IBM

IBM z/OS Version 2 Release 1 Technical Updates

- Summary of changes
- Examples of use
- Migration considerations

Karan Singh
Paul-Robert Hering
George Kozakos
Lutz Kuehner
Jacqueline Mourão
Diana Nakajima
Nelson Oliveira
Gil Peleg
Alvaro Salla
Yu Uchida
Antonio Vilarins

Redbooks

ibm.com/redbooks

International Technical Support Organization

IBM z/OS Version 2 Release 1 Technical Updates

August 2014

SG24-8140-00

Note: Before using this information and the product it supports, read the information in “Notices” on page xvii.

First Edition (August 2014)

This edition applies to Version 2 Release 1, Modification 0 of IBM z/OS (product number 5650-ZOS).

© Copyright International Business Machines Corporation 2014. All rights reserved.
Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices xvii
Trademarks xviii

Preface xix
Authors xix
Now you can become a published author, too xx
Comments welcome xxi
Stay connected to IBM Redbooks xxi

Chapter 1. Installation and migration 1
1.1 Planning for z/OS 2.1 installation 2
1.1.1 Hardware requirements 2
1.1.2 Identifying coupling facility requirements 3
1.1.3 Software requirements 3
1.1.4 Coexistence service requirements for z/OS 2.1 3
1.1.5 Installation considerations 6
1.2 Installation of z/OS 2.1 6
1.2.1 Installing with ServerPac 7
1.2.2 Installing with CBPDO 7
1.3 Driving system requirements for z/OS 2.1 7
1.3.1 Customized Offerings Driver V3 (5751-COD) 8
1.3.2 Identifying driving system requirements for z/OS 2.1 by using ServerPac 8
1.3.3 Identifying driving system requirements and installing a CBPDO 12
1.4 Elements changed in z/OS 2.1 13
1.4.1 Cryptographic Services 14
1.5 Preventive Service Planning buckets 14
1.6 Remove deleted data sets, paths, and references 15
1.7 Documentation and references 16

Chapter 2. IBM Health Checker for z/OS 17
2.1 The auto-start feature 18
2.1.1 Invocation and use 18
2.1.2 Installation 19
2.1.3 Migration and coexistence considerations 20
2.2 HZSPRINT PARM longer than 100 characters 21
2.2.1 Invocation and use 21
2.2.2 Migration and coexistence considerations 21
2.3 Refreshed METAL C sample health checks 22
2.3.1 Invocation and use 22
2.4 System name in message buffer 22
2.4.1 Invocation and use 22
2.5 The and updated checks of IBM Health Checker 23
2.5.1 or updated checks by component or subsystem 23
2.5.2 Use of the system health checks 28
2.5.3 Installation 29
2.5.4 Migration and coexistence considerations 29
2.6 Documentation and reference 29

Chapter 3. Virtual lookaside facility 31

© Copyright IBM Corp. 2014. All rights reserved.

3.1 MODIFY VLF command 32
3.1.1 Invocation and use 32
3.2 VLF PARMLIB concatenation 32
3.2.1 Invocation and use 32
3.2.2 Migration and coexistence considerations 33
3.3 The VLF Health Check 33
3.3.1 Invocation and use 33
3.4 Documentation and reference 34

Chapter 4. System Modification Program/Extended 35
4.1 SMP/E ISPF dialog split screen display 36
4.1.1 Installation 37
4.1.2 Invocation and use 37
4.2 Documentation and reference 38

Chapter 5. Dynamic Channel Path Management 41
5.1 DCM support for cascaded FICON switches 42
5.1.1 Invocation and use 42
5.1.2 Installation 45
5.1.3 Interactions and dependencies 45
5.1.4 Migration and coexistence considerations 45
5.2 Documentation and reference 46

Chapter 6. Hardware configuration and management console changes 47
6.1 I/O discovery and autoconfiguration 48
6.1.1 Discovery by serial number 48
6.1.2 Directly attached (non-switched) controllers in discovery 49
6.1.3 User-assigned control unit and device numbers 49
6.1.4 Channel path or switch inclusion and exclusion lists 51
6.1.5 Policy change between controller discoveries 55
6.1.6 Toleration of incapable systems 56
6.1.7 Save discovery and auto-definition results 56
6.2 Verify Configuration function with I/O autoconfiguration 56
6.2.1 Invocation and use 57
6.2.2 Interactions and dependencies 60
6.3 HMC-wide activation of dynamic I/O changes 60
6.3.1 Installation 61
6.3.2 Invocation and use 63
6.4 Unconditional generation of DR site OS configurations 66
6.4.1 Invocation and use 66
6.5 HCD validation 67
6.5.1 Alert for ACTIVATE command with NOVALIDATE 67
6.5.2 Alert when CF CIB connection changes for connectivity updates 67
6.6 HCD OS group change action for device groups 67
6.6.1 Invocation and use 67
6.7 HCD batch utility 68
6.7.1 Activate command now available as HCD batch command 68
6.7.2 Filter parameters for graphical reports that are created by a batch utility 69
6.8 HCD support of PCIe functions 69
6.8.1 PCIe function dialog 70
6.8.2 FUNCTION IOCP Statement 73
6.8.3 Physical Network ID keyword added for OSD and IQD CHPID types 73
6.8.4 PCIe Function Summary Report 74
6.8.5 PCIe Function Compare Report 75

