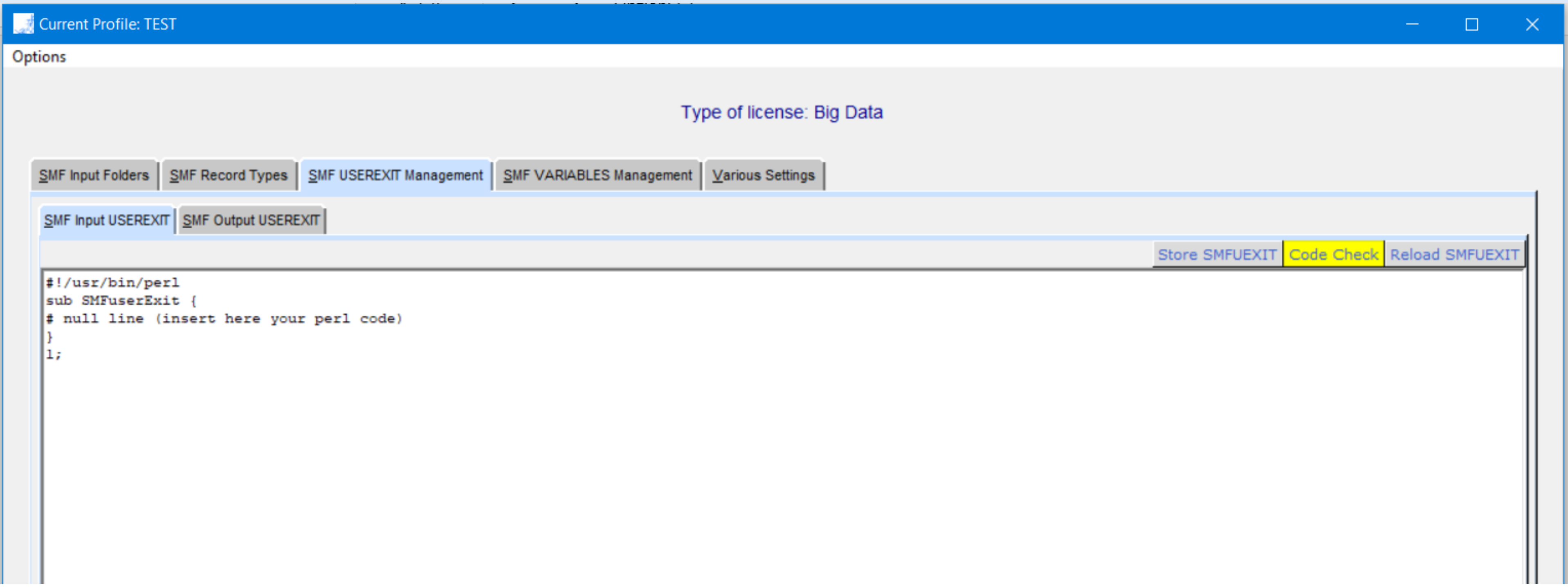
- To customize, open one Input Engine section and click on the SMF USEREXIT Management



```
#!/usr/bin/perl
sub SMFuserExit {
# null line (insert here your perl code)
}
1;
```

zParser advanced processing - Input User Exit

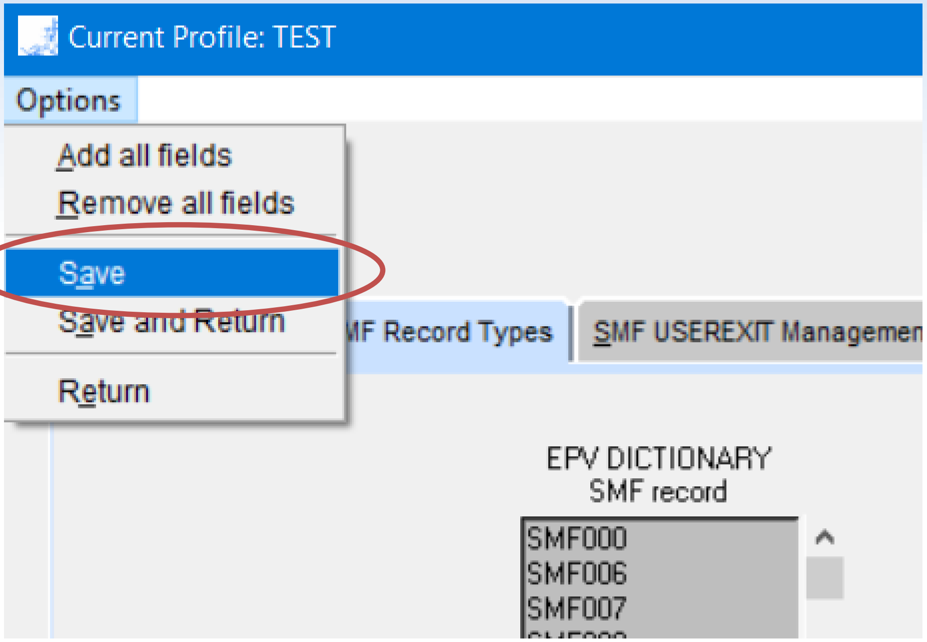
- Once written the Perl code, click on Code Check to make an analysis on the syntax and then Store USEREXIT to confirm



```
#!/usr/bin/perl
sub SMFuserExit {
# null line (insert here your perl code)
}
1;
```

zParser advanced processing - Input User Exit

- Once finished, select Options, then Save

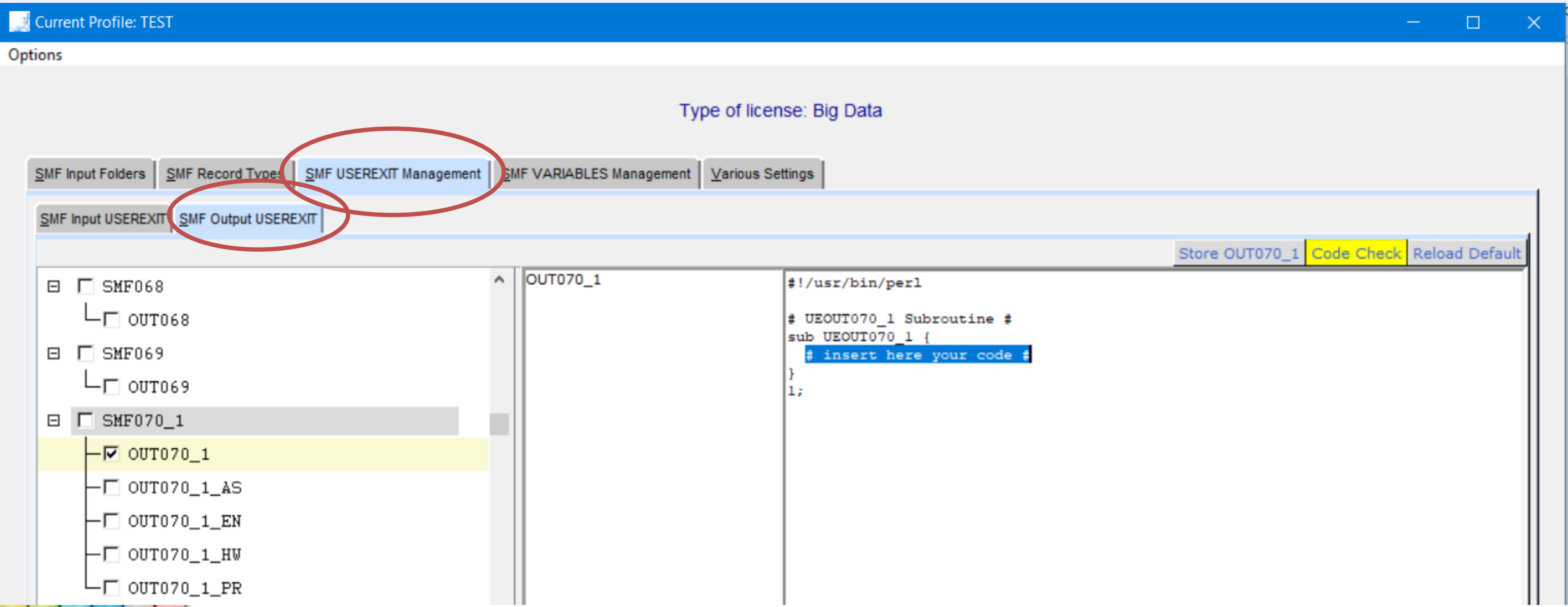


zParser advanced processing - Output User Exit

- Users can write here their own Perl code to filter some data during the writing of the TXT files
- EPV Support can help them
- This kind of user exit is applied to specific record types/subtypes selected by the user

zParser advanced processing - Output User Exit

- To customize, open one Input Engine section and click on the SMF USEREXIT Management, then SMF Output USEREXIT
- Select the record type in which you want to apply the user exit

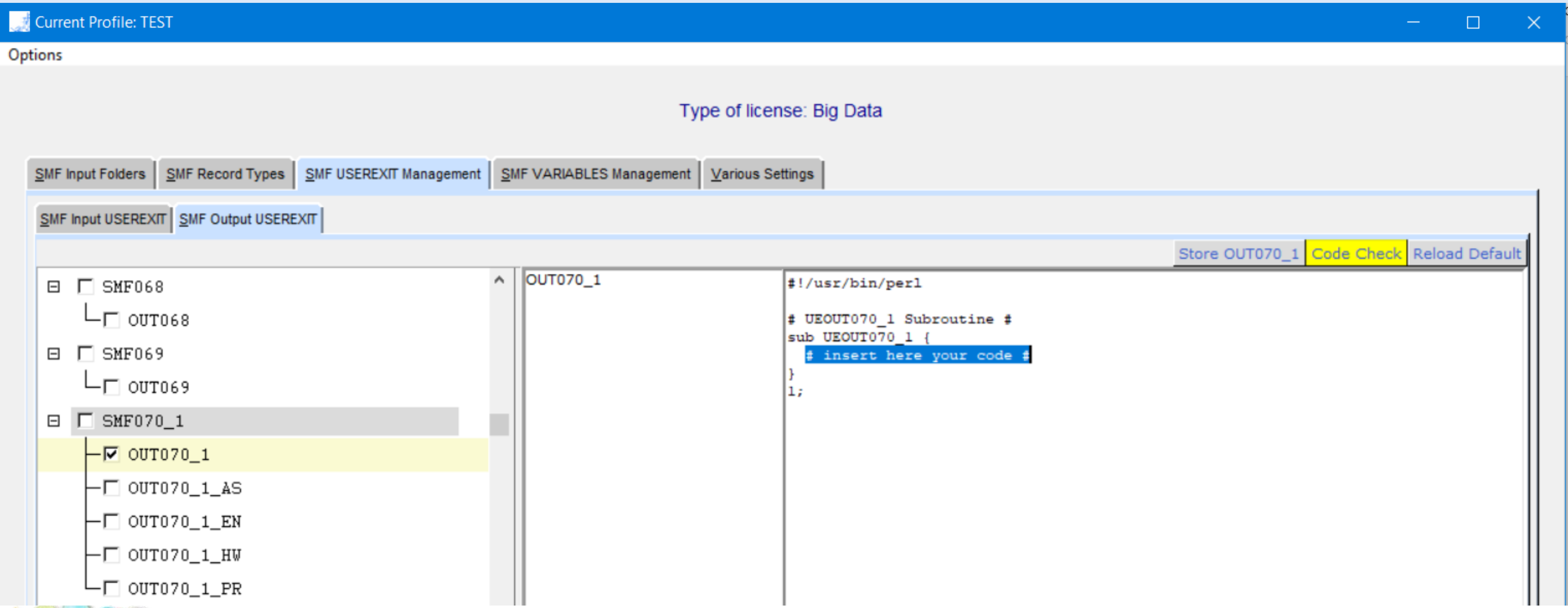


```
#!/usr/bin/perl

# UEOUT070_1 Subroutine #
sub UEOUT070_1 {
  # insert here your code #
}
1;
```

zParser advanced processing - Output User Exit

- Once written the Perl code, click on Code Check to make an analysis on the syntax and then Store USEREXIT to confirm

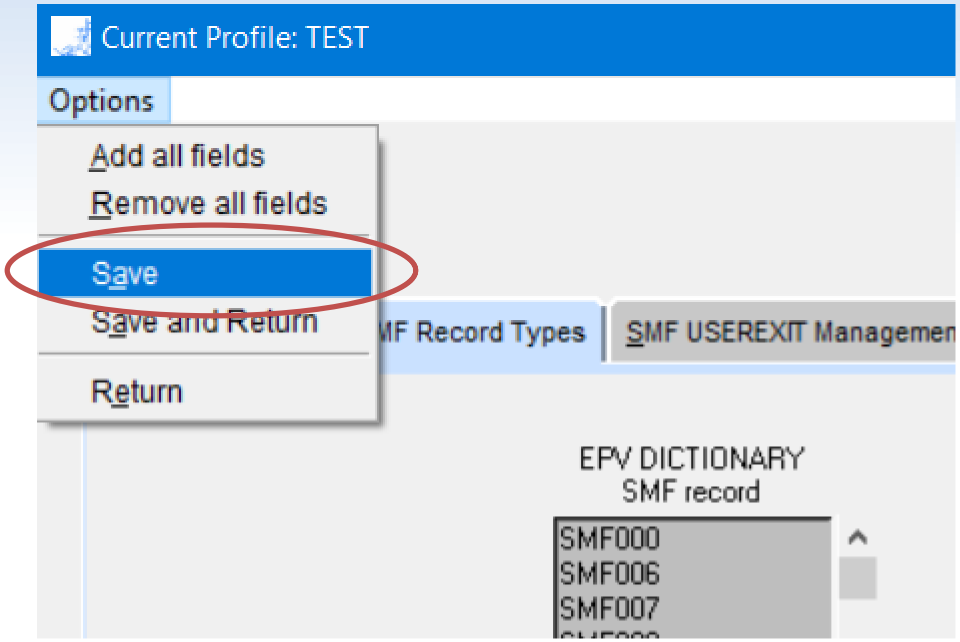


```
#!/usr/bin/perl

# UEOUT070_1 Subroutine #
sub UEOUT070_1 {
  # insert here your code #
}
1;
```

zParser advanced processing - Output User Exit

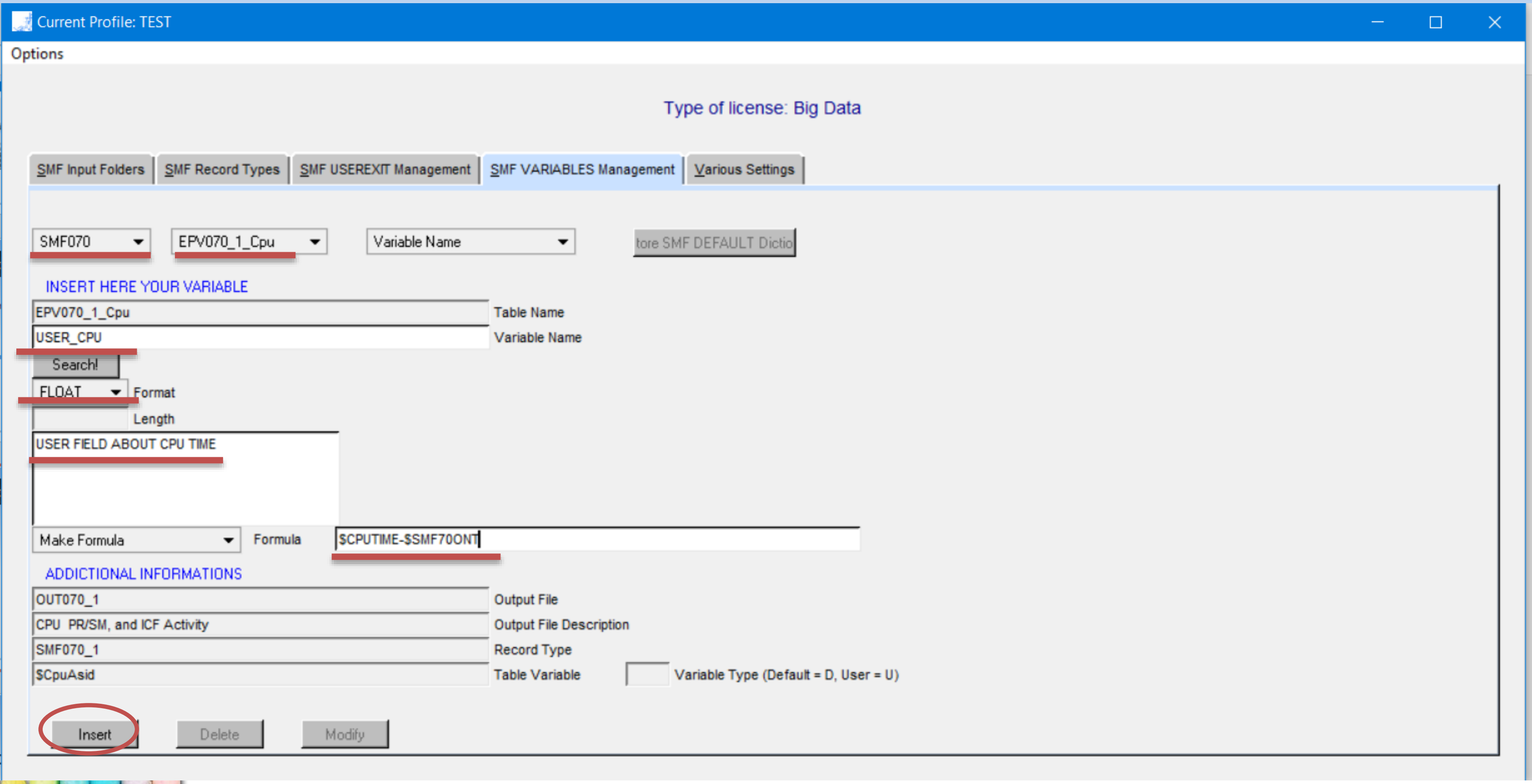
- Once finished, select Options, then Save



zParser advanced processing – User Fields

- For each Input Engine it is possible to define user fields in each zParser table
- To customize, open one Input Engine section and click on the SMF VARIABLES Management

zParser advanced processing – User Fields



Current Profile: TEST

Options

Type of license: Big Data

SMF Input Folders | SMF Record Types | SMF USEREXIT Management | SMF VARIABLES Management | Various Settings

SMF070 | EPV070_1_Cpu | Variable Name | tore SMF DEFAULT Dictio

INSERT HERE YOUR VARIABLE

EPV070_1_Cpu	Table Name
USER_CPU	Variable Name
FLOAT	Format
	Length

USER FIELD ABOUT CPU TIME

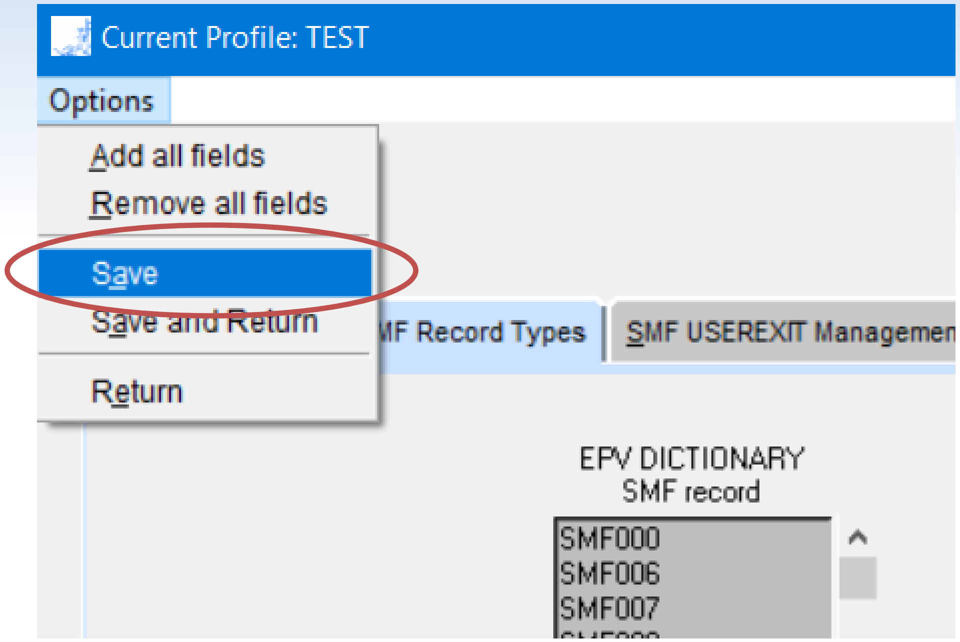
Make Formula — Formula: $CPUTIME-$SMF70ONT