6.9 PCHID Summary Report 76
6.9.1 Invocation and use 76
6.10 OFFLINE parameter in the MVS Device Detail Report 77
6.10.1 Invocation and use 77
6.11 HCD CF Channel Path Connectivity List 77
6.11.1 Invocation and use 77
6.12 HCD hardware support 78
6.13 HCM port number status line 78
6.13.1 Invocation and use 78
6.14 Connecting CHPIDs to control units in HCM 79
6.15 HCM explanatory message on load production IODF 79
6.15.1 Invocation and use 80
6.16 Documentation and reference 80

Chapter 7. Base Control Program Internal Interface (BCPii) 81
7.1 BCPii support for REXX 82
7.1.1 Invocation and use 83
7.2 Setting up a BCPii REXX environment 85
7.2.1 BCPii services that are available from REXX 85
7.2.2 Sample codes 86
7.3 BCPii performance improvement for HWILIST and HWIQUERY 94
7.3.1 Invocation and use 94
7.3.2 Interactions and dependencies 94
7.4 Documentation and reference 94

Chapter 8. SDSF: System Display and Search Facility 95
8.1 64-bit support 96
8.1.1 Invocation and use 96
8.2 8-character job class 96
8.2.1 Invocation and use 96
8.3 Print enhancements 99
8.3.1 Invocation and use 100
8.4 Extended console name for the user log 102
8.4.1 Invocation and use 103
8.5 Sort enhancement 104
8.5.1 Invocation and use 104
8.5.2 Migration and coexistence considerations 106
8.6 SET DISPLAY enhancement 107
8.6.1 Invocation and use 107
8.7 JESPlex scoping 108
8.7.1 Invocation and use 108
8.8 System symbols and filtering support 109
8.8.1 Invocation and use 109
8.9 REXX and Java enhancements (BROWSE and LOG) 111
8.9.1 Invocation and use 111
8.10 Security Assist SECTRACE facility 113
8.10.1 Invocation and use 114
8.11 Migration and coexistence considerations 116
8.12 Documentation and reference 116

Chapter 9. Security server 117
9.1 Removal of the BPX.DEFAULT.USER profile 118
9.1.1 Invocation and use 118
9.2 RACF database unload utility record type 121

9.2.1 Invocation and use 121
9.3 RACF Remote Sharing Facility TCP/IP support for TLS V1.2. 123
9.3.1 Invocation and use 123
9.3.2 Interactions and dependencies 123
9.4 RRSF TCP/IP support for IPv6 124
9.4.1 Invocation and use 124
9.4.2 Interactions and dependencies 125
9.4.3 Migration and coexistence considerations 125
9.5 Comments and blank lines now allowed in the RACF parameter library 125
9.5.1 Invocation and use 126
9.5.2 Migration and coexistence considerations 127
9.6 Enterprise PKCS#11 support 127
9.6.1 Invocation and use 128
9.6.2 Interactions and dependencies 130
9.6.3 Migration and coexistence considerations 130
9.7 RACDCERT displays additional information for digital certificate chains 130
9.7.1 Invocation and use 130
9.8 Prevent deletion of a certificate with a pending request 136
9.8.1 Invocation and use 137
9.9 Documentation and reference 138

Chapter 10. PKI Services 139
10.1 Enterprise PKCS #11 support 140
10.1.1 Invocation and use 140
10.1.2 Interactions and dependencies 141
10.1.3 Migration and coexistence considerations 141
10.2 Extended Validation Certificate support 141
10.2.1 Invocation and use 142
10.2.2 Interaction and dependencies 142
10.3 Granular certificate administration control. 142
10.3.1 Invocation and use 142
10.3.2 Migration and coexistence 143
10.4 Enhancement for Basic Constraints extension 144
10.4.1 Invocation and use 144
10.5 IBM HTTP Server V7.0 support 145
10.5.1 Invocation and use 145
10.6 More about Certificate Revocation List processing 146
10.6.1 Invocation and use 146
10.7 Documentation and reference 147

Chapter 11. System SSL 149
11.1 Enhancements for TLS V1.2 150
11.1.1 Cipher suites 150
11.1.2 TLS handshake extension support 151
11.1.3 Invocation and use 153
11.1.4 Interactions and dependencies 154
11.1.5 Migration and coexistence considerations 154
11.2 Enhancement for Suite B cryptography for TLS 154
11.2.1 Invocation and use 155
11.2.2 Interactions and dependencies 156
11.3 RFC 5280 certificate validation support 156
11.3.1 Invocation and use 156
11.4 SAF key ring certificate chain validation support 157

11.4.1 Invocation and use 158
11.5 DSA 2048-bit key support 159
11.5.1 Invocation and use 159
11.5.2 Migration and coexistence considerations 161
11.6 System SSL applications in FIPS mode using ICSF functions 161
11.6.1 Invocation and use 161
11.6.2 Migration and coexistence considerations 162
11.7 Enterprise PKCS #11 support 162
11.7.1 Invocation and use 162
11.8 Documentation and reference 163

Chapter 12. z/OS Batch Runtime 165
12.1 Overview of Batch Runtime changes 166
12.2 Support for launching PL/I applications 166
12.2.1 Invocation and use 167
12.2.2 Installation 169
12.3 IBM batch programming model support 169
12.3.1 Invocation and use 170
12.3.2 Installation 171
12.4 Documentation and reference 171