ADDICTIONAL INFORMATIONS

OUT070_1	Output File
CPU PR/SM, and ICF Activity	Output File Description
SMF070_1	Record Type
$CpuAsid	Table Variable
	Variable Type (Default = D, User = U)

zParser advanced processing - User Fields

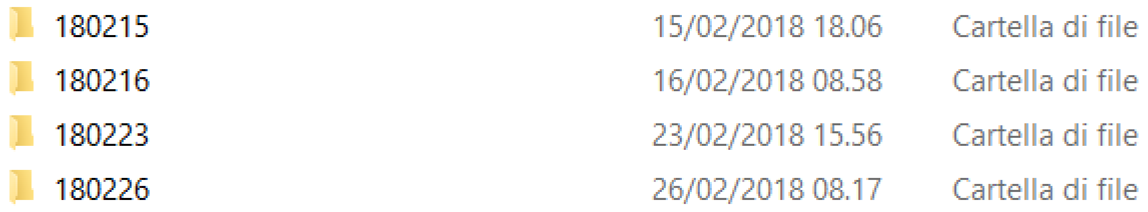
- Once finished, select Options, then Save



Processing controls

Processing controls - Logging

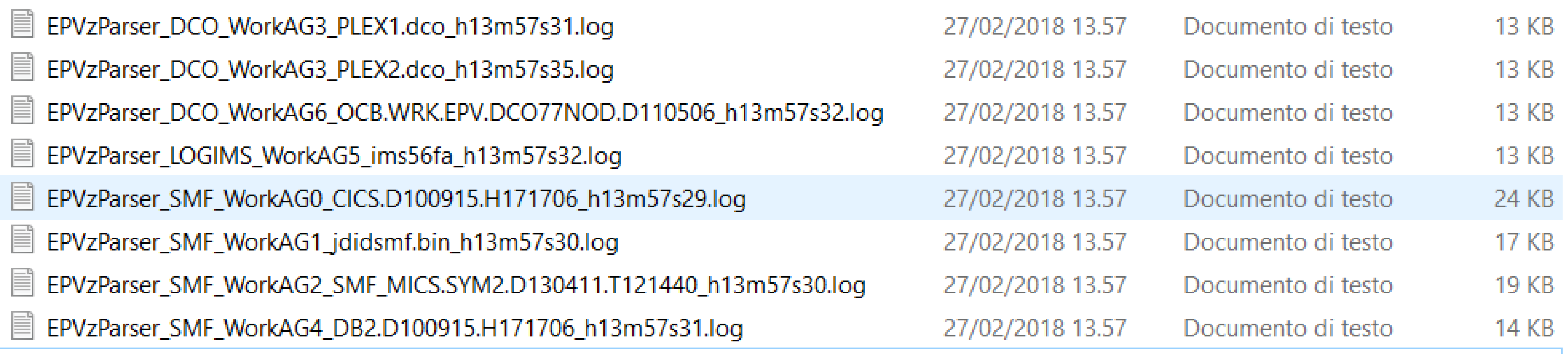
- Every zParser action is being written to a specific log inside EPVROOT/USERPROFILE/*PROFILENAME*/LOGS/EPVZPARSER_LOGS
- Inside this folder, one subfolder for each day is created. In this example: 15th, 16th, 23rd, 26th of February 2018



```
180215    15/02/2018 18.06    Cartella di file
180216    16/02/2018 08.58    Cartella di file
180223    23/02/2018 15.56    Cartella di file
180226    26/02/2018 08.17    Cartella di file
```

Processing controls - Logging

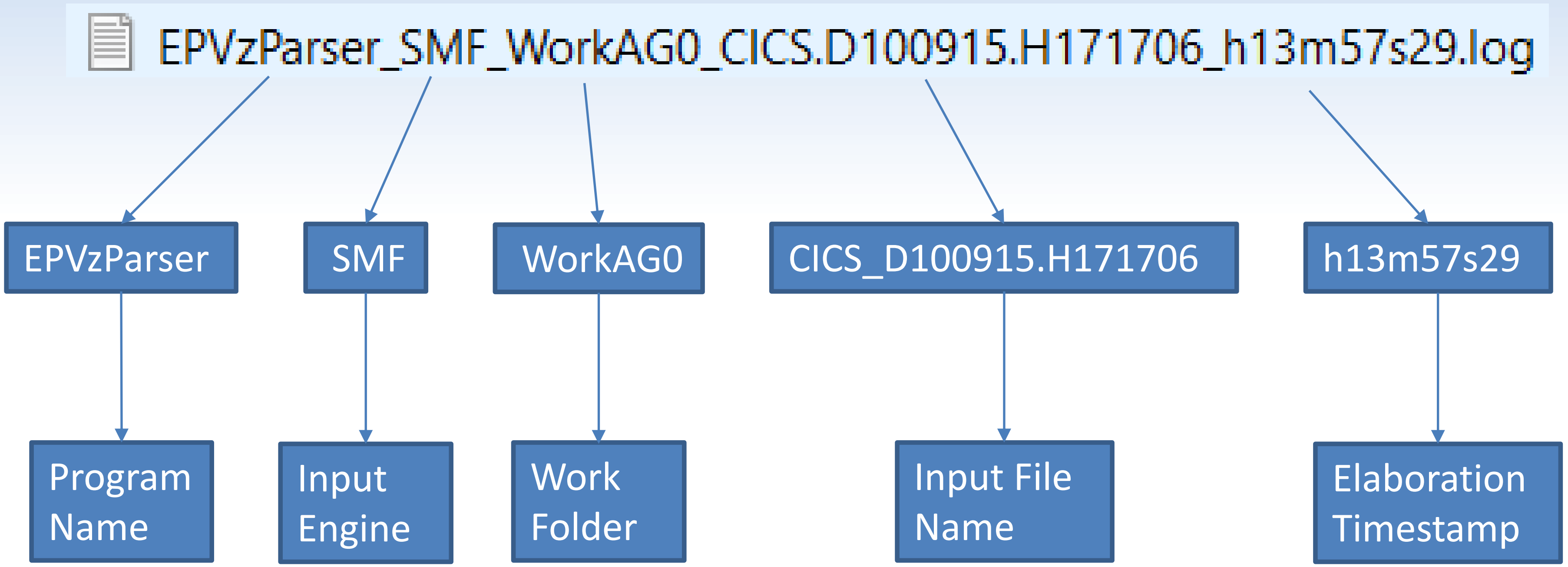
- For each input file that is elaborated, one log is created



EPVzParser_DCO_WorkAG3_PLEX1.dco_h13m57s31.log	27/02/2018 13.57	Documento di testo	13 KB
EPVzParser_DCO_WorkAG3_PLEX2.dco_h13m57s35.log	27/02/2018 13.57	Documento di testo	13 KB
EPVzParser_DCO_WorkAG6_OCB.WRK.EPV.DCO77NOD.D110506_h13m57s32.log	27/02/2018 13.57	Documento di testo	13 KB
EPVzParser_LOGIMS_WorkAG5_ims56fa_h13m57s32.log	27/02/2018 13.57	Documento di testo	13 KB
EPVzParser_SMF_WorkAG0_CICS.D100915.H171706_h13m57s29.log	27/02/2018 13.57	Documento di testo	24 KB
EPVzParser_SMF_WorkAG1_jdidsmf.bin_h13m57s30.log	27/02/2018 13.57	Documento di testo	17 KB
EPVzParser_SMF_WorkAG2_SMF_MICS.SYM2.D130411.T121440_h13m57s30.log	27/02/2018 13.57	Documento di testo	19 KB
EPVzParser_SMF_WorkAG4_DB2.D100915.H171706_h13m57s31.log	27/02/2018 13.57	Documento di testo	14 KB

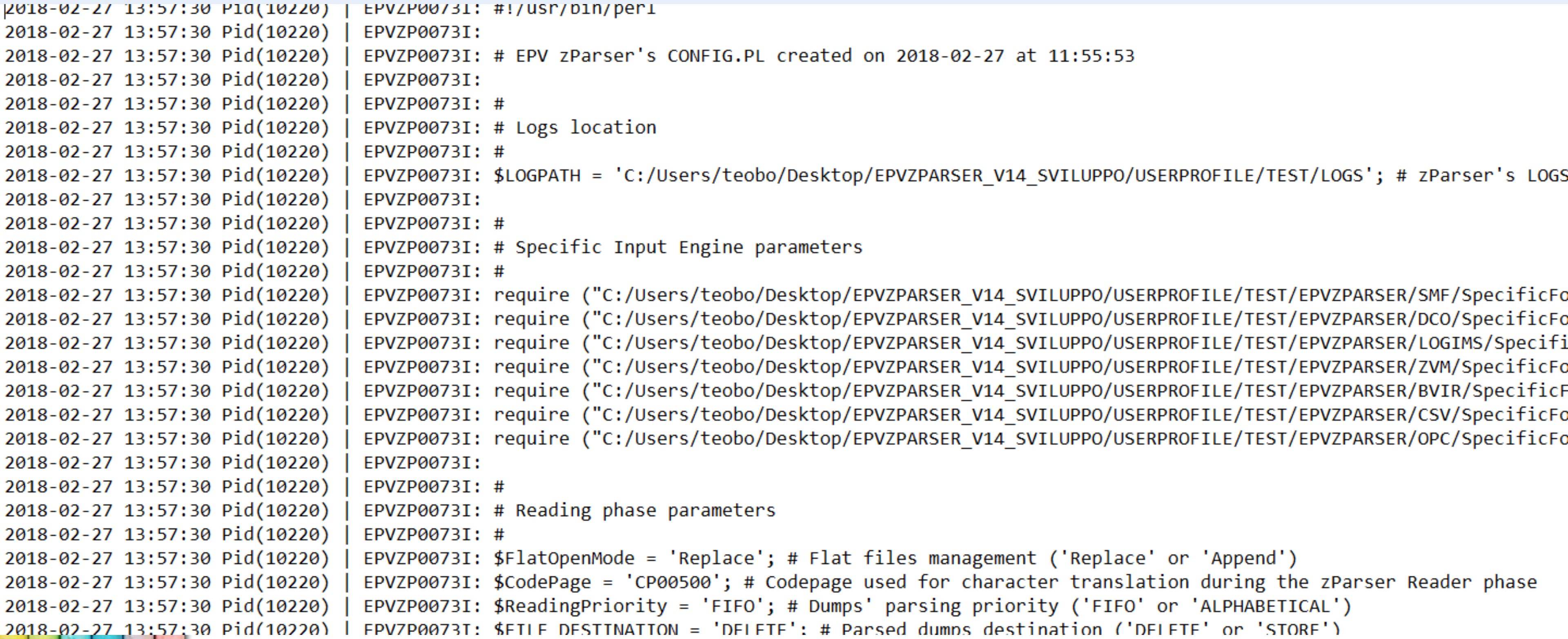
Processing controls - Logging

- Log name is composed by different parts divided by _



Processing controls - Logging

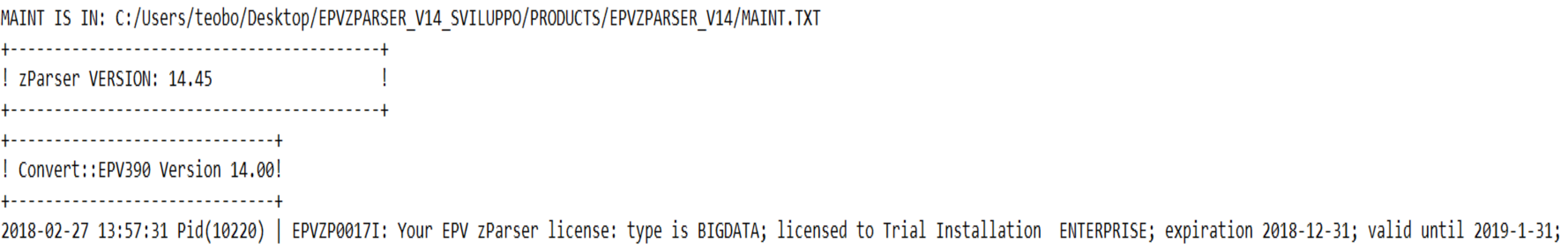
- These log files contain various important information about the zParser activity. First, all the current parameters are listed



```
2018-02-27 13:57:30 Pid(10220) | EPVZP0073I: #!/usr/bin/perl
2018-02-27 13:57:30 Pid(10220) | EPVZP0073I:
2018-02-27 13:57:30 Pid(10220) | EPVZP0073I: # EPV zParser's CONFIG.PL created on 2018-02-27 at 11:55:53
2018-02-27 13:57:30 Pid(10220) | EPVZP0073I:
2018-02-27 13:57:30 Pid(10220) | EPVZP0073I: #
2018-02-27 13:57:30 Pid(10220) | EPVZP0073I: # Logs location
2018-02-27 13:57:30 Pid(10220) | EPVZP0073I: #
2018-02-27 13:57:30 Pid(10220) | EPVZP0073I: $LOGPATH = 'C:/Users/teobo/Desktop/EPVZPARSER_V14_SVILUPPO/USERPROFILE/TEST/LOGS'; # zParser's LOGS
2018-02-27 13:57:30 Pid(10220) | EPVZP0073I:
2018-02-27 13:57:30 Pid(10220) | EPVZP0073I: #
2018-02-27 13:57:30 Pid(10220) | EPVZP0073I: # Specific Input Engine parameters
2018-02-27 13:57:30 Pid(10220) | EPVZP0073I: #
2018-02-27 13:57:30 Pid(10220) | EPVZP0073I: require ("C:/Users/teobo/Desktop/EPVZPARSER_V14_SVILUPPO/USERPROFILE/TEST/EPVZPARSER/SMF/SpecificFo
2018-02-27 13:57:30 Pid(10220) | EPVZP0073I: require ("C:/Users/teobo/Desktop/EPVZPARSER_V14_SVILUPPO/USERPROFILE/TEST/EPVZPARSER/DCO/SpecificFo
2018-02-27 13:57:30 Pid(10220) | EPVZP0073I: require ("C:/Users/teobo/Desktop/EPVZPARSER_V14_SVILUPPO/USERPROFILE/TEST/EPVZPARSER/LOGIMS/Specifi
2018-02-27 13:57:30 Pid(10220) | EPVZP0073I: require ("C:/Users/teobo/Desktop/EPVZPARSER_V14_SVILUPPO/USERPROFILE/TEST/EPVZPARSER/ZVM/SpecificFo
2018-02-27 13:57:30 Pid(10220) | EPVZP0073I: require ("C:/Users/teobo/Desktop/EPVZPARSER_V14_SVILUPPO/USERPROFILE/TEST/EPVZPARSER/BVIR/SpecificF
2018-02-27 13:57:30 Pid(10220) | EPVZP0073I: require ("C:/Users/teobo/Desktop/EPVZPARSER_V14_SVILUPPO/USERPROFILE/TEST/EPVZPARSER/CSV/SpecificFo
2018-02-27 13:57:30 Pid(10220) | EPVZP0073I: require ("C:/Users/teobo/Desktop/EPVZPARSER_V14_SVILUPPO/USERPROFILE/TEST/EPVZPARSER/OPC/SpecificFo
2018-02-27 13:57:30 Pid(10220) | EPVZP0073I:
2018-02-27 13:57:30 Pid(10220) | EPVZP0073I: #
2018-02-27 13:57:30 Pid(10220) | EPVZP0073I: # Reading phase parameters
2018-02-27 13:57:30 Pid(10220) | EPVZP0073I: #
2018-02-27 13:57:30 Pid(10220) | EPVZP0073I: $FlatOpenMode = 'Replace'; # Flat files management ('Replace' or 'Append')
2018-02-27 13:57:30 Pid(10220) | EPVZP0073I: $CodePage = 'CP00500'; # Codepage used for character translation during the zParser Reader phase
2018-02-27 13:57:30 Pid(10220) | EPVZP0073I: $ReadingPriority = 'FIFO'; # Dumps' parsing priority ('FIFO' or 'ALPHABETICAL')
2018-02-27 13:57:30 Pid(10220) | EPVZP0073I: $FILE_DESTINATION = 'DELETE'; # Parsed dumps destination ('DELETE' or 'STORE')
```

Processing controls - Logging

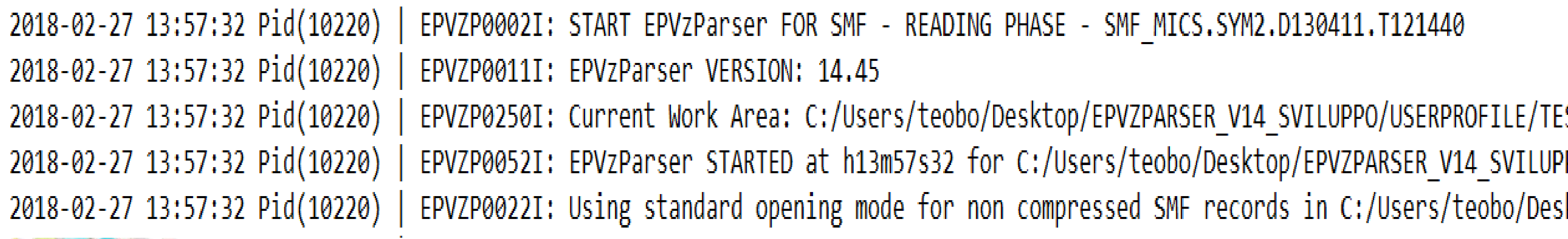
- Scrolling down in the log…
  - ✓Maint file location
  - ✓Product version
  - ✓Convert::EPV390 version (parsing engine)
  - ✓License information like Type, Customer Name, Expiration..



```
MAINT IS IN: C:/Users/teobo/Desktop/EPVZPARSER_V14_SVILUPPO/PRODUCTS/EPVZPARSER_V14/MAINT.TXT
+-----------------------------------------+
! zParser VERSION: 14.45                  !
+-----------------------------------------+
+-----------------------------+
! Convert::EPV390 Version 14.00!
+-----------------------------+
2018-02-27 13:57:31 Pid(10220) | EPVZP0017I: Your EPV zParser license: type is BIGDATA; licensed to Trial Installation  ENTERPRISE; expiration 2018-12-31; valid until 2019-1-31;
```

Processing controls - Logging

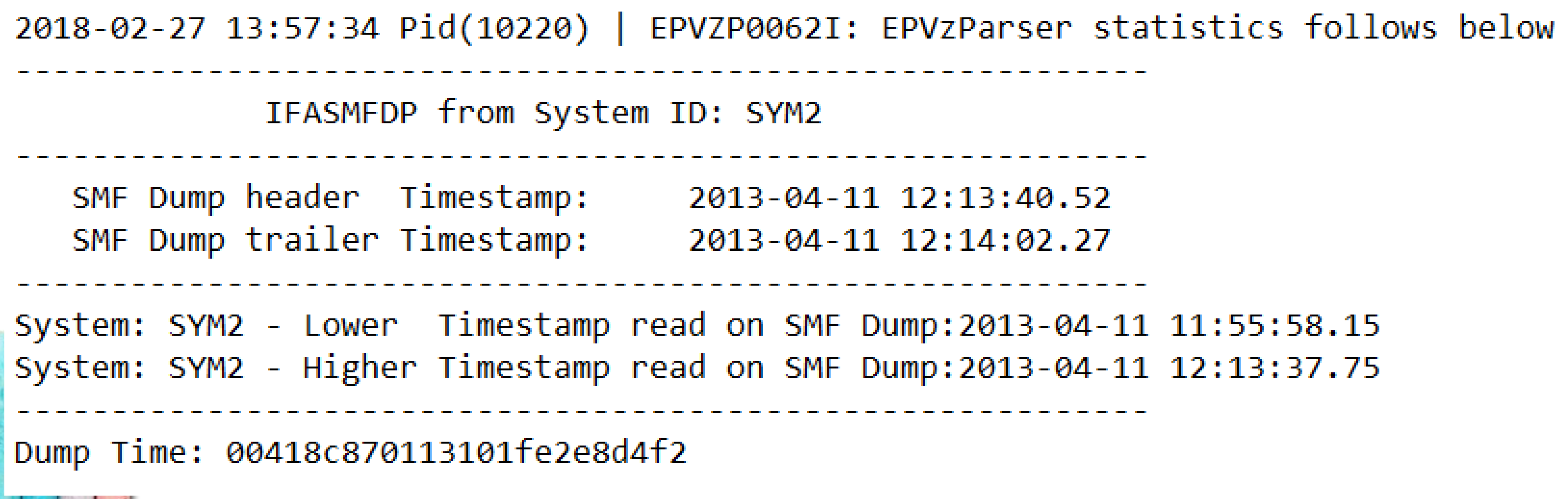
- Scrolling down in the log…
  - ✓Start message for the reading phase
  - ✓Again.. Product version
  - ✓Work area location
  - ✓Information about the eventual file compression (in this case the file wasn't compressed)



```
2018-02-27 13:57:32 Pid(10220) | EPVZP0002I: START EPVzParser FOR SMF - READING PHASE - SMF_MICS.SYM2.D130411.T121440
2018-02-27 13:57:32 Pid(10220) | EPVZP0011I: EPVzParser VERSION: 14.45
2018-02-27 13:57:32 Pid(10220) | EPVZP0250I: Current Work Area: C:/Users/teobo/Desktop/EPVZPARSER_V14_SVILUPPO/USERPROFILE/TE
2018-02-27 13:57:32 Pid(10220) | EPVZP0052I: EPVzParser STARTED at h13m57s32 for C:/Users/teobo/Desktop/EPVZPARSER_V14_SVILUP
2018-02-27 13:57:32 Pid(10220) | EPVZP0022I: Using standard opening mode for non compressed SMF records in C:/Users/teobo/Des
```

Processing controls - Logging

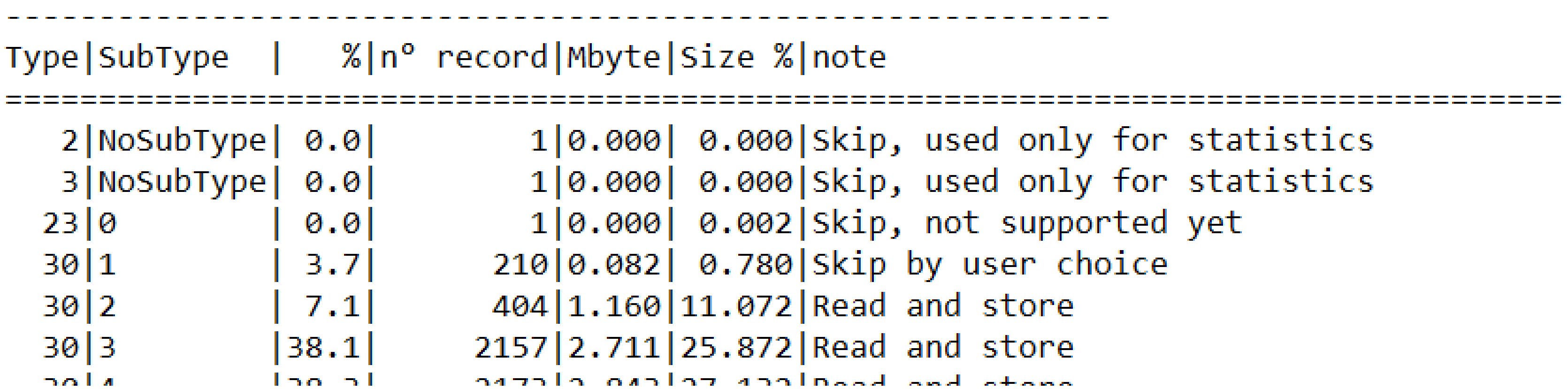
- Scrolling down in the log…
  - ✓ System that run the IFASMFDP command
  - ✓ Header/trailer timestamp
  - ✓ For each system contained in the file, lower/higher timestamp
  - ✓ Dump Time: a unique identifier of this specific SMF file, composed by the SYSTEM and the SMFTIME fields



```
2018-02-27 13:57:34 Pid(10220) | EPVZP0062I: EPVzParser statistics follows below
-----------------------------------------------------------
             IFASMFDP from System ID: SYM2
-----------------------------------------------------------
   SMF Dump header  Timestamp:     2013-04-11 12:13:40.52
   SMF Dump trailer Timestamp:     2013-04-11 12:14:02.27
-----------------------------------------------------------
System: SYM2 - Lower  Timestamp read on SMF Dump:2013-04-11 11:55:58.15
System: SYM2 - Higher Timestamp read on SMF Dump:2013-04-11 12:13:37.75
-----------------------------------------------------------
Dump Time: 00418c870113101fe2e8d4f2
```

Processing controls - Logging

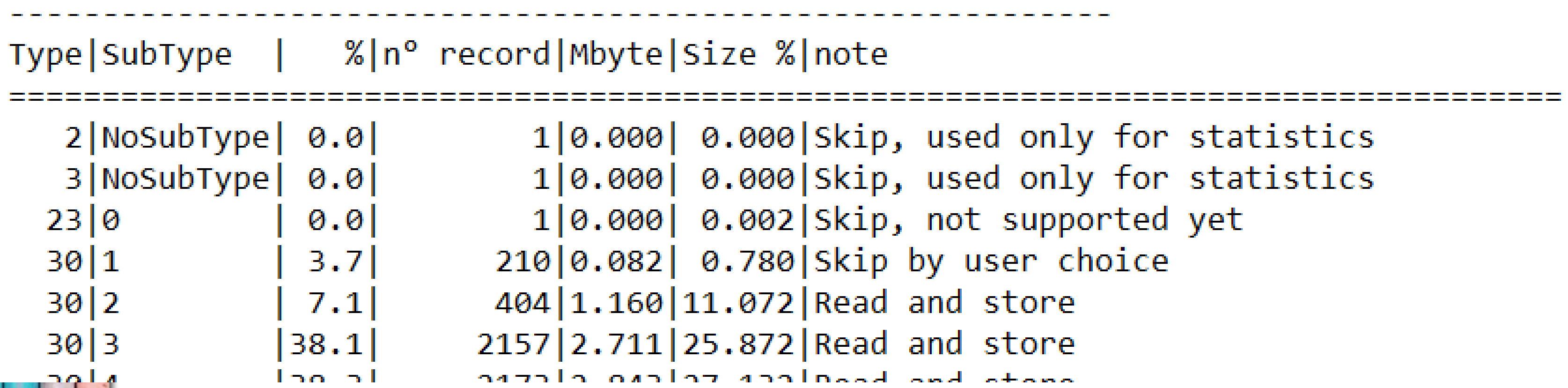
- Scrolling down in the log…
  - ✓ For each record type/subtype: %, number of records, MB size, % Size, note



```
-------------------------------------------------------------
Type|SubType  |   %|n° record|Mbyte|Size %|note
====================================================================================
   2|NoSubType| 0.0|        1|0.000| 0.000|Skip, used only for statistics
   3|NoSubType| 0.0|        1|0.000| 0.000|Skip, used only for statistics
  23|0        | 0.0|        1|0.000| 0.002|Skip, not supported yet
  30|1        | 3.7|      210|0.082| 0.780|Skip by user choice
  30|2        | 7.1|      404|1.160|11.072|Read and store
  30|3        |38.1|     2157|2.711|25.872|Read and store
```

Processing controls - Logging

- Scrolling down in the log…
  - ✓ In the note section it is possible to find:
    - ➢ Skip, used only for statistics
    - ➢ Skip, not supported yet
    - ➢ Skip, duplicated
    - ➢ Skip by user choice
    - ➢ Read and store



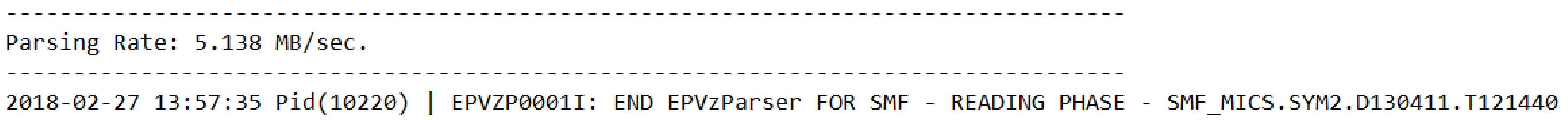
```
-----------------------------------------------------------
Type|SubType  |   %|n° record|Mbyte|Size %|note
==================================================================================
   2|NoSubType| 0.0|        1|0.000| 0.000|Skip, used only for statistics
   3|NoSubType| 0.0|        1|0.000| 0.000|Skip, used only for statistics
  23|0        | 0.0|        1|0.000| 0.002|Skip, not supported yet
  30|1        | 3.7|      210|0.082| 0.780|Skip by user choice
  30|2        | 7.1|      404|1.160|11.072|Read and store
  30|3        |38.1|     2157|2.711|25.872|Read and store
  30|4 [illegible]
```

Processing controls - Logging

- Scrolling down in the log…

  ✓ Parsing rate expressed in MB/second

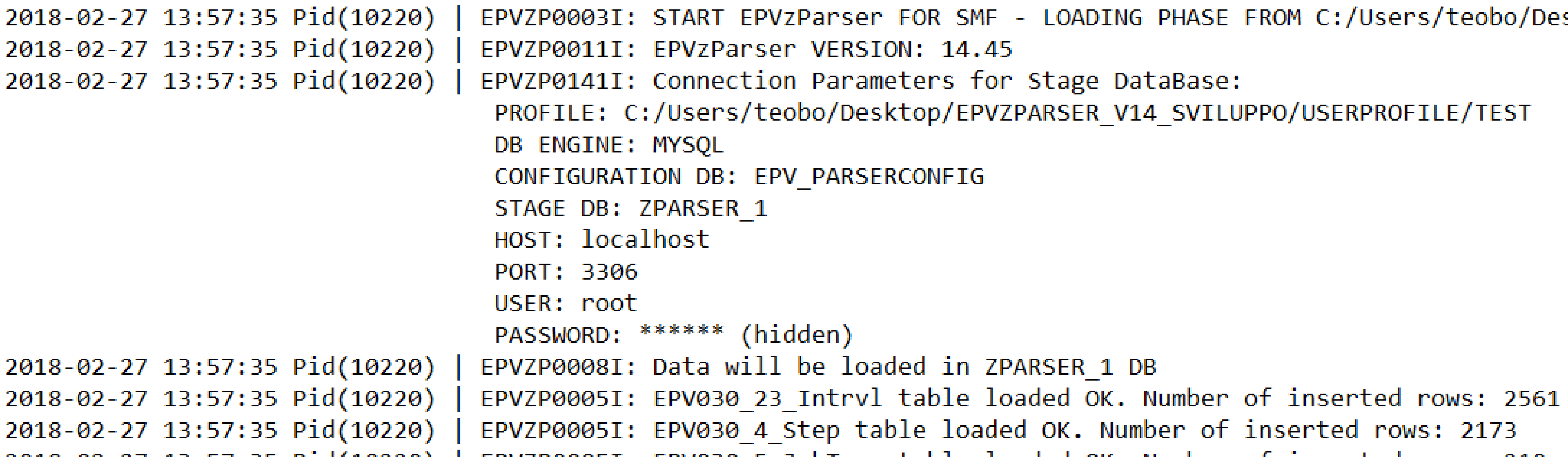
  ✓ End message for the reading phase



```
-----------------------------------------------------------------------------------
Parsing Rate: 5.138 MB/sec.
-----------------------------------------------------------------------------------
2018-02-27 13:57:35 Pid(10220) | EPVZP0001I: END EPVzParser FOR SMF - READING PHASE - SMF_MICS.SYM2.D130411.T121440
```

Processing controls - Logging

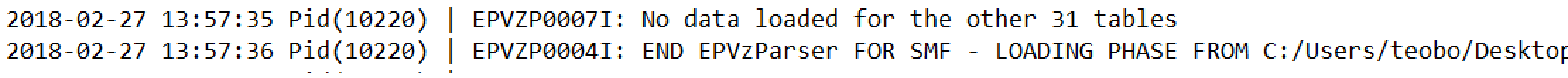
- Scrolling down in the log…
  - ✓ Start message for the loading phase
  - ✓ If you’re still confused about the product’s version.. product version!
  - ✓ Database connection parameters
  - ✓ For each table, number of inserted rows



```
2018-02-27 13:57:35 Pid(10220) | EPVZP0003I: START EPVzParser FOR SMF - LOADING PHASE FROM C:/Users/teobo/Des
2018-02-27 13:57:35 Pid(10220) | EPVZP0011I: EPVzParser VERSION: 14.45
2018-02-27 13:57:35 Pid(10220) | EPVZP0141I: Connection Parameters for Stage DataBase:
                                  PROFILE: C:/Users/teobo/Desktop/EPVZPARSER_V14_SVILUPPO/USERPROFILE/TEST
                                  DB ENGINE: MYSQL
                                  CONFIGURATION DB: EPV_PARSERCONFIG
                                  STAGE DB: ZPARSER_1
                                  HOST: localhost
                                  PORT: 3306
                                  USER: root
                                  PASSWORD: ****** (hidden)
2018-02-27 13:57:35 Pid(10220) | EPVZP0008I: Data will be loaded in ZPARSER_1 DB
2018-02-27 13:57:35 Pid(10220) | EPVZP0005I: EPV030_23_Intrvl table loaded OK. Number of inserted rows: 2561
2018-02-27 13:57:35 Pid(10220) | EPVZP0005I: EPV030_4_Step table loaded OK. Number of inserted rows: 2173
```

Processing controls - Logging

- Scrolling down in the log…
    - ✓Number of tables that were not loaded because of missing data in the input file
    - ✓End message for the loading phase



```
2018-02-27 13:57:35 Pid(10220) | EPVZP0007I: No data loaded for the other 31 tables
2018-02-27 13:57:36 Pid(10220) | EPVZP0004I: END EPVzParser FOR SMF - LOADING PHASE FROM C:/Users/teobo/Desktop
```

Processing controls - Logging

- Other component logs are provided:
  - ✓EPVzParserAgentsHandler
  - ✓EPVzParserEndOfDay
  - ✓EPVzParserDbDeaccum
  - ✓EPVzParserDbClear

Processing controls - Focal Point

- Checking log by log if all the processes are running fine could be… Frustrating for small customers, impossible for big customers

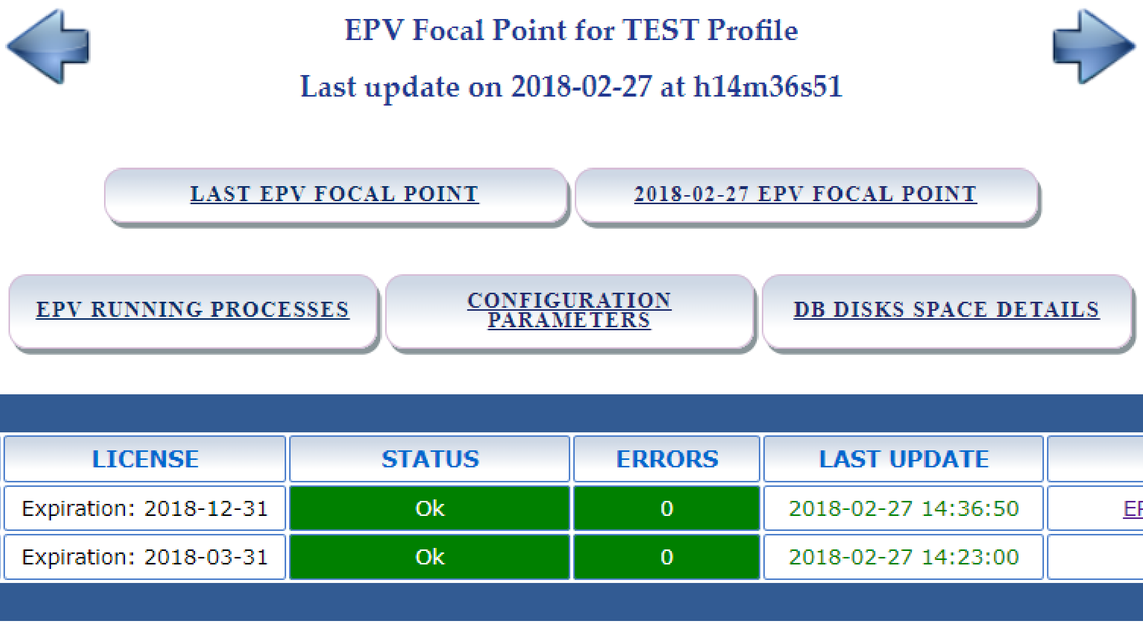
- This is why we created EPV Focal Point: to make an automatic analysis of ALL the product’s log and report the current status in an easy-to-understand format (HTML report)

Processing controls - Focal Point

- In Once a day mode, it is created by the ALLPHASES script and updated by each product that is run during the process
- In Continuous mode, it is updated by the EPVzParserAgentsHandler every 100 (default) seconds
- Its name is EPVFocalPoint.HTML and it is located in EPVROOT/USERPROFILE/*PROFILENAME*/LOGS