Chapter 13. Consoles and auto-reply 173
13.1 Dynamically add and delete consoles 174
13.1.1 Invocation and use 174
13.1.2 Installation 178
13.1.3 Migration and coexistence considerations 178
13.2 z/OS support for the Integrated 3270 Console interface 178
13.2.1 Invocation and use 179
13.2.2 Installation 185
13.2.3 Migration and coexistence considerations 185
13.3 Enhanced synchronous WTOR notification 185
13.3.1 Invocation and use 185
13.3.2 Installation 186
13.4 FORCE TCB command 187
13.4.1 Invocation and use 187
13.5 Auto-reply support for synchronous WTOR messages 188
13.5.1 Invocation and use 188
13.5.2 Installation 191
13.6 Auto-reply rate limit 191
13.6.1 Invocation and use 191
13.6.2 Migration and coexistence considerations 192
13.7 Documentation and reference 192

Chapter 14. Generic Tracker Facility 193
14.1 Generic Tracker 194
14.1.1 Invocation and use 194
14.1.2 Installation 199
14.1.3 Migration and coexistence considerations 199
14.2 Documentation and reference 200

Chapter 15. Job Entry Subsystem 2 (JES2) 201
15.1 In-stream symbolic substitution 202
15.1.1 Invocation and use 202
15.1.2 Migration and coexistence considerations 204

15.2 Passing symbols by using the internal reader 204
15.2.1 Invocation and use 204
15.2.2 Migration and coexistence considerations 206
15.3 JES Symbol Service 206
15.4 Requesting job notification through ENF 78 207
15.4.1 Invocation and use 207
15.4.2 Interactions and dependencies 208
15.5 8-character job classes 208
15.5.1 Invocation and use 208
15.5.2 Interactions and dependencies 211
15.5.3 Migration and coexistence considerations 211
15.6 SAF controls on input class use 211
15.6.1 Invocation and use 212
15.7 JES2-related enhancements to JCL 212
15.7.1 Invocation and use 212
15.8 Improved processing of input phase errors 215
15.8.1 Invocation and use 215
15.9 Interpretation with conversion 216
15.9.1 Invocation and use 216
15.9.2 Migration and coexistence considerations 218
15.10 Controlling where a job converts 218
15.10.1 Invocation and use 218
15.10.2 Migration and coexistence considerations 218
15.11 Displaying JES2 initialization information 218
15.11.1 Invocation and use 218
15.12 Performance improvements for SYSOUT processing 219
15.12.1 Invocation and use 219
15.12.2 Installation 220
15.12.3 Migration and coexistence considerations 220
15.13 SSI 85 - Unauthorized services for spool access 220
15.13.1 Invocation and use 221
15.13.2 Interactions and dependencies 222
15.13.3 Migration and coexistence considerations 222
15.14 Documentation and reference 222

Chapter 16. Job Entry Subsystem 3 (JES3) 223
16.1 Dynamic spool deletion 224
16.1.1 Invocation and use 224
16.1.2 Migration and coexistence considerations 224
16.2 Enhanced spool use data 224
16.2.1 Invocation and use 224
16.3 In-stream data in JCL procedures 226
16.3.1 Invocation and use 227
16.3.2 Migration and coexistence considerations 227
16.4 Multiple simultaneous logons in a Parallel Sysplex 227
16.4.1 Invocation and use 227
16.5 8-character job classes on the JOB card 228
16.5.1 Invocation and use 228
16.5.2 Migration and coexistence considerations 228
16.6 SAF control over job class 228
16.6.1 Invocation and use 229
16.6.2 Migration and coexistence considerations 229
16.7 JES3-related enhancements to JCL 229

16.7.1 Invocation and use 229
16.8 ENF 70 events 230
16.8.1 Invocation and use 230
16.8.2 Migration and coexistence considerations 231
16.9 Documentation and reference 231

Chapter 17. Cross-system coupling facility and cross-system extended services . 233
17.1 Serial rebuild with structure priority change 234
17.1.1 Invocation and use 235
17.1.2 Installation 236
17.1.3 Interactions and dependencies 238
17.1.4 Migration and coexistence 238
17.2 Documentation and reference 238

Chapter 18. Global resource serialization 239
18.1 ISGENQ macro, REQUEST=CHANGE 240
18.1.1 Invocation and use 240
18.1.2 Interactions and dependencies 241
18.1.3 Migration and coexistence considerations 242
18.2 FICON channel-to-channel connections 242
18.2.1 Invocation and use 242
18.2.2 Installation 242
18.2.3 Migration and coexistence considerations 243
18.3 Documentation and reference 243

Chapter 19. System Management Facilities 245
19.1 SMF 30 Instruction Counts 246
19.1.1 Invocation and use 246
19.1.2 Installation 248
19.1.3 Interactions and dependencies 248
19.2 IFASEXIT to stop reading before the end of the logstream 248
19.2.1 Invocation and use 249
19.2.2 Migration and coexistence 250
19.3 DSPSIZEMAX for SMF logstreams 251
19.3.1 Invocation and use 251
19.3.2 Migration and coexistence 252
19.4 Documentation and reference 252

Chapter 20. Resource Measurement Facility 253
20.1 zIIP enablement 254
20.1.1 Invocation and use 254
20.1.2 Installation 254
20.1.3 Interactions and dependencies 255
20.1.4 Migration and coexistence 255
20.2 XML support for additional RMF postprocessor reports 255
20.2.1 Invocation and use 258
20.3 RMF zHybrid SMF 261
20.3.1 Invocation and use 265
20.3.2 Interactions and dependencies 267
20.3.3 Installation 267
20.4 RMF zHybrid Microsoft Windows support 267
20.4.1 Invocation and use 268
20.4.2 Installation 271
20.5 Documentation and reference 272