Processing controls - Focal Point

- Focal Point home page



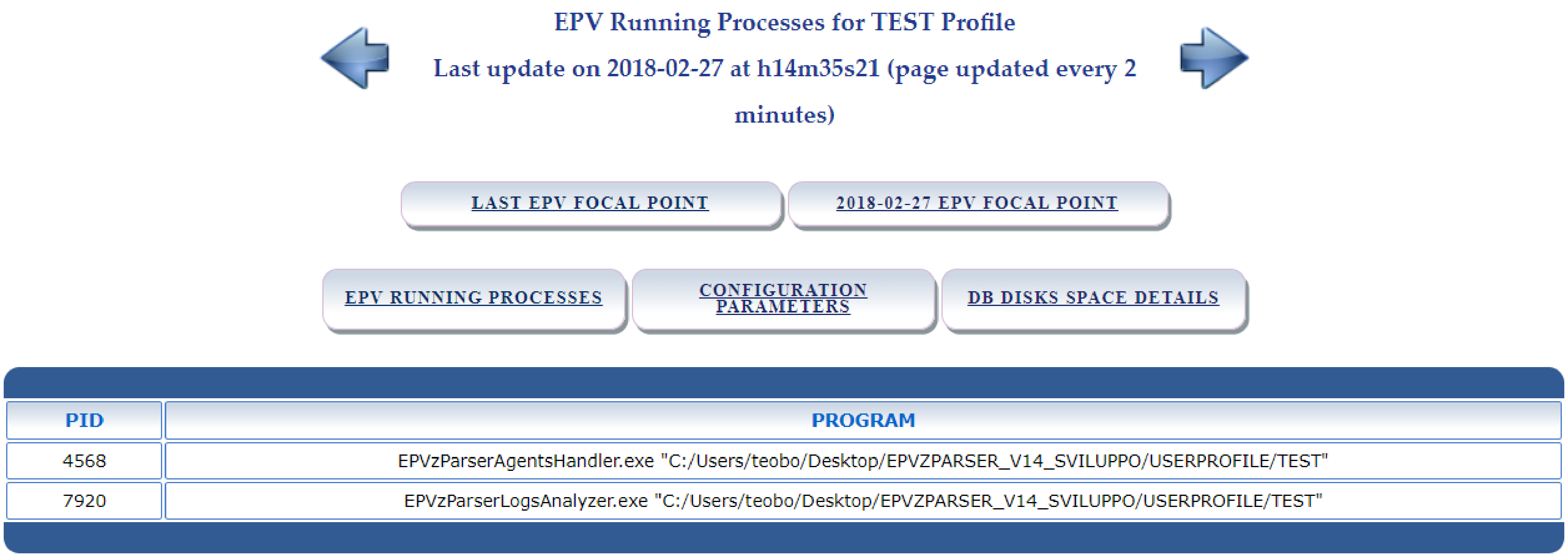
EPV Focal Point for TEST Profile

Last update on 2018-02-27 at h14m36s51

PRODUCT	LICENSE	STATUS	ERRORS	LAST UPDATE	LOG PAGE
EPV zParser V14.45	Expiration: 2018-12-31	Ok	0	2018-02-27 14:36:50	EPVzParserLogsAnalyzer.HTML
EPV for z/OS V14.21	Expiration: 2018-03-31	Ok	0	2018-02-27 14:23:00	NIGHTBATCH_ZOS.HTML

Processing controls - Focal Point

- EPV running processes

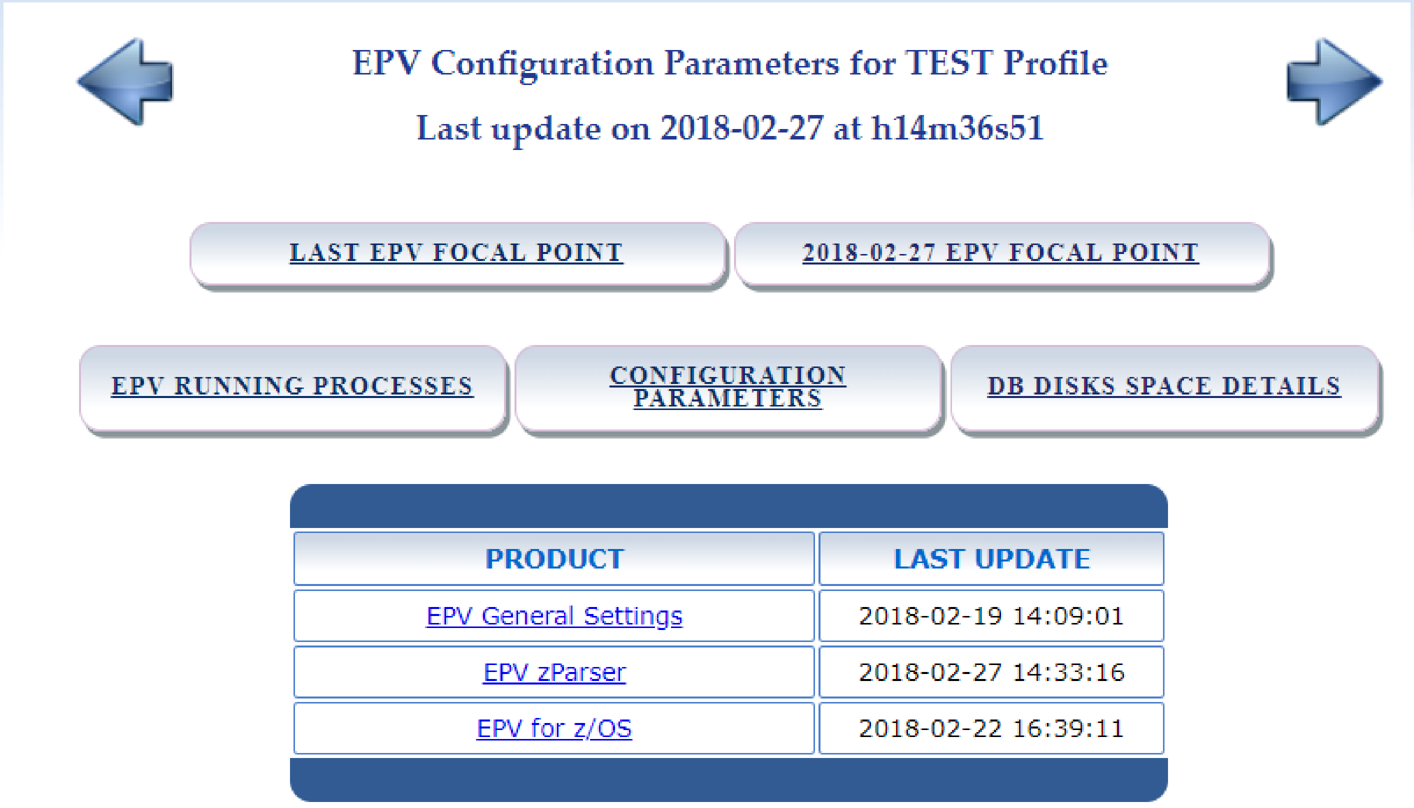

EPV Running Processes for TEST Profile

Last update on 2018-02-27 at h14m35s21 (page updated every 2 minutes)

PID	PROGRAM
4568	EPVzParserAgentsHandler.exe "C:/Users/teobo/Desktop/EPVZPARSER_V14_SVILUPPO/USERPROFILE/TEST"
7920	EPVzParserLogsAnalyzer.exe "C:/Users/teobo/Desktop/EPVZPARSER_V14_SVILUPPO/USERPROFILE/TEST"

Processing controls - Focal Point

- Configuration parameters



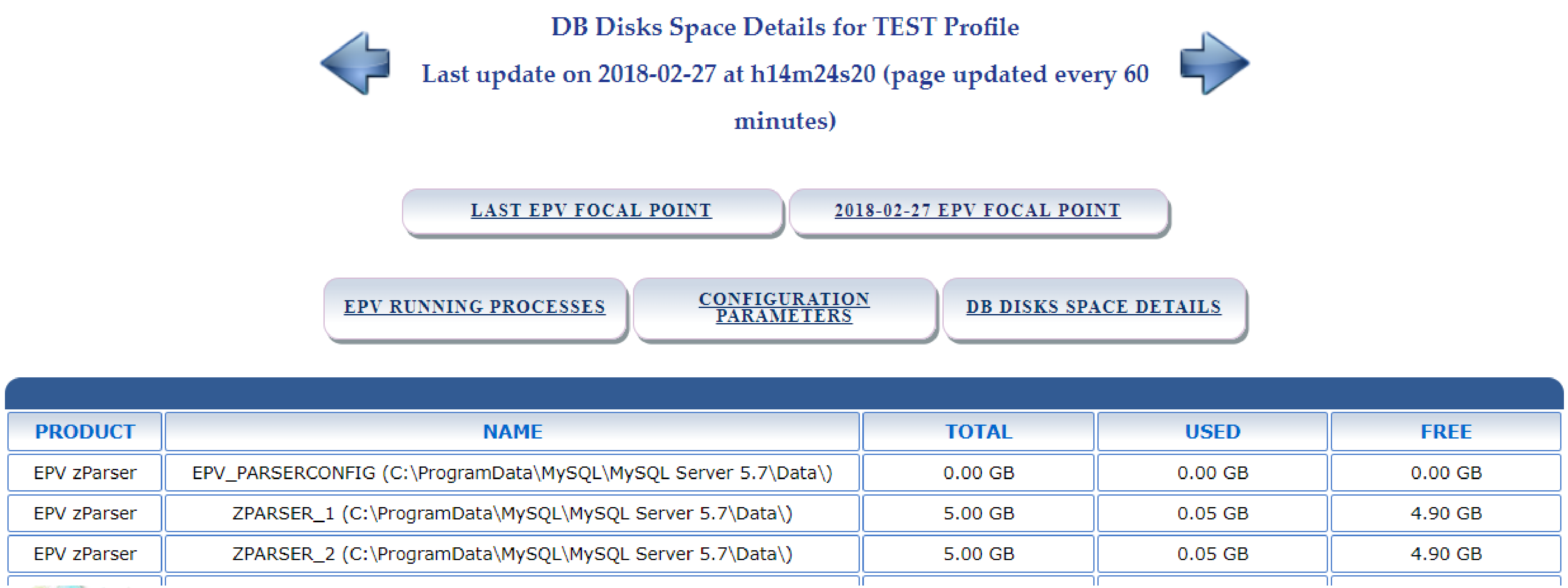
EPV Configuration Parameters for TEST Profile

Last update on 2018-02-27 at h14m36s51

PRODUCT	LAST UPDATE
EPV General Settings	2018-02-19 14:09:01
EPV zParser	2018-02-27 14:33:16
EPV for z/OS	2018-02-22 16:39:11

Processing controls - Focal Point

- DB disks space details



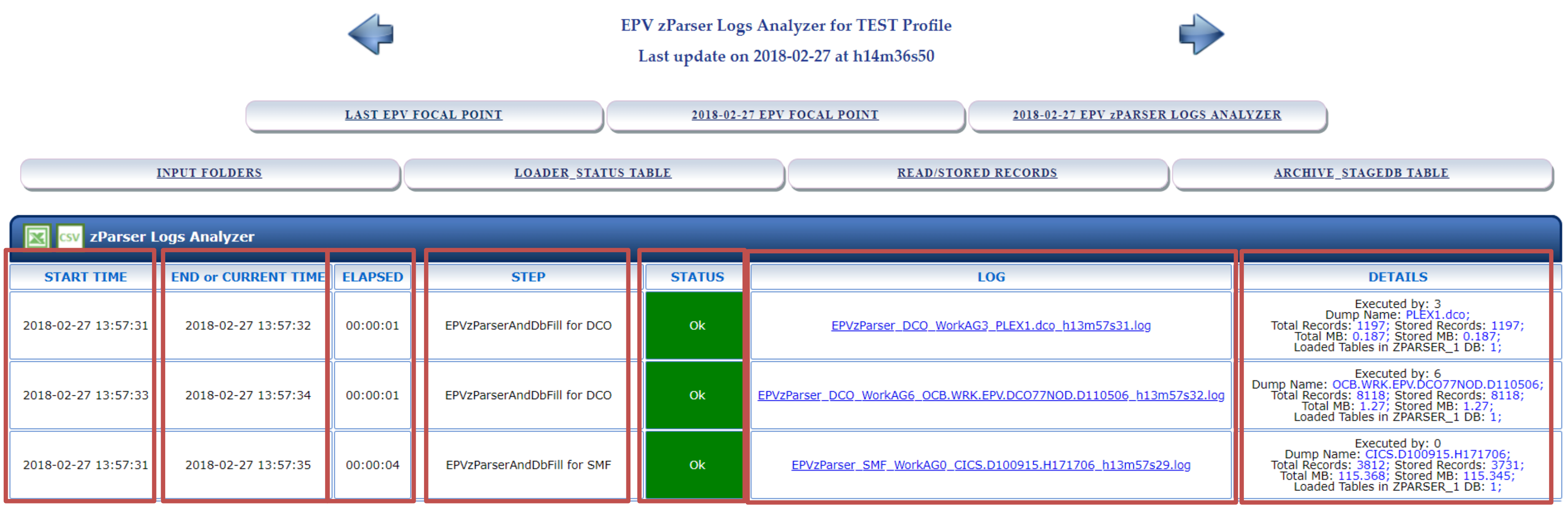
DB Disks Space Details for TEST Profile

Last update on 2018-02-27 at h14m24s20 (page updated every 60 minutes)

PRODUCT	NAME	TOTAL	USED	FREE
EPV zParser	EPV_PARSERCONFIG (C:\ProgramData\MySQL\MySQL Server 5.7\Data\)	0.00 GB	0.00 GB	0.00 GB
EPV zParser	ZPARSER_1 (C:\ProgramData\MySQL\MySQL Server 5.7\Data\)	5.00 GB	0.05 GB	4.90 GB
EPV zParser	ZPARSER_2 (C:\ProgramData\MySQL\MySQL Server 5.7\Data\)	5.00 GB	0.05 GB	4.90 GB

Processing controls - Focal Point

- zParser dedicated section



EPV zParser Logs Analyzer for TEST Profile

Last update on 2018-02-27 at h14m36s50

START TIME	END or CURRENT TIME	ELAPSED	STEP	STATUS	LOG	DETAILS

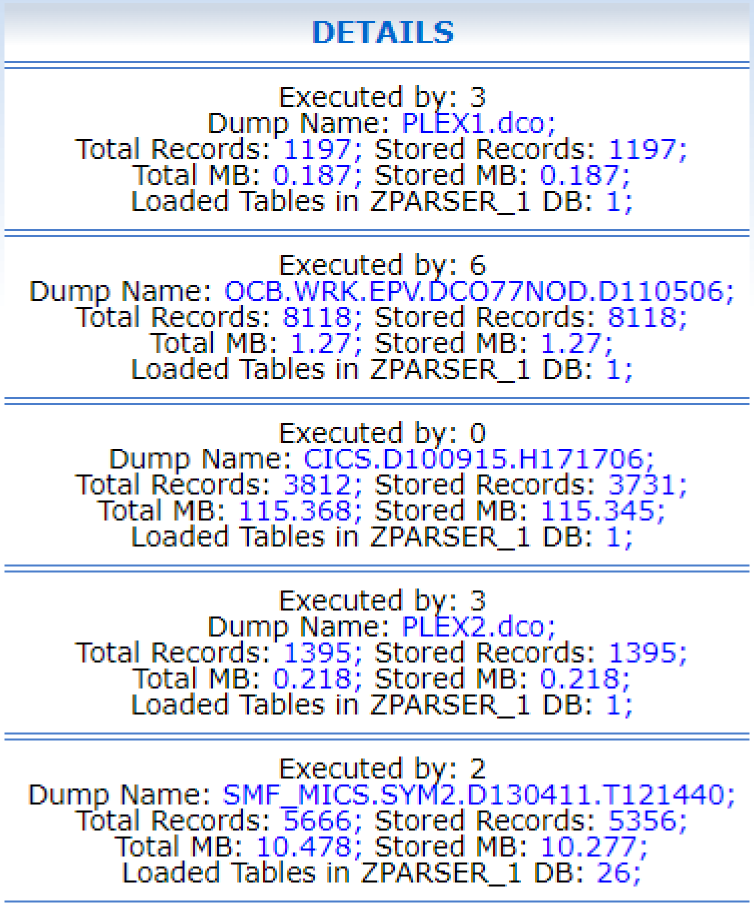
2018-02-27 13:57:31	2018-02-27 13:57:32	00:00:01	EPVzParserAndDbFill for DCO	Ok	EPVzParser_DCO_WorkAG3_PLEX1.dco_h13m57s31.log	Executed by: 3
						Dump Name: PLEX1.dco;
						Total Records: 1197; Stored Records: 1197;
						Total MB: 0.187; Stored MB: 0.187;
						Loaded Tables in ZPARSER_1 DB: 1;

2018-02-27 13:57:33	2018-02-27 13:57:34	00:00:01	EPVzParserAndDbFill for DCO	Ok	EPVzParser_DCO_WorkAG6_OCB.WRK.EPV.DCO77NOD.D110506_h13m57s32.log	Executed by: 6
						Dump Name: OCB.WRK.EPV.DCO77NOD.D110506;
						Total Records: 8118; Stored Records: 8118;
						Total MB: 1.27; Stored MB: 1.27;
						Loaded Tables in ZPARSER_1 DB: 1;

2018-02-27 13:57:31	2018-02-27 13:57:35	00:00:04	EPVzParserAndDbFill for SMF	Ok	EPVzParser_SMF_WorkAG0_CICS.D100915.H171706_h13m57s29.log	Executed by: 0
						Dump Name: CICS.D100915.H171706;
						Total Records: 3812; Stored Records: 3731;
						Total MB: 115.368; Stored MB: 115.345;
						Loaded Tables in ZPARSER_1 DB: 1;

Processing controls - Focal Point

- zParser dedicated section – Details column



DETAILS

Executed by: 3
Dump Name: PLEX1.dco;
Total Records: 1197; Stored Records: 1197;
Total MB: 0.187; Stored MB: 0.187;
Loaded Tables in ZPARSER_1 DB: 1;

Executed by: 6
Dump Name: OCB.WRK.EPV.DCO77NOD.D110506;
Total Records: 8118; Stored Records: 8118;
Total MB: 1.27; Stored MB: 1.27;
Loaded Tables in ZPARSER_1 DB: 1;

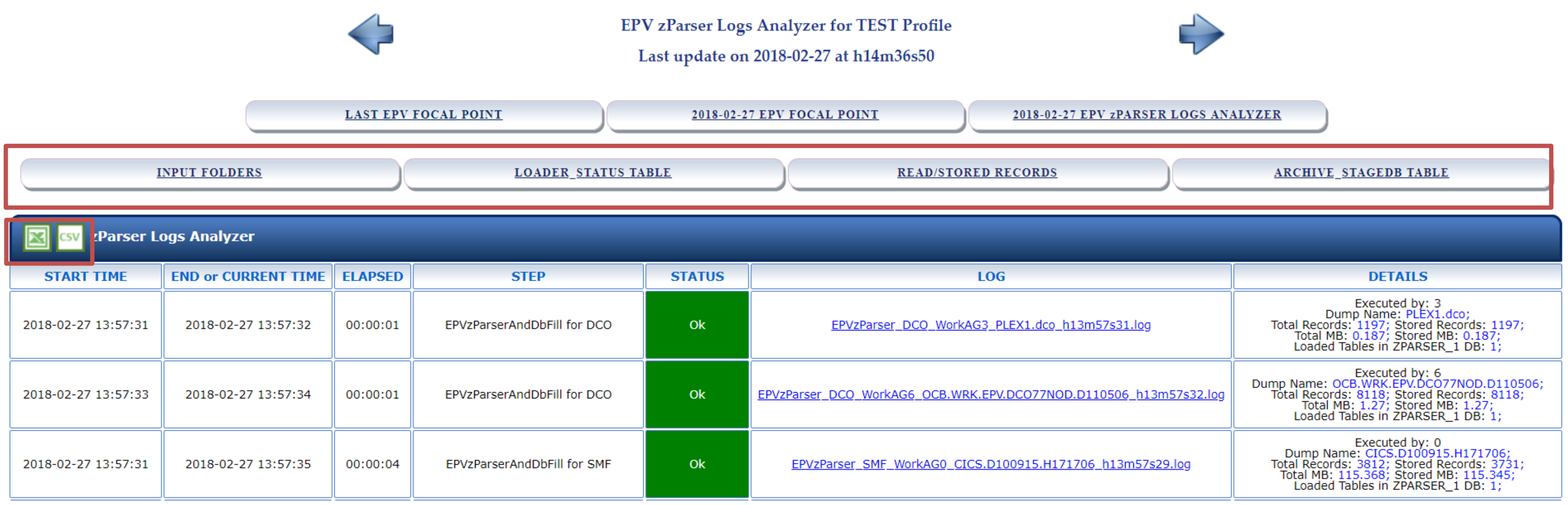
Executed by: 0
Dump Name: CICS.D100915.H171706;
Total Records: 3812; Stored Records: 3731;
Total MB: 115.368; Stored MB: 115.345;
Loaded Tables in ZPARSER_1 DB: 1;

Executed by: 3
Dump Name: PLEX2.dco;
Total Records: 1395; Stored Records: 1395;
Total MB: 0.218; Stored MB: 0.218;
Loaded Tables in ZPARSER_1 DB: 1;

Executed by: 2
Dump Name: SMF_MICS.SYM2.D130411.T121440;
Total Records: 5666; Stored Records: 5356;
Total MB: 10.478; Stored MB: 10.277;
Loaded Tables in ZPARSER_1 DB: 26;

Processing controls - Focal Point

- zParser dedicated section – Other useful information



EPV zParser Logs Analyzer for TEST Profile

Last update on 2018-02-27 at h14m36s50

zParser Logs Analyzer

START TIME	END or CURRENT TIME	ELAPSED	STEP	STATUS	LOG	DETAILS

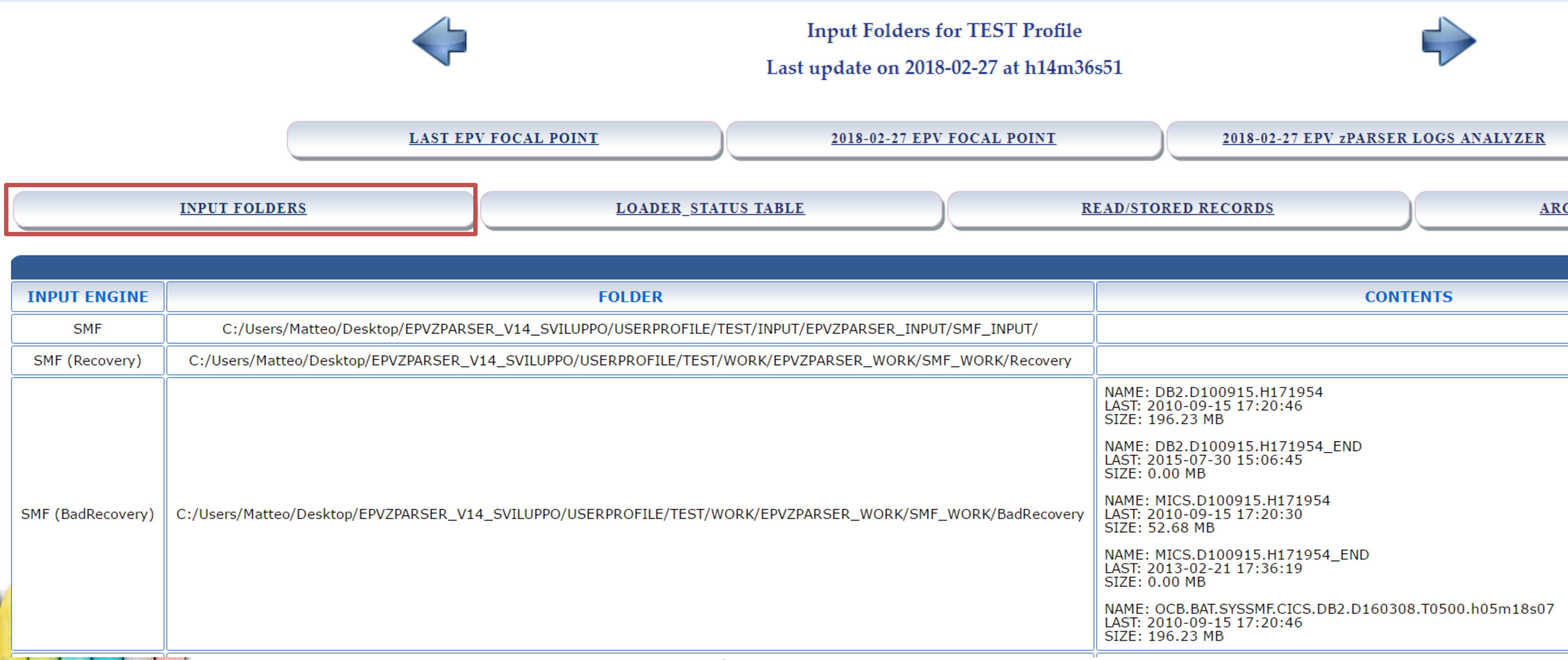
2018-02-27 13:57:31	2018-02-27 13:57:32	00:00:01	EPVzParserAndDbFill for DCO	Ok	EPVzParser_DCO_WorkAG3_PLEX1.dco_h13m57s31.log	Executed by: 3
						Dump Name: PLEX1.dco;
						Total Records: 1197; Stored Records: 1197;
						Total MB: 0.187; Stored MB: 0.187;
						Loaded Tables in ZPARSER_1 DB: 1;

2018-02-27 13:57:33	2018-02-27 13:57:34	00:00:01	EPVzParserAndDbFill for DCO	Ok	EPVzParser_DCO_WorkAG6_OCB.WRK.EPV.DCO77NOD.D110506_h13m57s32.log	Executed by: 6
						Dump Name: OCB.WRK.EPV.DCO77NOD.D110506;
						Total Records: 8118; Stored Records: 8118;
						Total MB: 1.27; Stored MB: 1.27;
						Loaded Tables in ZPARSER_1 DB: 1;

2018-02-27 13:57:31	2018-02-27 13:57:35	00:00:04	EPVzParserAndDbFill for SMF	Ok	EPVzParser_SMF_WorkAG0_CICS.D100915.H171706_h13m57s29.log	Executed by: 0
						Dump Name: CICS.D100915.H171706;
						Total Records: 3812; Stored Records: 3731;
						Total MB: 115.368; Stored MB: 115.345;
						Loaded Tables in ZPARSER_1 DB: 1;

Processing controls - Focal Point

- zParser dedicated section – Input folders



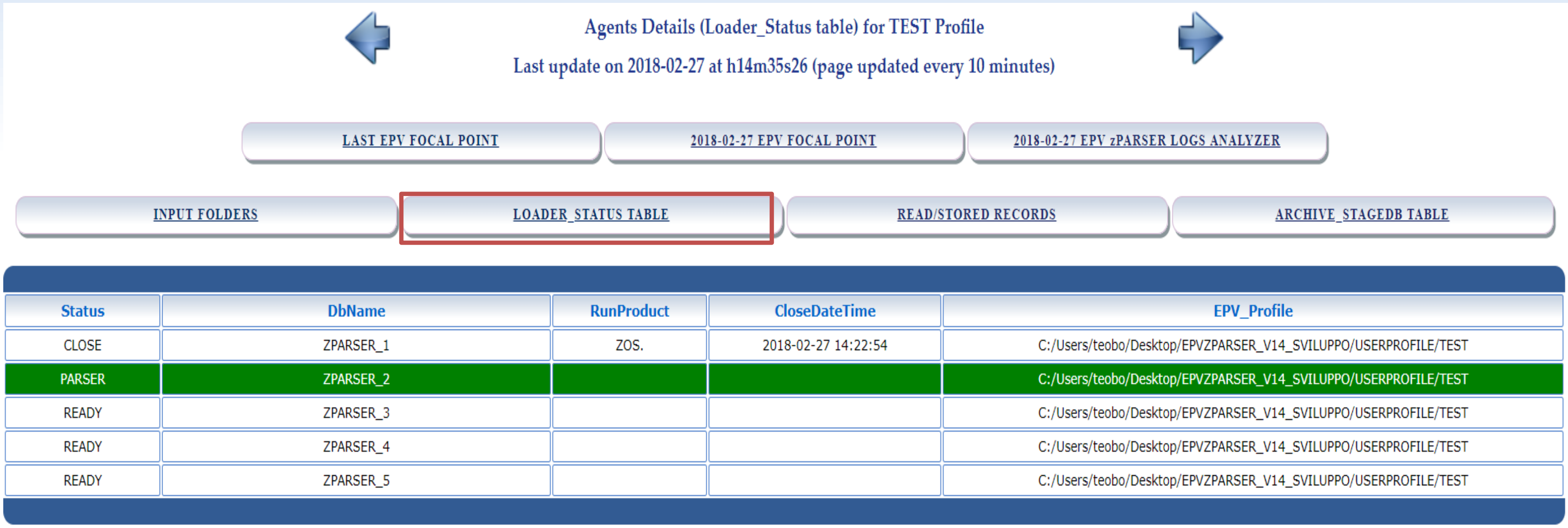
Input Folders for TEST Profile

Last update on 2018-02-27 at h14m36s51

INPUT ENGINE	FOLDER	CONTENTS
SMF	C:/Users/Matteo/Desktop/EPVZPARSER_V14_SVILUPPO/USERPROFILE/TEST/INPUT/EPVZPARSER_INPUT/SMF_INPUT/	
SMF (Recovery)	C:/Users/Matteo/Desktop/EPVZPARSER_V14_SVILUPPO/USERPROFILE/TEST/WORK/EPVZPARSER_WORK/SMF_WORK/Recovery	
SMF (BadRecovery)	C:/Users/Matteo/Desktop/EPVZPARSER_V14_SVILUPPO/USERPROFILE/TEST/WORK/EPVZPARSER_WORK/SMF_WORK/BadRecovery	NAME: DB2.D100915.H171954 LAST: 2010-09-15 17:20:46 SIZE: 196.23 MB NAME: DB2.D100915.H171954_END LAST: 2015-07-30 15:06:45 SIZE: 0.00 MB NAME: MICS.D100915.H171954 LAST: 2010-09-15 17:20:30 SIZE: 52.68 MB NAME: MICS.D100915.H171954_END LAST: 2013-02-21 17:36:19 SIZE: 0.00 MB NAME: OCB.BAT.SYSSMF.CICS.DB2.D160308.T0500.h05m18s07 LAST: 2010-09-15 17:20:46 SIZE: 196.23 MB

Processing controls - Focal Point

- zParser dedicated section – Loader_Status table



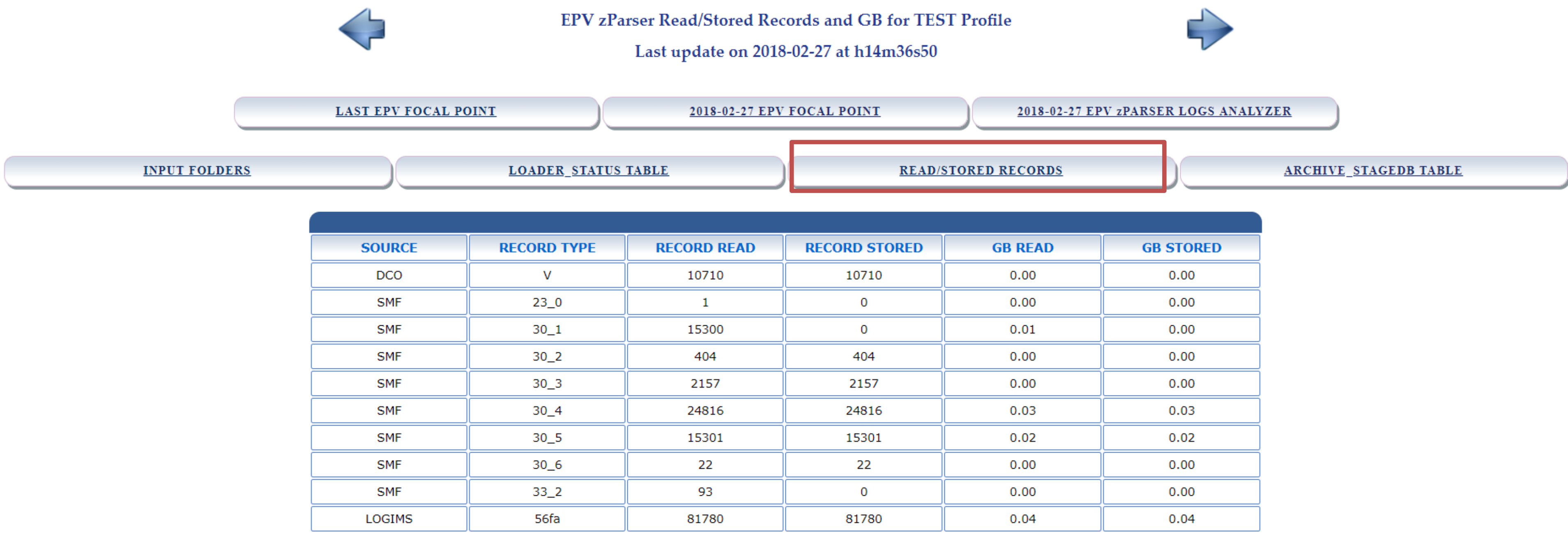
Agents Details (Loader_Status table) for TEST Profile

Last update on 2018-02-27 at h14m35s26 (page updated every 10 minutes)

Status	DbName	RunProduct	CloseDateTime	EPV_Profile
CLOSE	ZPARSER_1	ZOS.	2018-02-27 14:22:54	C:/Users/teobo/Desktop/EPVZPARSER_V14_SVILUPPO/USERPROFILE/TEST
PARSER	ZPARSER_2			C:/Users/teobo/Desktop/EPVZPARSER_V14_SVILUPPO/USERPROFILE/TEST
READY	ZPARSER_3			C:/Users/teobo/Desktop/EPVZPARSER_V14_SVILUPPO/USERPROFILE/TEST
READY	ZPARSER_4			C:/Users/teobo/Desktop/EPVZPARSER_V14_SVILUPPO/USERPROFILE/TEST
READY	ZPARSER_5			C:/Users/teobo/Desktop/EPVZPARSER_V14_SVILUPPO/USERPROFILE/TEST

Processing controls - Focal Point

- zParser dedicated section – Read/stored records



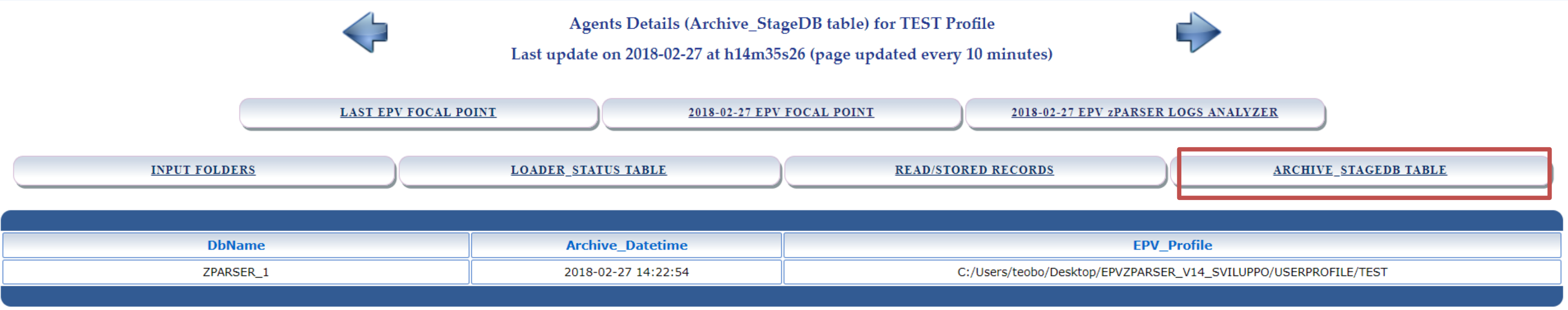
EPV zParser Read/Stored Records and GB for TEST Profile

Last update on 2018-02-27 at h14m36s50

SOURCE	RECORD TYPE	RECORD READ	RECORD STORED	GB READ	GB STORED
DCO	V	10710	10710	0.00	0.00
SMF	23_0	1	0	0.00	0.00
SMF	30_1	15300	0	0.01	0.00
SMF	30_2	404	404	0.00	0.00
SMF	30_3	2157	2157	0.00	0.00
SMF	30_4	24816	24816	0.03	0.03
SMF	30_5	15301	15301	0.02	0.02
SMF	30_6	22	22	0.00	0.00
SMF	33_2	93	0	0.00	0.00
LOGIMS	56fa	81780	81780	0.04	0.04

Processing controls - Focal Point

- zParser dedicated section – Archive_StageDB table


Agents Details (Archive_StageDB table) for TEST Profile
Last update on 2018-02-27 at h14m35s26 (page updated every 10 minutes)

DbName	Archive_Datetime	EPV_Profile
ZPARSER_1	2018-02-27 14:22:54	C:/Users/teobo/Desktop/EPVZPARSER_V14_SVILUPPO/USERPROFILE/TEST

Processing controls - Focal Point Mobile

- If users want to remotely check the EPV status with their smartphones, there's also this possibility thanks to a new responsive set of HTML pages

- Users can choose where to put them:
  - ✓ EPV server
  - ✓ Remote folder
  - ✓ FTP folder

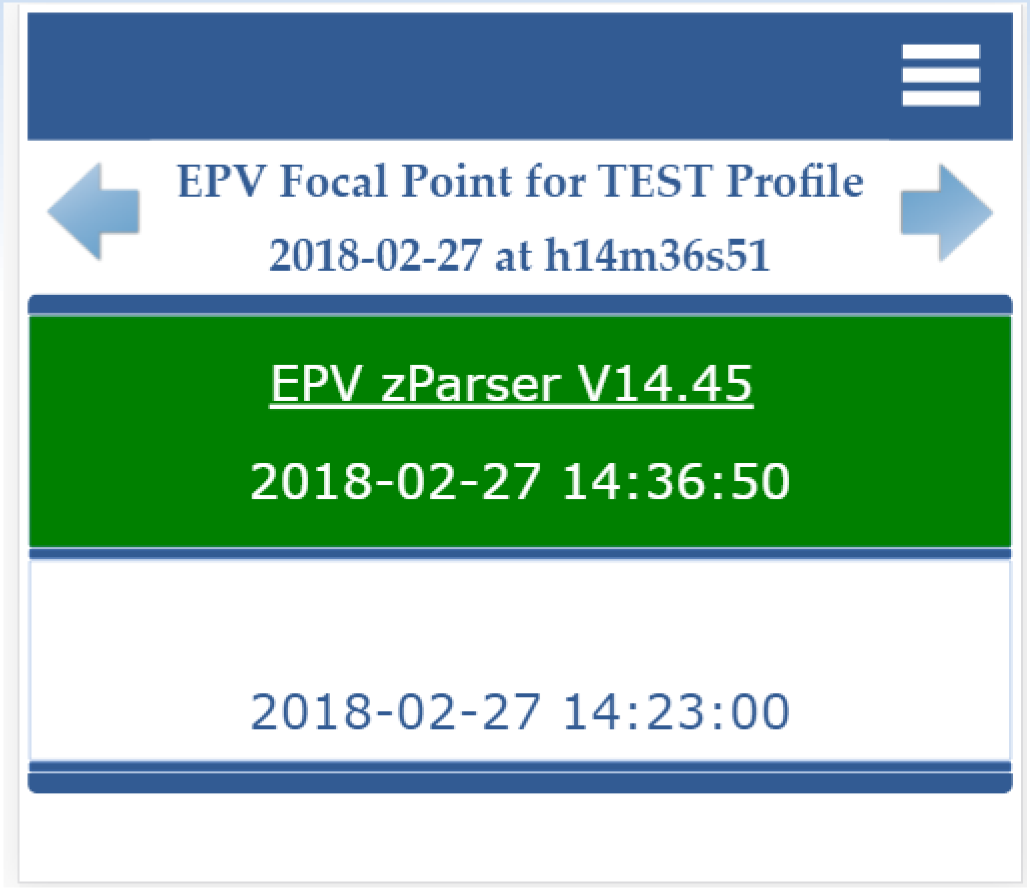
Processing controls - Focal Point Mobile