Chapter 21. IBM Time Sharing Option Extensions and TSO/E REXX 273
21.1 TSO/E logon failure messages 274
21.1.1 Invocation and use 274
21.2 TSO/E REXX EXECIO enhancements 275
21.2.1 Invocation and use 275
21.3 TSO/E REXX TRAPMSG function 277
21.3.1 Invocation and use 277
21.4 64-bit support for the REXX STORAGE function 280
21.4.1 Invocation and use 280
21.5 Enhanced TSO/E REXX LISTDSI function 281
21.5.1 Invocation and use 281
21.6 Multiple TSO/E login support for JES3 286
21.7 TSO/E LOGOFF updates RACF TSO segment only on a change 286
21.8 Documentation and reference 286

Chapter 22. Workload manager 287
22.1 Extend IWMCLSFY macro interface to AMODE64 288
22.1.1 Invocation and use 288
22.2 Performance blocks (PBs) above the 2 GB bar 288
22.2.1 Invocation and use 288
22.2.2 Migration and coexistence considerations 289
22.3 Fast path classification 289
22.3.1 Invocation and use 290
22.4 Workload Manager I/O priority group 290
22.4.1 Invocation and use 290
22.4.2 Migration and coexistence considerations 291
22.5 Improved accuracy for the IWMEQTME service 292
22.5.1 Invocation and use 292
22.5.2 Migration and coexistence considerations 293
22.6 More Workload Manager application environments 293
22.6.1 Invocation and use 293
22.6.2 Installation 294
22.6.3 Migration and coexistence considerations 294
22.7 Documentation and reference 294

Chapter 23. IBM Data Facility Sort 295
23.1 Functional enhancements 296
23.1.1 Invocation and use 296
23.1.2 Migration and coexistence 298
23.2 RAS enhancements 299
23.2.1 Invocation and use 299
23.3 Dynamic RAS enhancements 300
23.3.1 Invocation and use 300
23.3.2 Migration and coexistence 302
23.4 Support for 64-Bit callers with 64-bit addressed records 302
23.4.1 Invocation and use 302
23.4.2 Interactions and dependencies 302
23.4.3 Migration and coexistence 303
23.4.4 Documentation and reference 303

Chapter 24. z/OS Binder feature 305
24.1 Boundary alignment 306
24.1.1 Invocation and use of ALIGNT control statement 306
24.1.2 Invocation and use of alignment from ESD records 307

24.1.3 Interaction and dependencies 309
24.1.4 Migration and coexistence 309
24.2 Symbol resolution tracing 309
24.2.1 Invocation and use 309
24.3 API DLL data sets resident support 311
24.3.1 Invocation and use 312
24.3.2 Installation 313
24.3.3 Migration and coexistence considerations 313
24.4 LONGPARM option 313
24.4.1 Invocation and use 313
24.4.2 Migration and coexistence 314
24.5 Documentation and reference 314

Chapter 25. z/OS XL C/C++ 315
25.1 Architecture use 316
25.1.1 Invocation and use 316
25.1.2 Interactions and dependencies 318
25.1.3 C11 and C11++ feature enhancements 318
25.1.4 Complex type initialization. 318
25.1.5 C11: Generic-type generics 320
25.1.6 C11: Function never returns specifier 320
25.1.7 C11: Static assertions 321
25.1.8 C++11: Explicit conversion operators 322
25.1.9 C++11: Scoped enums 323
25.1.10 C++11: Generalized constant expression 323
25.2 Command line include files 325
25.2.1 Invocation and use 325
25.3 Metal C: SYSSTATE macro enhancements 325
25.3.1 Invocation and use 326
25.3.2 Migration and coexistence considerations 326
25.4 Metal C: User-nominated “main” function 326
25.4.1 Invocation and use 327
25.5 Hardware use and tuning 327
25.5.1 Invocation and use 329
25.5.2 Interactions and dependencies 329
25.6 Decimal floating point and zoned conversions 330
25.6.1 Invocation and use 330
25.7 Expected path selection 330
25.7.1 Invocation and use 330
25.8 Packed decimal built-in functions 331
25.8.1 Invocation and use 331
25.9 [NO]THREADED option 331
25.9.1 Invocation and use 331
25.10 Metal C: AMODE-switching support at IPA LINK 332
25.10.1 Invocation and use 332
25.11 DEBUG(LEVEL) option to debug optimized code 333
25.11.1 Invocation and use 333
25.12 Debugging 333
25.12.1 Debugging inlined functions 334
25.12.2 Relative statement number 334
25.13 Documentation and reference 335

Chapter 26. Language Environment 337

26.1 Heap overlay tolerance 338
26.1.1 Invocation and use 338
26.1.2 Interactions and dependencies 339
26.2 Remove Language Environment USERMODs options 339
26.2.1 Interactions and dependencies 339
26.2.2 Migration and coexistence considerations 340
26.3 Runtime system symbol access 341
26.3.1 Invocation and use 341
26.4 AMODE 64 support for 1M and 2G large pages 344
26.4.1 Invocation and use 344
26.4.2 Interactions and dependencies 346
26.5 Nested CEEPIPI environments 346
26.5.1 Invocation and use 346
26.5.2 Interactions and dependencies 346
26.6 BSAM type=blocked support 347
26.6.1 Invocation and use 347
26.6.2 Interactions and dependencies 348
26.7 N1040 functions 348
26.7.1 Invocation and use 348
26.8 Standard I/O extensions 350
26.8.1 Invocation and use 350
26.9 Unicode auto-conversion support 352
26.9.1 Invocation and use 352
26.10 C11 validation Items 352
26.10.1 Migration and coexistence considerations 354
26.11 C11 alignment-related support 354
26.11.1 Invocation and use 354
26.11.2 Migration and coexistence considerations 355
26.12 Documentation and reference 355

Chapter 27. Resource Recovery Services 357
27.1 RSS Internal Cold Start feature 358
27.1.1 Invocation and use 358
27.1.2 Interactions and dependencies 360
27.2 Documentation and reference 360

Chapter 28. Base Control supervisor and timer supervisor changes 361
28.1 Transaction execution 362
28.1.1 Overview 362
28.1.2 Invocation and use 363
28.1.3 Interactions and dependencies 370
28.1.4 Installation 371
28.2 Offload time at task level 371
28.2.1 Invocation and use 371
28.2.2 Migration and coexistence 372
28.3 Larger and extendable facility list 372
28.3.1 Invocation and use 372
28.4 Tracing disabled interrupt exits 373
28.4.1 Invocation and use 373
28.5 Entry table entries expanded 373
28.5.1 Invocation and use 373
28.6 Support for clock values in the second UNIX epoch 373
28.6.1 Invocation and use 374

28.7 Reserve SVC entries to use yet maintain ABEND upon use 374
28.7.1 Invocation and use 375
28.8 ACCURACY keyword in CLOCKxx 375
28.8.1 Installation 375
28.8.2 Invocation and use 375
28.8.3 Migration and coexistence considerations 376
28.9 Documentation and reference 377

Chapter 29. Scheduler and Device Allocation 379
29.1 Dynamic SYSDSN ENQ downgrade 380
29.1.1 Invocation and use 381
29.1.2 Migration and coexistence considerations 382
29.2 Parallel batch recall 382
29.2.1 Invocation and use 382
29.3 PARMDD keyword to pass a parameter of up to 32,000 characters 384
29.3.1 Invocation and use 384
29.3.2 Interactions and dependencies 385
29.3.3 Migration and coexistence 385

Chapter 30. Recovery Termination Manager 387
30.1 CALLRTM TYPE=SRBTERM service 388
30.1.1 Invocation and use 388
30.1.2 Migration and coexistence 391
30.2 SdwaReleaseCode 391
30.2.1 Invocation and use 391
30.2.2 Migration and coexistence 391
30.3 64-bit FRR retry address in the system trace 391
30.3.1 Invocation and use 392
30.4 Cross-memory failing instruction for the ESTAEX macro 392
30.5 SDWAMABD added to SDWANMFS 392
30.5.1 Invocation and use 393
30.6 ASID added to the VRA for ABEND30D 393
30.6.1 Invocation and use 394
30.7 Documentation and reference 394

Chapter 31. Service aids 395
31.1 IPCS append dump directory 396
31.1.1 Invocation and use 396
31.2 IPCS SYSTRACE enhancement 398
31.2.1 Invocation and use 399
31.3 System trace branch and mode health check 400
31.3.1 Invocation and use 401
31.4 SLIP health check 403
31.4.1 Invocation and use 404
31.5 SLIP zero address detection support 405
31.5.1 Avoiding non-error ZAD events 405
31.5.2 Invocation and use 406
31.6 Documentation and reference 409

Chapter 32. Hardware Instrumentation Services 411
32.1 HIS enhancements 412
32.2 HISSERV interface 412
32.2.1 Invocation and use 413
32.2.2 Installation 418

32.2.3 Interactions and dependencies 423
32.3 SMF 113 Changes 423
32.3.1 Invocation and use 424
32.4 Documentation and Reference 424

Chapter 33. System REXX 425
33.1 Running System REXX via the terminal monitor program 426
33.1.1 Installation 426
33.1.2 Invocation and use 426
33.2 Documentation and reference 427

Chapter 34. IBM Interactive System Productivity Facility 429
34.1 Editor Unicode support 430
34.1.1 Invocation and use 430
34.1.2 Interactions and dependencies 431
34.2 Editor compare enhancements 432
34.2.1 Invocation and use 432
34.3 XTIOT support 433
34.3.1 Invocation and use 433
34.3.2 Interactions and dependencies 433
34.3.3 Installation 433
34.4 Directory and member list enhancements 434
34.4.1 Invocation and use 434
34.5 Swapbar visibility enhancement 436
34.5.1 Invocation and use 436
34.6 ISPF support for PDSE version 2 437
34.6.1 Invocation and use 437
34.7 Multiple logical screens when starting ISPF 438
34.7.1 Invocation and use 438
34.8 Edit find and change support for regular expressions 440
34.8.1 Invocation and use 440
34.9 ISPF support for scroll amounts greater than 9999 441
34.9.1 Invocation and use 442
34.9.2 Migration and coexistence 442
34.10 ISPF edit hilite command enhancement for JCL 442
34.10.1 Invocation and use 443
34.11 ISPF filtering of z/OS UNIX path names 443
34.11.1 Invocation and use 443
34.12 Expanded member count field 445
34.12.1 Invocation and use 445
34.13 Default keyword for memlist service 446
34.13.1 Invocation and use 446
34.14 ISPF editor expandable command field 446
34.14.1 Invocation and use 447
34.15 Enhancement for the udlist command 447
34.15.1 Invocation and use 447
34.16 Free unused space for multi-volume data sets 448
34.16.1 Invocation and use 448
34.17 External data commands support ASCII and UTF-8 448
34.17.1 Invocation and use 448
34.18 z/OS directory list utility 449
34.18.1 Invocation and use 449
34.19 Documentation and reference 452