- Only the most important pages:

  - ✓ Focal Point home page
  - ✓ One page for each EPV product
  - ✓ EPV zParser Input Folders
  - ✓ EPV Running Processes
  - ✓ EPV zParser DBs Status (only in ‘Continuous Mode’)

Processing controls - Focal Point Mobile

- Focal Point Mobile home page



EPV Focal Point for TEST Profile
2018-02-27 at h14m36s51

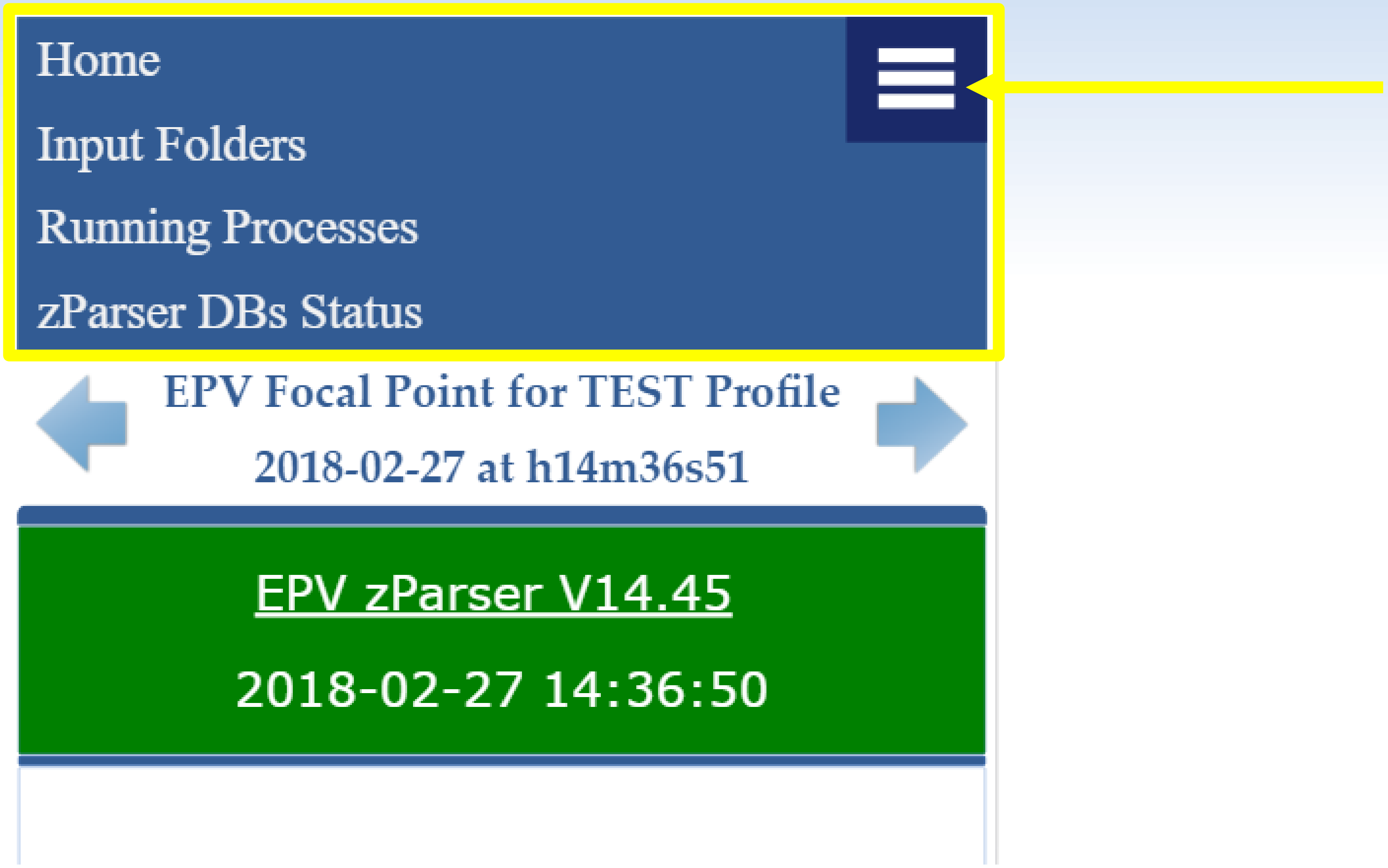
EPV zParser V14.45

2018-02-27 14:36:50

2018-02-27 14:23:00

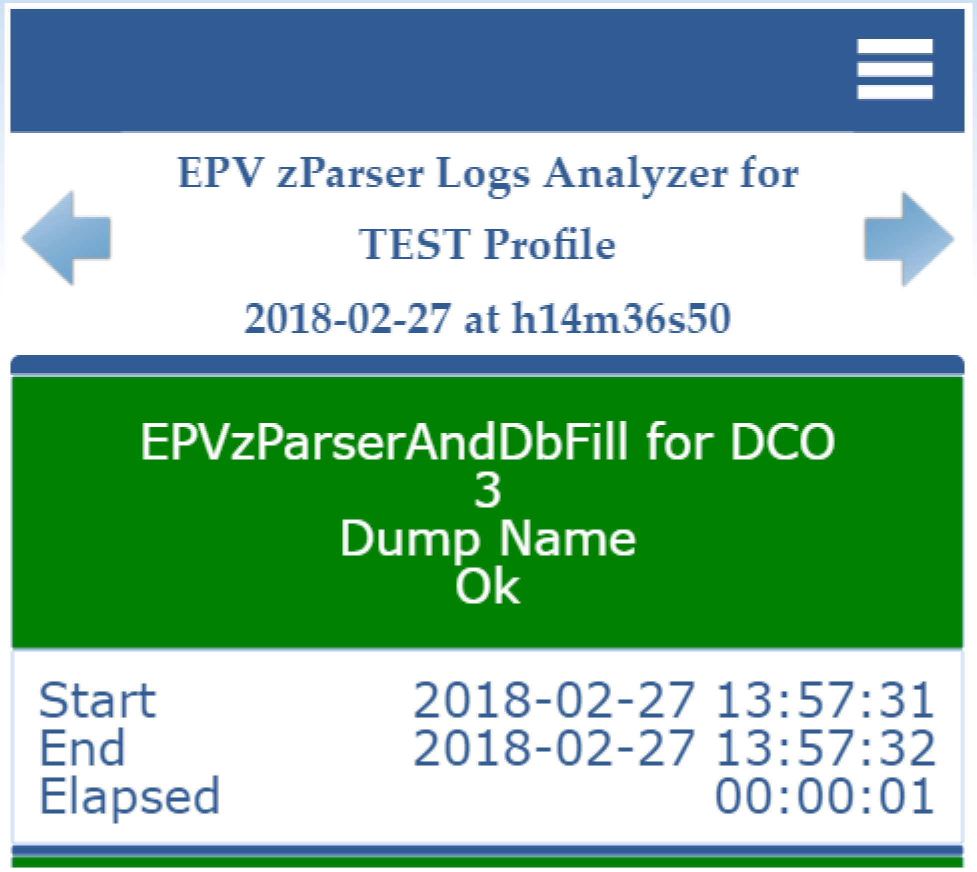
Processing controls - Focal Point Mobile

- Focal Point Mobile – Menu button



Processing controls - Focal Point Mobile

- zParser specific section



EPV zParser Logs Analyzer for

TEST Profile

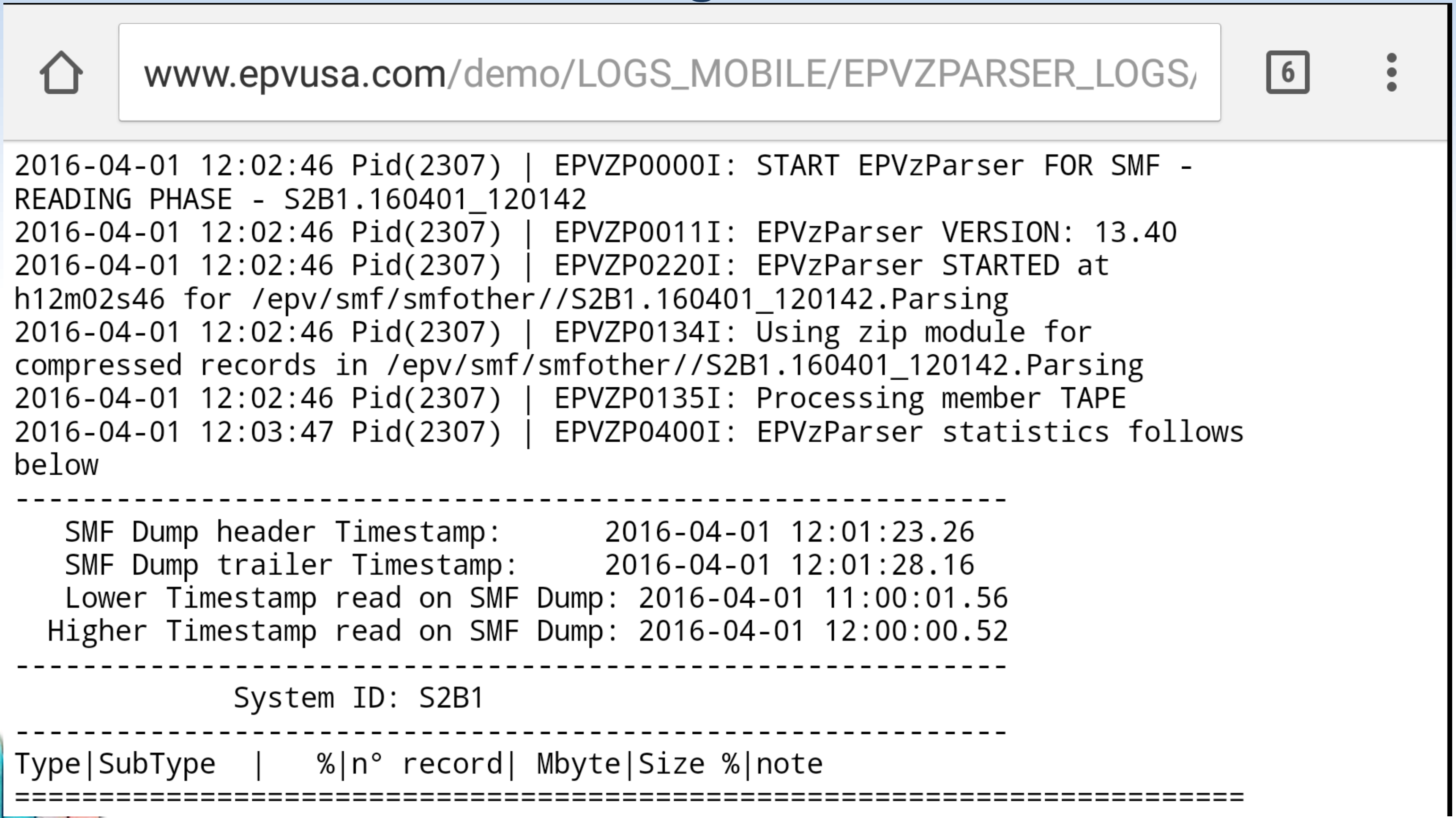
2018-02-27 at h14m36s50

EPVzParserAndDbFill for DCO
3
Dump Name
Ok

Start	2018-02-27 13:57:31
End	2018-02-27 13:57:32
Elapsed	00:00:01

Processing controls - Focal Point Mobile

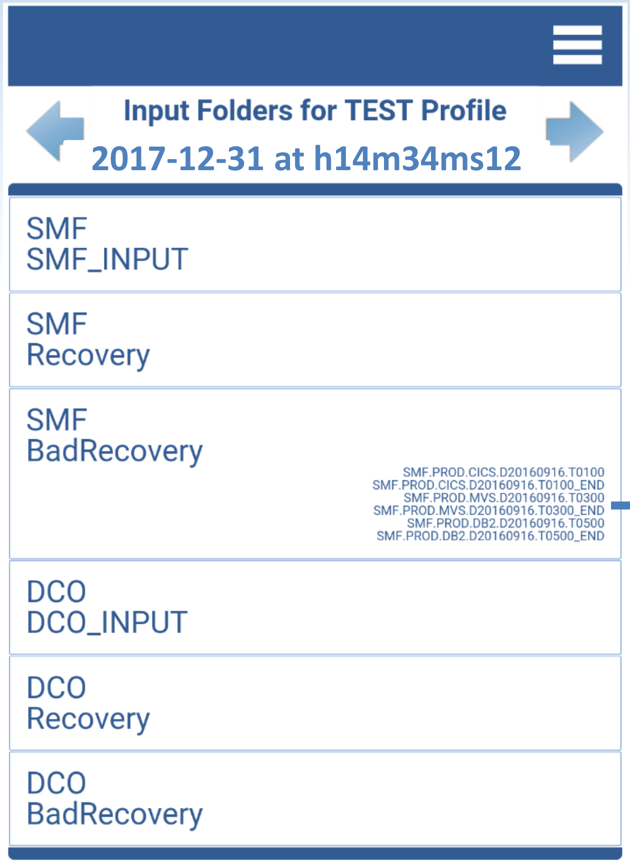
- Direct access to error logs



```
2016-04-01 12:02:46 Pid(2307) | EPVZP0000I: START EPVzParser FOR SMF -
READING PHASE - S2B1.160401_120142
2016-04-01 12:02:46 Pid(2307) | EPVZP0011I: EPVzParser VERSION: 13.40
2016-04-01 12:02:46 Pid(2307) | EPVZP0220I: EPVzParser STARTED at
h12m02s46 for /epv/smf/smfother//S2B1.160401_120142.Parsing
2016-04-01 12:02:46 Pid(2307) | EPVZP0134I: Using zip module for
compressed records in /epv/smf/smfother//S2B1.160401_120142.Parsing
2016-04-01 12:02:46 Pid(2307) | EPVZP0135I: Processing member TAPE
2016-04-01 12:03:47 Pid(2307) | EPVZP0400I: EPVzParser statistics follows
below
-----------------------------------------------------------
   SMF Dump header Timestamp:      2016-04-01 12:01:23.26
   SMF Dump trailer Timestamp:     2016-04-01 12:01:28.16
   Lower Timestamp read on SMF Dump: 2016-04-01 11:00:01.56
  Higher Timestamp read on SMF Dump: 2016-04-01 12:00:00.52
-----------------------------------------------------------
             System ID: S2B1
-----------------------------------------------------------
Type|SubType  |   %|n° record| Mbyte|Size %|note
=========================================================================
```

Processing controls - Focal Point Mobile

- Input folders – Pinch or double tap to zoom



Input Folders for TEST Profile
2017-12-31 at h14m34ms12

SMF
SMF_INPUT

SMF
Recovery

SMF
BadRecovery

SMF.PROD.CICS.D20160916.T0100
SMF.PROD.CICS.D20160916.T0100_END
SMF.PROD.MVS.D20160916.T0300
SMF.PROD.MVS.D20160916.T0300_END
SMF.PROD.DB2.D20160916.T0500
SMF.PROD.DB2.D20160916.T0500_END

DCO
DCO_INPUT

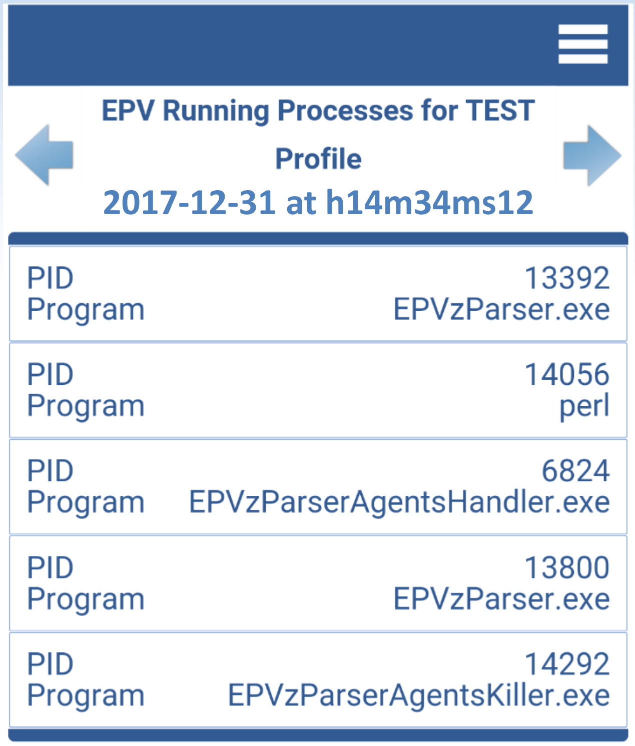
DCO
Recovery

DCO
BadRecovery

SMF.PROD.CICS.D20160916.T0100
SMF.PROD.CICS.D20160916.T0100_END
SMF.PROD.MVS.D20160916.T0300
SMF.PROD.MVS.D20160916.T0300_END
SMF.PROD.DB2.D20160916.T0500
SMF.PROD.DB2.D20160916.T0500_END

Processing controls - Focal Point Mobile

- EPV Running Processes

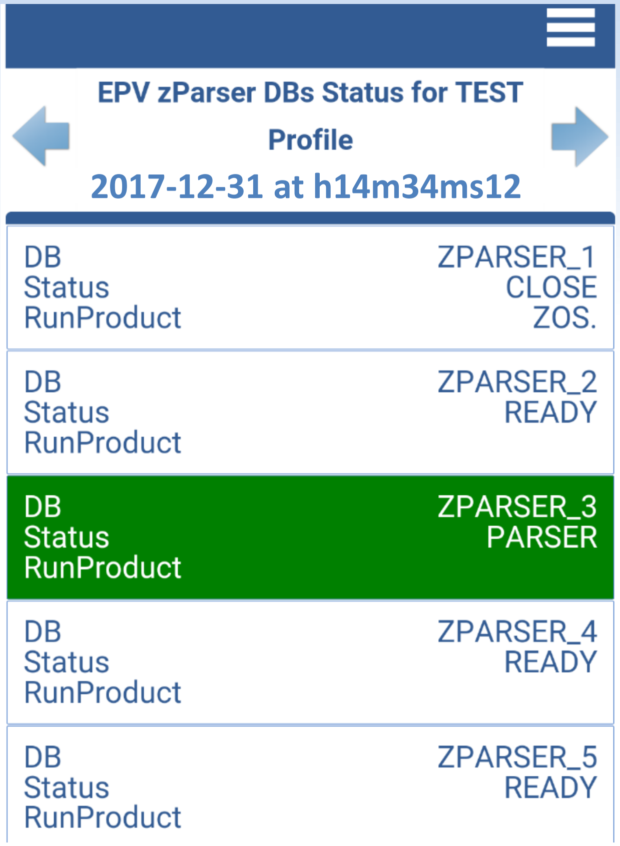


EPV Running Processes for TEST Profile
2017-12-31 at h14m34ms12

PID	13392
Program	EPVzParser.exe
PID	14056
Program	perl
PID	6824
Program	EPVzParserAgentsHandler.exe
PID	13800
Program	EPVzParser.exe
PID	14292
Program	EPVzParserAgentsKiller.exe