Chapter 35. z/OS UNIX System Services 453
35.1 Pipes enhancements 454
35.1.1 Invocation and use 454
35.1.2 Migration and coexistence 459
35.2 The zlsof command 459
35.2.1 Invocation and use 460
35.3 I/O count correction in SMF records 463
35.3.1 Invocation and use 464
35.3.2 Migration and coexistence 465
35.4 Automount enhancements 465
35.4.1 Invocation and use 466
35.4.2 Migration and coexistence 469
35.5 TFS enhancements 469
35.5.1 Invocation and use 469
35.5.2 Installation 471
35.6 Unicode services use 472
35.6.1 Invocation and use 472
35.7 REXX enhancements 476
35.7.1 Invocation and use 477
35.8 Avoiding access loss for previous default users 483
35.8.1 Invocation and use 483
35.9 Documentation and reference 487

Chapter 36. dbx, the IBM z/OS debugger for UNIX source code 489
36.1 Debug optimized code 490
36.1.1 Invocation and use 490
36.1.2 Migration and coexistence 496
36.2 Documentation and reference 496

Chapter 37. z/OS UNIX shell and utilities 497
37.1 ASCII or Unicode option on a subset of the utilities 498
37.1.1 Invocation and use 498
37.1.2 Migration and coexistence 502
37.2 Documentation and reference 502

Chapter 38. Enhancements to zFS, the IBM System z file system 503
38.1 zFS extended or large directory support 504
38.1.1 Invocation and use 504
38.1.2 Migration and coexistence 512
38.1.3 Installation 513
38.2 Improved zFS salvager 513
38.2.1 Invocation and use 514
38.2.2 Interactions and dependencies 515
38.2.3 Migration and coexistence considerations 515
38.3 Enhancements for zFS quiesce and unquiesce processing 516
38.3.1 Invocation and use 516
38.3.2 Interactions and dependencies 518
38.4 zFS backup bit 518
38.5 zFS auditfid default setting 519
38.5.1 Migration and coexistence 519
38.6 Elimination of multi-file system aggregates and cloning 520
38.6.1 Invocation and use 520
38.6.2 Migration and coexistence 520

38.7 Renamed z/OS file system 521
38.8 Using non-SMS zFS aggregates with extended addressability. 521
38.8.1 Invocation and use 521
38.8.2 Interactions and dependencies 523
38.9 Documentation and reference 523

Chapter 39. SMB: Server Message Block 525
39.1 Support for Microsoft Windows Server 2008. 526
39.1.1 Invocation and use 526
39.1.2 Interactions and dependencies 526
39.1.3 Migration and coexistence 526
39.2 Documentation and reference 527

Chapter 40. Network File System. 529
40.1 NFS server reregistration with RPCBIND 530
40.1.1 Invocation and use 530
40.1.2 Interactions and dependencies 530
40.2 Documentation and reference 530

Chapter 41. z/OS Font Collection 531
41.1 Font library 532
41.2 Invocation and use 532
41.2.1 Installation 532
41.2.2 Migration and coexistence 533

Appendix A. Material available to download. 535
Finding the web material 535
Using the web material. 535

Related publications 537
IBM Redbooks 537
Documentation 538
Online resources 538
Help from IBM 538

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

© Copyright IBM Corp. 2014. All rights reserved.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AIX®
APL2®
CICS®
DB2®
DS8000®
FICON®
GDPS®
Geographically Dispersed Parallel Sysplex™
IBM®
IMS™
Language Environment®
MVS™
OS/390®
Parallel Sysplex®
Print Services Facility™
ProductPac®
RACF®
Rational®
RDN®
Redbooks®
Redbooks (logo) ®
Resource Measurement Facility™
RETAIN®
RMF™
S/390®
System p®
System x®
System z®
System z10®
System z9®
Tivoli®
UC™
WebSphere®
z/OS®
z/VM®
z/VSE®
z10™
z9®
zEnterprise®

The following terms are trademarks of other companies:

PostScript, the Adobe logo, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

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

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Preface

This IBM® Redbooks® publication provides a broad understanding of the changes, features, and functions introduced with IBM z/OS® Version 2 Release 1 (2.1).

This version marks a era of z/OS. Version 2 lays the groundwork for the next tier of mainframe computing, enabling you to pursue the innovation to drive highly scalable workloads, including private clouds, support for mobile and social applications, and more. Its unrivaled security infrastructure helps secure vast amounts of data. Its highly optimized availability can help you deliver data analytics solutions. And its continued improvements in management help automate the operations of IBM zEnterprise® systems.

With support for IBM zEnterprise EC12 (zEC12, Enterprise Class) and zEnterprise BC12 (zBC12, Business Class) systems, z/OS 2.1 offers unmatched availability, scalability, and security to meet the business challenges of cloud services and data analytics and the security demands of mobile and social network applications. Through its unique design and qualities of service, z/OS provides the foundation that you need to support these demanding workloads alongside your traditional mission-critical applications.

Authors

This book was produced by a team of specialists from around the world working at the International Technical Support Organization (ITSO), Poughkeepsie Center.

Karan Singh is a Project Leader at the International Technical Support Organization, Poughkeepsie Center. He writes extensively and teaches IBM classes worldwide on all areas of z/OS.

Paul-Robert Hering is an IT Specialist at the ITS Technical Support Center, Mainz, Germany. He provides support to clients with z/OS and z/OS UNIX-related questions and problems. He has participated in several ITSO residencies since 1988, writing about UNIX-related topics. Before supporting IBM OS/390® and z/OS, Robert worked for many years with the IBM VM operating system and its variations (VM/370, VM/HPO, VM/XA, and VM/ESA).