Processing controls - Focal Point Mobile

- zParser's databases status (continuous mode)



EPV zParser DBs Status for TEST Profile
2017-12-31 at h14m34ms12

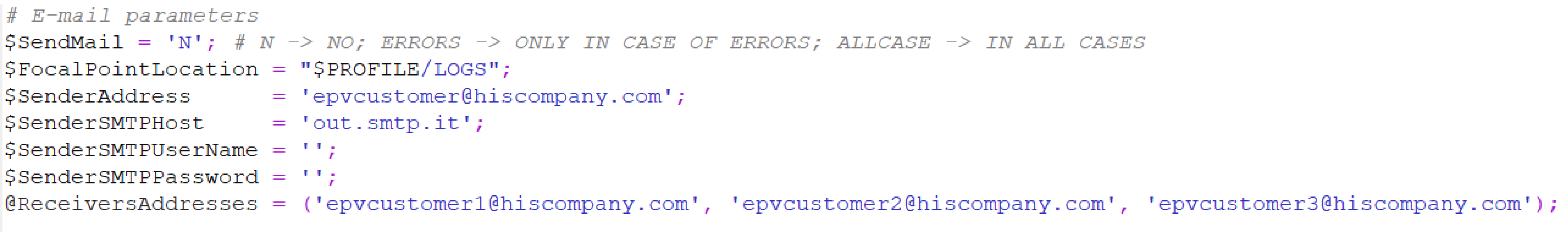
DB	ZPARSER_1
Status	CLOSE
RunProduct	ZOS.
DB	ZPARSER_2
Status	READY
RunProduct	
DB	ZPARSER_3
Status	PARSER
RunProduct	
DB	ZPARSER_4
Status	READY
RunProduct	
DB	ZPARSER_5
Status	READY
RunProduct	

Processing controls - E-mail alerting

- Another way to control the processing is e-mail
- Users can customize zParser to send the textual version of the Focal Point
  - ✓ Once a day after the end of the reports production
  - ✓ Once a day after the end of the reports production but just in case of errors
  - ✓ Multiple times per day

Processing controls - E-mail alerting

- To activate the automatic sending of e-mail once a day after the reports production, customize EPVROOT/PRODUCTS/EPVZPARSER_V14/PARSER_AGENT/ EPVFocalPoint_USER_EXIT.PL



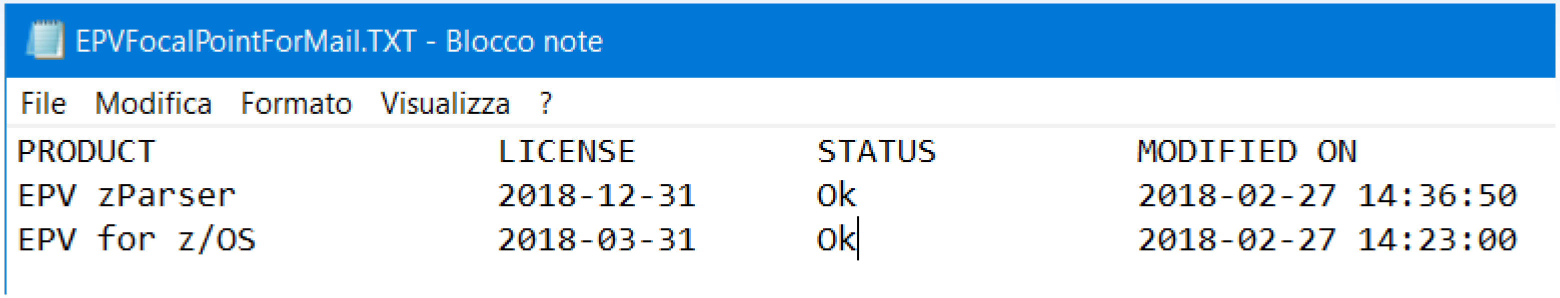
```
# E-mail parameters
$SendMail = 'N'; # N -> NO; ERRORS -> ONLY IN CASE OF ERRORS; ALLCASE -> IN ALL CASES
$FocalPointLocation = "$PROFILE/LOGS";
$SenderAddress      = 'epvcustomer@hiscompany.com';
$SenderSMTPHost     = 'out.smtp.it';
$SenderSMTPUserName = '';
$SenderSMTPPassword = '';
@ReceiversAddresses = ('epvcustomer1@hiscompany.com', 'epvcustomer2@hiscompany.com', 'epvcustomer3@hiscompany.com');
```

Processing controls - E-mail alerting

- ✓ To send e-mails about the status of EPV multiple times per day, users can also schedule their own e-mail tools to send
  - ✓ EPVFocalPointForMail.TXT -> e-mail body
  - ✓ EPVFocalPointForMail_OBJECT.TXT -> e-mail object
- EPVFocalPointForMail.TXT is the textual version of the Focal Point and it is updated every 100 (default) seconds when in Continuous mode
- These files are located in EPVROOT/USERPROFILE/*PROFILENAME*/LOGS/

Processing controls - E-mail alerting

- EPVFocalPointForMail.TXT


EPVFocalPointForMail.TXT - Blocco note
File Modifica Formato Visualizza ?

```
PRODUCT                LICENSE         STATUS          MODIFIED ON
EPV zParser            2018-12-31      Ok              2018-02-27 14:36:50
EPV for z/OS           2018-03-31      Ok              2018-02-27 14:23:00
```

Data recovery

Data recovery - Recovery & BadRecovery

- During the parsing process (reader or loader) many temporary problems could occur:
  - ✓Work area disconnection (if remote)
  - ✓Antivirus scansion
  - ✓Missing database connection

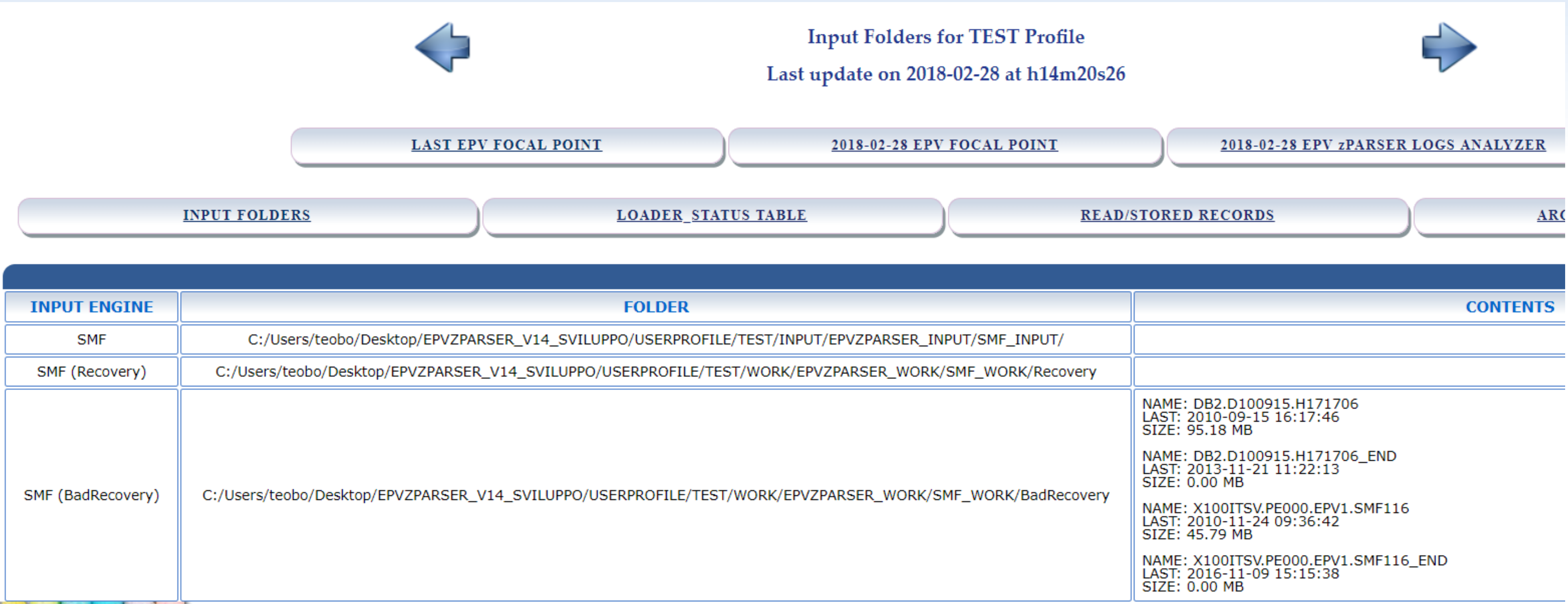
- When in Continuous mode, by default, an automatic process of data recovery is executed in case of errors

Data recovery - Recovery & BadRecovery

- This automatic process consists in moving the input file from the originary input folder to another input folder called “Recovery” where zParser will try again to elaborate the file
- If the problem occurs again, the file is being moved in a trash folder called “BadRecovery”
- Recovery and BadRecovery folders are provided for each Input Engine (SMF, DCO, IMS, …)

Data recovery - Recovery & BadRecovery

- So, having SMF files inside the “BadRecovery” folder always indicates that something went wrong



Input Folders for TEST Profile

Last update on 2018-02-28 at h14m20s26

INPUT ENGINE	FOLDER	CONTENTS
SMF	C:/Users/teobo/Desktop/EPVZPARSER_V14_SVILUPPO/USERPROFILE/TEST/INPUT/EPVZPARSER_INPUT/SMF_INPUT/	
SMF (Recovery)	C:/Users/teobo/Desktop/EPVZPARSER_V14_SVILUPPO/USERPROFILE/TEST/WORK/EPVZPARSER_WORK/SMF_WORK/Recovery	
SMF (BadRecovery)	C:/Users/teobo/Desktop/EPVZPARSER_V14_SVILUPPO/USERPROFILE/TEST/WORK/EPVZPARSER_WORK/SMF_WORK/BadRecovery	NAME: DB2.D100915.H171706 LAST: 2010-09-15 16:17:46 SIZE: 95.18 MB NAME: DB2.D100915.H171706_END LAST: 2013-11-21 11:22:13 SIZE: 0.00 MB NAME: X100ITSV.PE000.EPV1.SMF116 LAST: 2010-11-24 09:36:42 SIZE: 45.79 MB NAME: X100ITSV.PE000.EPV1.SMF116_END LAST: 2016-11-09 15:15:38 SIZE: 0.00 MB

Data recovery - Restart procedures

- When running in Once a day mode the following type of restart scenarios may happen:

  ✓Reloading the input files in error

  ✓Restarting the other EPV products processing

Data recovery - Restart procedures

- Reloading the input files in error:
  - ✓ Restart the ALLPHASES script

- Executing the single steps:
  - ✓ EPVzParserDbClear
  - ✓ EPVzParser_And_DbFill_ZOS
  - ✓ EPVzParserDbDeaccum_ZOS
  - ✓ NIGHTBATCH

Data recovery - Restart procedures

- Restarting the other EPV products processing (restart NIGHTBATCH)

- The following restart procedures are provided for Windows/Unix environments:
  - ✓ RESTART_NIGHTBATCH_xxx.BAT resumes all EPV phases from the step gone in error.
  - ✓ FORCE_NIGHTBATCH_xxx.BAT restarts all EPV phases from the beginning.
  - ✓ XXX can be: ZOS, Db2, ZGRAPH, MQ, ZLINUX

Data recovery - Restart procedures

- When running in Continuous mode the following type of restart scenarios may happen:

  - ✓Restarting the EPVzParserAgentsHandler
  - ✓Reloading files in error
  - ✓Restarting the other EPV products processing

Data recovery - Restart procedures

- If, for any reason, EPVzParserAgentsHandler is not in your system task manager, you need to start it again
- In case it is installed as system service:
  - ✓ Windows: open the Services panel, select EPVzParserAgentsHandler, then click on START
  - ✓ Unix: run command **service epv start**
- In case it is not installed as system service simply run EPVzParserAgentsHandler.BAT/sh under EPVROOT/USERPROFILE/PROFILENAME/EPVZPARSER/PROCS/AGENT_PROCS

Data recovery - Restart procedures

- If, for example, you need to apply some customization, you may need to restart the EPVzParserAgentsHandler
- In case it is installed as system service:
  - ✓ Windows: put the SHUTDOWN flag file in one of the input folders. Once you don’t see anymore the process in the Task Manager, open the Services panel, select EPVzParserAgentsHandler, click on STOP and then on START
  - ✓ Unix: put the SHUTDOWN flag file in one of the input folders. Once you don’t see anymore the process running, run the command **service epv stop**, then **service epv start**
- In case it is **not** installed as system service, simply put the RESTART flag file inside one of the input folders

Data recovery - Restart procedures

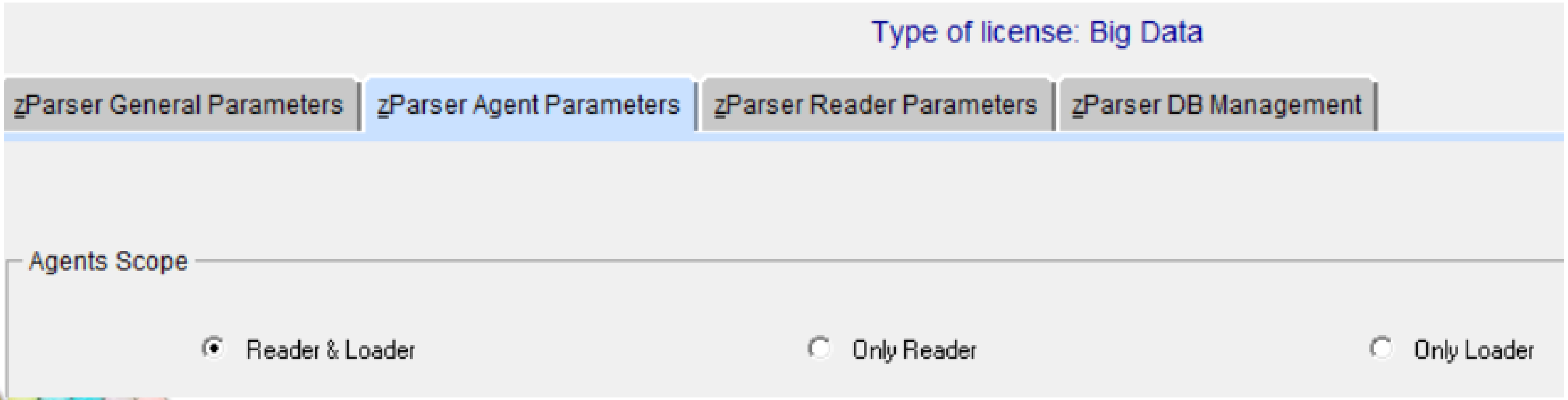
- As seen in the “Recovery & BadRecovery” slides, it may happen that some files are moved into the BadRecovery folder

- In these cases, maybe a fix from EPV is needed. Once the issue is fixed, users must manually move the files from the BadRecovery to an input folder

zParser for Big Data

zParser for Big Data - Parameters

- With Big Data license, some settings about the zParser architecture can be customized
- For example, users could make the reading phase on one server and the loading phase on another server



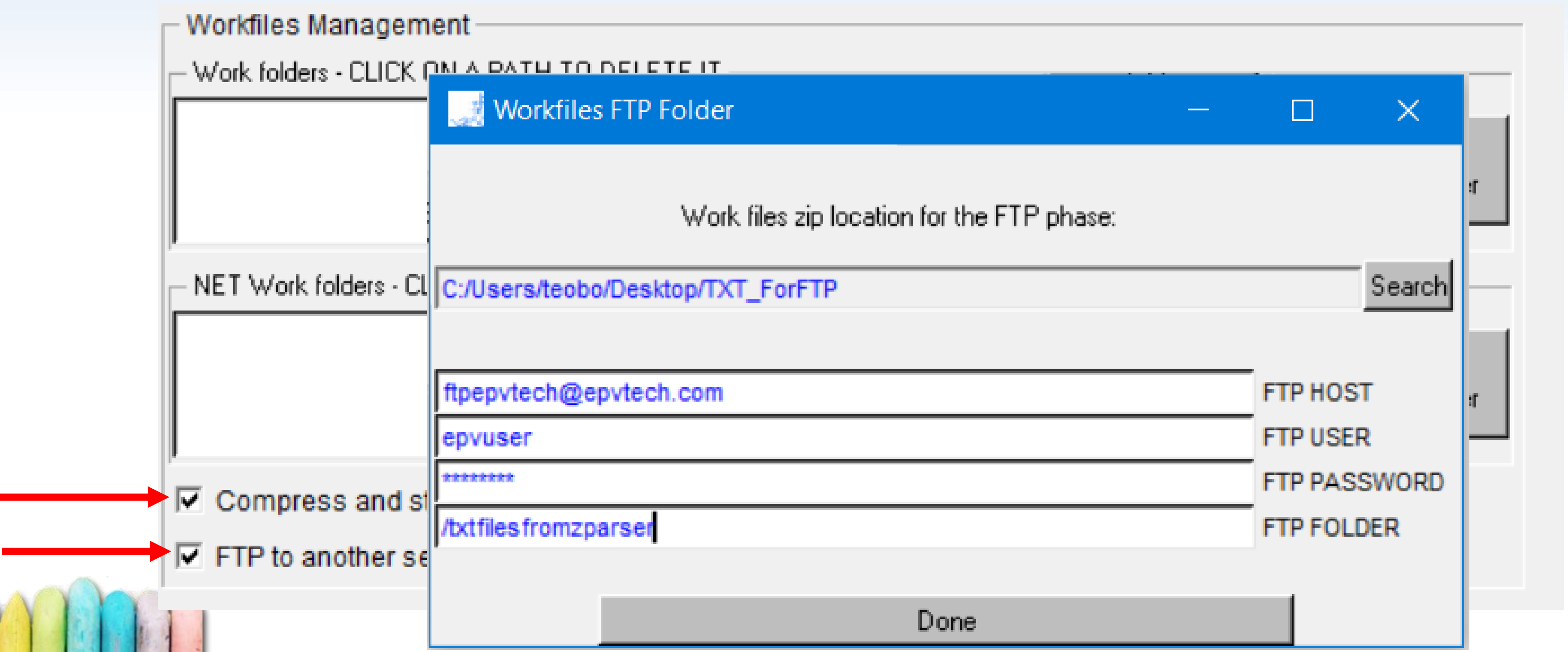
Type of license: Big Data

Agents Scope

[✓ Reader & Loader] Only Reader Only Loader

zParser for Big Data - Parameters

- This is possible because, after the reading phase, all the TXT/HDR files related to an input file can be compressed in a folder decided by the user and then also be sent via FTP