George Kozakos is a Senior IT Specialist with IBM, in Australia. He has more than 25 years of experience in IBM Multiple Virtual Storage (IBM MVS™) system programming. George's areas of expertise include Server Time Protocol and IBM Geographically Dispersed Parallel Sysplex™ (IBM GDPS®). He holds degrees in Computing Science and Pure Mathematics.

Lutz Kuehner is a Senior Software Engineer working for Credit Suisse AG in Switzerland. He previously worked for 10 years as Client Technical Specialist with the System Technologie Group sales organization at IBM in Germany. Lutz has 27 years of experience in the field of mainframe technology, with expertise in z/OS. He has written extensively about z/OS-related topics and contributed the ITSO team by teaching classes and writing IBM Redbooks publications.

Jaqueline Mourão is a Systems Programmer at Banco do Brasil, a government bank in Brazil. She has 12 years of mainframe experience and holds a degree in Information Systems. Her areas of expertise include z/OS installation and maintenance, IBM Parallel Sysplex and security.

© Copyright IBM Corp. 2014. All rights reserved.

Diana Nakajima works in IBM Sales and Distribution in Brazil, where she is a Technical Sales Specialist, working with clients that have large IBM zSeries systems.

Nelson Oliveira is a Senior IT Specialist with IBM in Brazil. He has 22 years of experience in the field of mainframe technology. His areas of expertise include z/OS, JES2, Parallel Sysplex, high availability, and GDPS. Nelson supports the Banco do Brasil in his areas of expertise.

Gil Peleg is a systems programmer and manager of ServFrame, a mainframe consulting and training company in Israel and an IBM Business Partner. He has 15 years of experience in mainframe system programming and in teaching mainframe-related courses. His areas of expertise include Parallel Sysplex, GDPS, IBM Workload Manager (WLM), and JES3. Gil holds a degree in Computer Science.

Alvaro Salla worked for IBM for more than 30 years, specializing in large mainframe systems. He has co-authored many IBM Redbooks publications and spent many years teaching about large systems from IBM S/360 to IBM S/390®. He has a degree in Chemical Engineering from the University of Sao Paulo, Brazil.

Yu Uchida is an IT specialist with IBM Japan. He has five years of experience in IBM System z® and z/OS technical support, mainly in security areas, such as Security Server RACF and Cryptographic Services. He is also a member of team that is conducting the largest System z modernization project in IBM Japan.

Antonio Vilarins is a Senior Systems Programmer at Banco do Brasil, a government bank in Brazil. He has six years of experience in the field of mainframe technology. His areas of expertise include Parallel Sysplex, z/OS performance, and GDPS.

Thanks to Robert Haimowitz in the IBM International Technical Support Organization, Poughkeepsie Center, for his contributions to this project:

Now you can become a published author, too

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time. Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us.

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form:

  ibm.com/redbooks

- Send your comments in email:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks

- Follow us on Twitter:

  http://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS feeds:

  http://www.redbooks.ibm.com/rss.html

1

Installation and migration

Note: This IBM Redbooks publication summarizes many of the changes, features, and functions provided with IBM z/OS Version 2 Release 1 (2.1). It does not include every change nor cover all features and functions, but it covers most topics that are of interest to a systems programmer.

This chapter gives an overview of IBM z/OS Version 2 Release 1 (2.1) installation considerations, including changes in installation when migrating from Version 1 Release 13 (1.13) to Version 2.1. It describes tasks to prepare for the installation of z/OS 2.1, such as ensuring that the driving system and target system requirements are met and coexistence requirements are satisfied. It includes the following sections:

- 1.1, “Planning for z/OS 2.1 installation” on page 2
- 1.2, “Installation of z/OS 2.1” on page 6
- 1.3, “Driving system requirements for z/OS 2.1” on page 7
- 1.4, “Elements changed in z/OS 2.1” on page 13
- 1.5, “Preventive Service Planning buckets” on page 14
- 1.6, “Remove deleted data sets, paths, and references” on page 15
- 1.7, “Documentation and references” on page 16

Changes to installations before z/OS 1.13 are covered in *z/OS Planning for Installation*, GA32-0890:

http://www.ibm.com/support/docview.wss?uid=pub1ga32089000

© Copyright IBM Corp. 2014. All rights reserved.

1.1 Planning for z/OS 2.1 installation

This section gives an overview of tasks that to prepare for the installation of z/OS 2.1. Be sure to complete these tasks:

- Read documentation that describes the installation and migration considerations
- Review the Preventive Service Planning (PSP) buckets for installation-related information.
- Ensure that driving system and target system requirements are met.
- Install the required coexistence and fallback service on z/OS V.12 and z/OS 1.13 systems that will coexist with z/OS 2.1 system.

For an overview of z/OS migration and to plan for the installation of z/OS 2.1, read the following documentation:

- *z/OS Migration,* GA32-0889

  http://www.ibm.com/support/docview.wss?uid=pub1ga32088900
- *z/OS Planning for Installation*, GA32-0890

  http://www.ibm.com/support/docview.wss?uid=pub1ga32089000

This section provides an overview of considerations when migrating from z/OS 1.13 to z/OS 2.1. It identifies the coexistence, migration, and fallback considerations. The z/OS 2.1 is supported for coexistence, migration, and fallback with z/OS Versions 1.12 and 1.13. Ensure that the required coexistence program temporary fixes (PTFs) are installed on earlier z/OS releases that will coexist with z/OS 2.1.