Workfiles Management

Work folders - CLICK ON A PATH TO DELETE IT

NET Work folders - Cl

☑ Compress and s

☑ FTP to another se

Workfiles FTP Folder

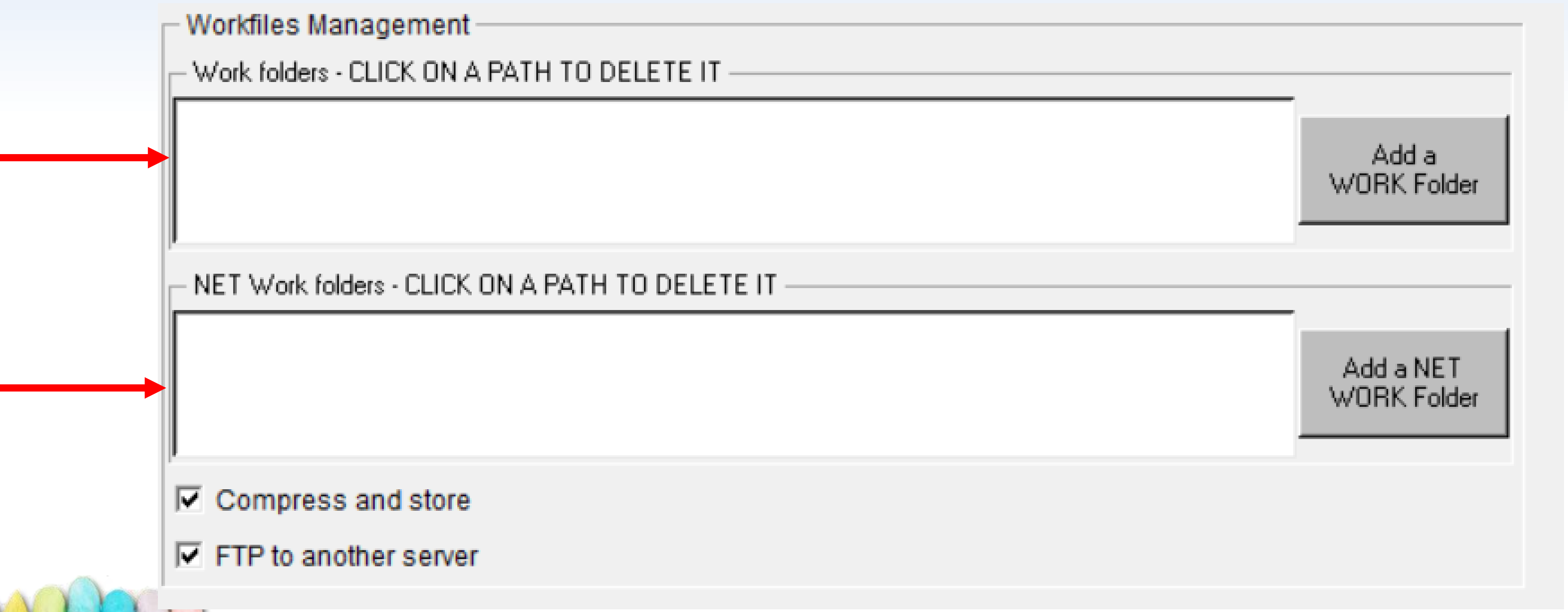
Work files zip location for the FTP phase:

C:/Users/teobo/Desktop/TXT_ForFTP [Search]

ftpepvtech@epvtech.com	FTP HOST
epvuser	FTP USER
*********	FTP PASSWORD
/txtfilesfromzparser	FTP FOLDER

zParser for Big Data - Parameters

- Splitted Work Areas allows the user to divide the I/O generated by the parsing process on multiple disks



Workfiles Management

Work folders - CLICK ON A PATH TO DELETE IT

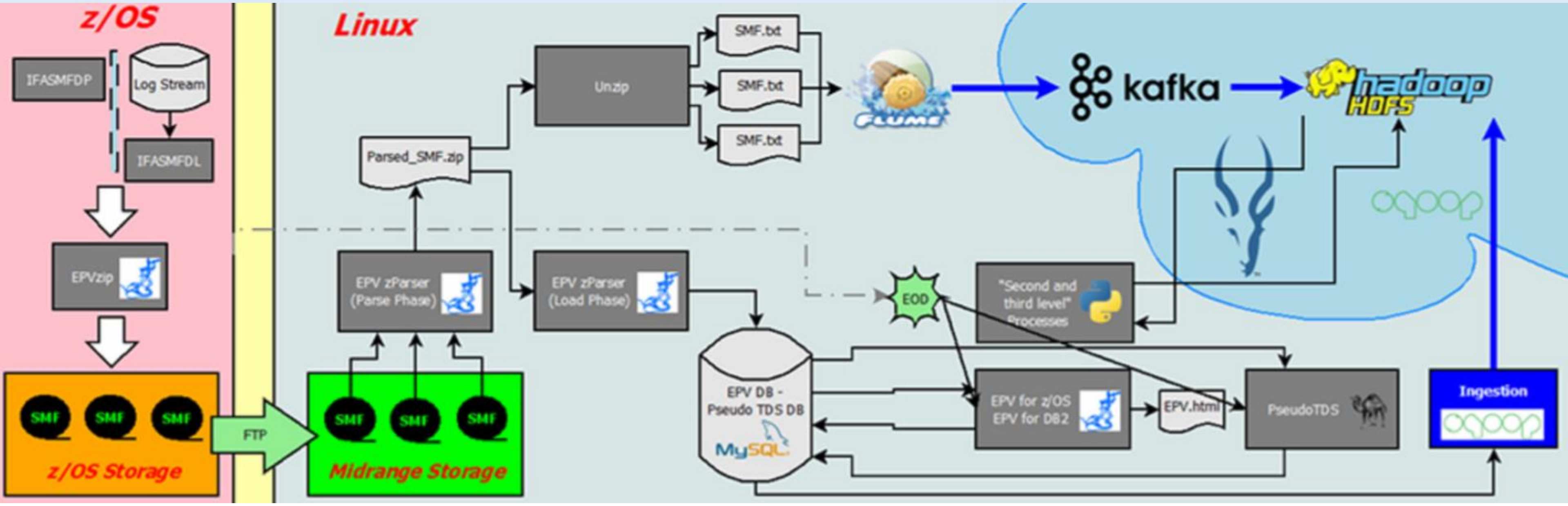
NET Work folders - CLICK ON A PATH TO DELETE IT

☑ Compress and store

☑ FTP to another server

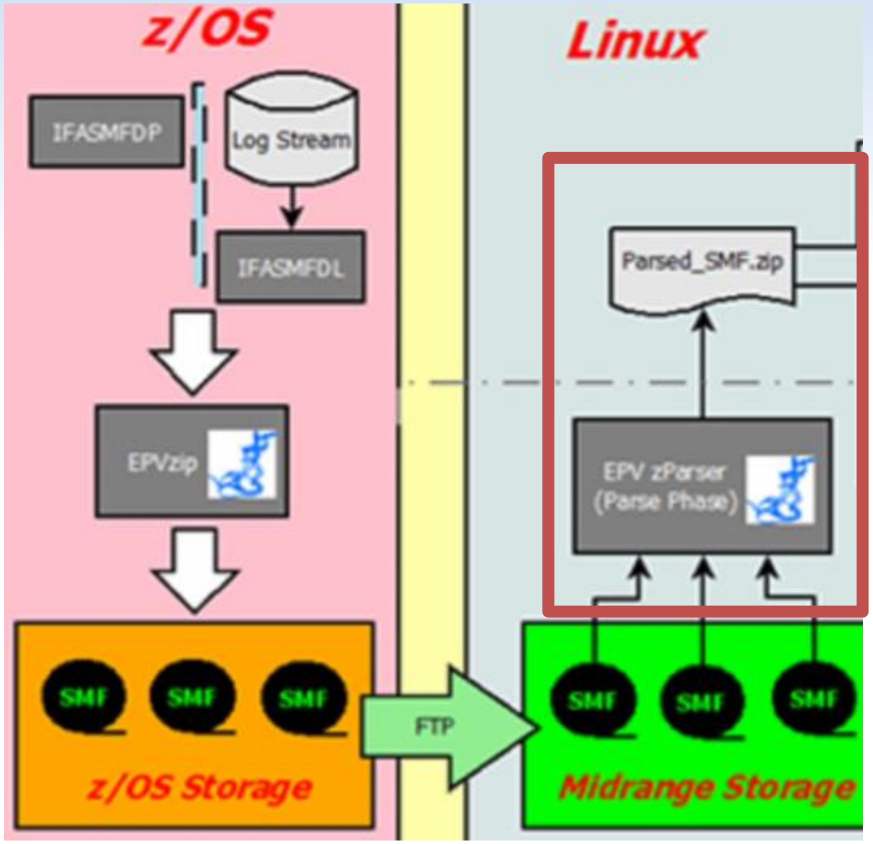
zParser for Big Data - Loading data in Hadoop

- Real life example at a customer's site



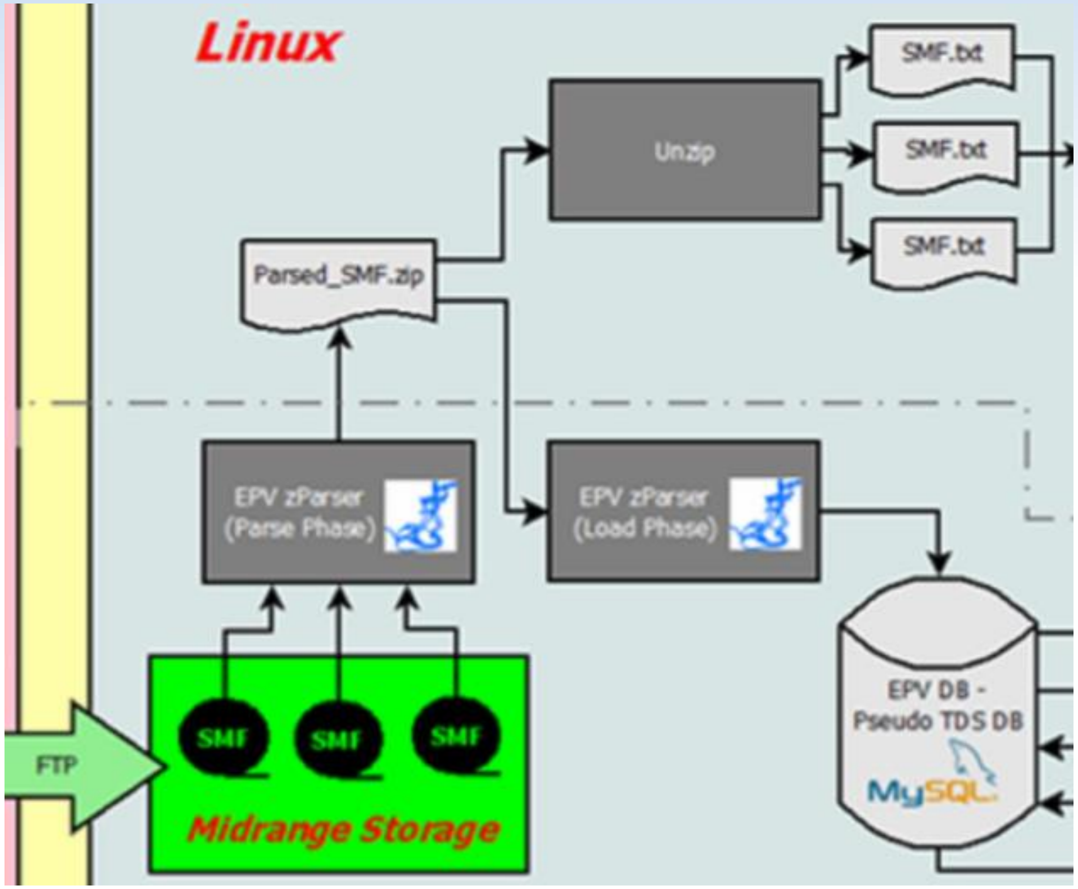
zParser for Big Data - Loading data in Hadoop

- Lots of input files produced during the day and sent to the EPV server where they start the reading phase and the TXT/HDR zip



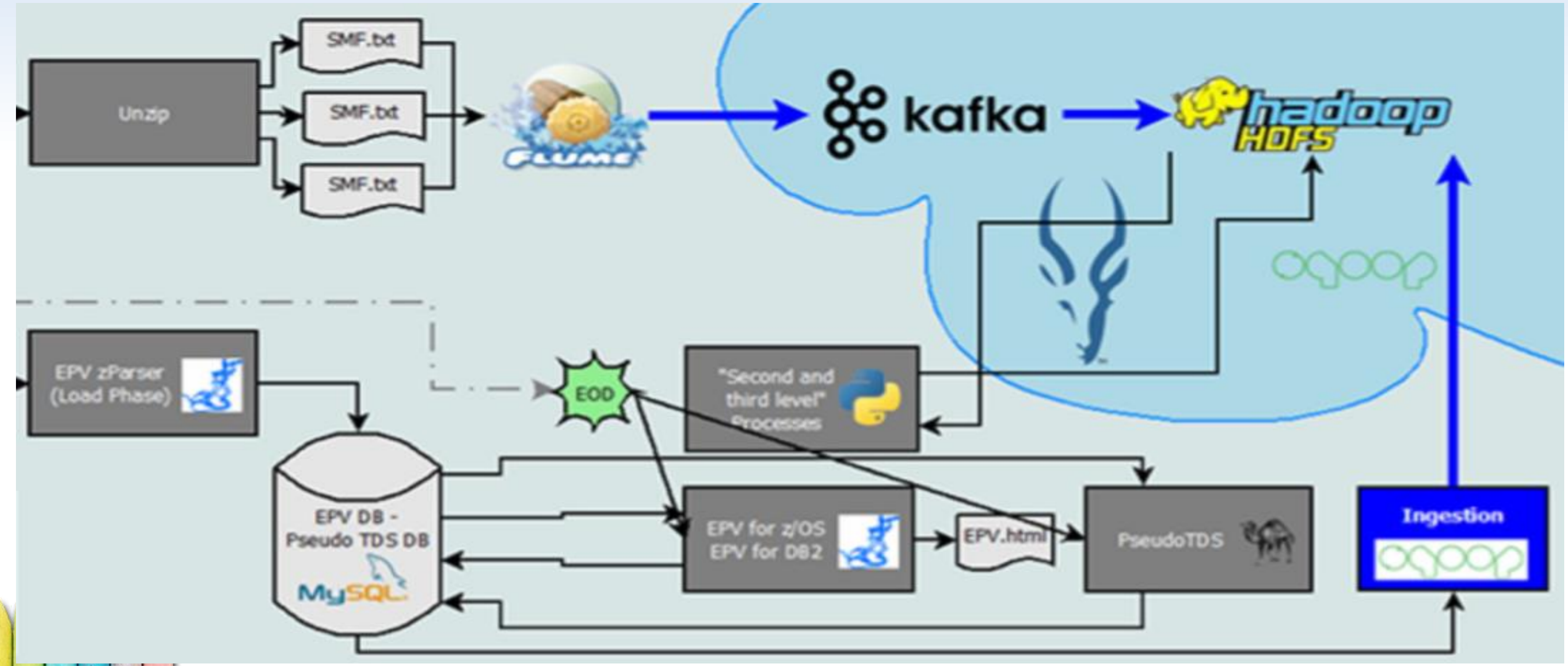
zParser for Big Data - Loading data in Hadoop

- After zip phase, two phases are executed in parallel: the loading phase in MySQL and the TXT/HDR unzip by Flume



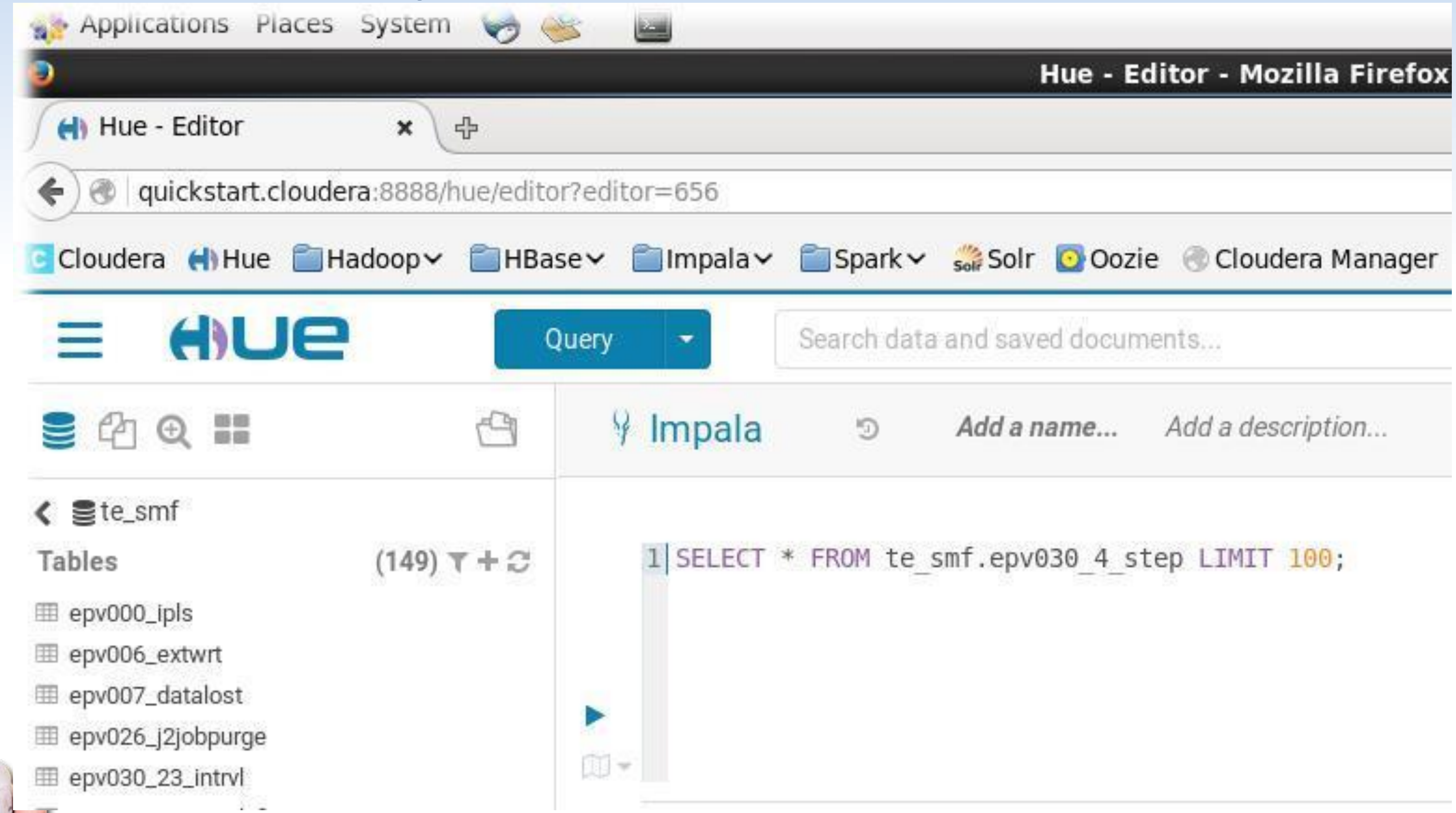
zParser for Big Data - Loading data in Hadoop

- On one side, Flume handles TXT/HDR flow through Kafka that physically inserts data in Hadoop's HDFS
- On the other side, the standard EPV flow is executed



zParser for Big Data - Loading data in Hadoop

- This is a sample query made with Impala on the data inserted in Hadoop's HDFS


Applications Places System
Hue - Editor - Mozilla Firefox
Hue - Editor
quickstart.cloudera:8888/hue/editor?editor=656
Cloudera Hue Hadoop HBase Impala Spark Solr Oozie Cloudera Manager
Query
Search data and saved documents...
Impala
Add a name...
Add a description...
te_smf
Tables (149)
epv000_ipls
epv006_extwrt
epv007_datalost
epv026_j2jobpurge
epv030_23_intrvl

```
SELECT * FROM te_smf.epv030_4_step LIMIT 100;
```

zParser for Big Data - Loading data in Hadoop

- In the near future, EPV is planning to support direct data ingestions in Impala

- This will be made in three steps:
  - ✓ TXT/HDR production during the reading phase
  - ✓ TXT/HDR move to Hadoop’s HDFS
  - ✓ TXT/HDR load in Impala by using the LOAD DATA INPATH command

z/OS Data Collection

End