When you are migrating your OS to z/OS 2.1, there are general actions that must be performed and actions for some of the elements in z/OS 2.1. For a description of all of the migration actions for z/OS 2.1, see *z/OS Migration,* GA32-0889.

1.1.1 Hardware requirements

The minimal hardware requirements for z/OS and the additional hardware that is needed by specific z/OS elements and features are documented in *z/OS Planning for Installation,* GA32-0890.

Review the hardware PSP buckets that are applicable for your installation. The *z/OS Migration,* GA32-0889 publication documents the hardware PSP buckets. See “Hardware PSP upgrade identifiers” on page 15.

Server requirements

z/OS 2.1 runs on the following IBM servers:

- IBM System z12 Enterprise Class (zEC12)
- IBM System z12 Business Class (zBC12)
- IBM System z196 zEnterprise 196 (z196)
- IBM System z114 zEnterprise 114 (z114)
- IBM System z10® Enterprise Class (z10 EC)
- IBM System z10 Business Class (z10 BC)
- IBM System z9® Enterprise Class (z9 EC), formerly IBM System z9 109 (z9-109)
- IBM System z9 Business Class (z9 BC)

1.1.2 Identifying coupling facility requirements

There are hardware and software requirements for coupling facility levels (CFLEVELs). See the “Coupling facility code level” table on this Parallel Sysplex page:

http://www.ibm.com/systems/z/pso/cftable.html

1.1.3 Software requirements

Before installing z/OS 2.1, review the Preventive Service Planning (PSP) bucket for z/OS 2.1, which is upgraded in z/OS 2.1. It includes subset ZOSGEN, which contains general information, and it includes subsets for the z/OS elements. The z/OS 2.1 PSP bucket might contain installation-related information, such as changes to the *z/OS Program Directory,* GI10-0670 since the document was written.

Choosing IBM products that you want to run with z/OS

Some z/OS elements and features require other IBM products that are not part of z/OS. The functional and minimal requirements for z/OS are described in *z/OS Planning for Installation,* GA32-0890.

For a list of products that are available to order with z/OS, you can use IBMLink, use the self-service tools on the Shopz page, or contact an IBM representative:

https://www.ibm.com/ibmlink/
http://www.ibm.com/software/shopzseries

If you are migrating to z/OS 2.1 from a prior release, you can find out which products have levels by using ShopzSeries or by using the SMP/E base element’s Planning and Migration Assistant. Both tools use data that is found on your system plus the latest IBM software product catalog.

Choosing vendors’ products that you want to run with z/OS

For more information, check this list of independent software vendors (ISVs) that support z/OS:

http://www.ibm.com/systems/z/solutions/isv/

Required service

Install the required service on the target system. Use the SMP/E Report MISSINGFIX, along with the enhanced HOLDDATA and the fix category for required service for z/OS 2.1 target system to identify required PTFs that must be installed.

Required CSI update

In the GLOBAl Zone add the **WORK=2M** parameter for the **IEBCOPY** utility. Without adding this value, you are likely to receive a failure during the SMP/E RECEIVE process for some of the z/OS 2.1 FMIDs.

1.1.4 Coexistence service requirements for z/OS 2.1

The required coexistence service must be installed on z/OS V1.12 and z/OS 1.13 systems that will coexist with z/OS 2.1. This is to enable the earlier z/OS releases to tolerate the changes in z/OS 2.1.

Use SMP/E V3.6 to identify required coexistence PTFs that must be installed on z/OS 1.12 and z/OS 1.13 systems in preparation for migration to z/OS 2.1.

SMP/E V3R5 and later provides an automated method to identify required coexistence service for a z/OS system by using the **REPORT MISSINGFIX** command.

Use the SMP/E **REPORT MISSINGFIX** command with the FIXCAT type HOLDDATA, as follows:

- Acquire and RECEIVE the latest HOLDDATA onto your z/OS 1.12 and z/OS 1.13 systems. Use your normal service acquisition portals, or download the HOLDDATA directly from the website:

  `http://service.software.ibm.com/holdata/390holddata.html`

- Run the SMP/E **REPORT MISSINGFIX** command on your z/OS 1.12 and z/OS 1.13 systems, and specify this Fix Category (FIXCAT) value:

  `IBM.Coexistence.z/OS.V2R1`

  The report identifies any missing coexistence and fallback program temporary fixes (PTFs) for that system. For complete information about the **REPORT MISSINGFIX** command, see *SMP/E Commands*, SA23-2275. Periodically, get updated HOLDDATA and rerun the **REPORT MISSINGFIX** command to find out whether there are any coexistence and fallback PTFs.

Coexistence considerations

Coexistence occurs when two or more systems at different software levels share resources. The resources can be shared at the same time by different systems in a multi-system configuration, or they can be shared over a period of time by the same system in a single-system configuration. Examples of coexistence are two different JES releases sharing a spool, two different service levels of DFSMSdfp sharing catalogs, multiple levels of SMP/E processing SYSMODs packaged to use the latest enhancements, or an earlier level of the system using the updated system control files of a newer level (even if a function is used in the newer level).

Note: The term *coexistence* does not refer to z/OS on a single system along with IBM z/VSE®, VSE/ESA, or IBM z/VM® in an LPAR or as a VM guest.

z/OS systems can coexist with specific prior releases. This is important because it gives you flexibility to migrate systems in a multi-system configuration by using rolling IPLs rather than requiring a systems-wide IPL. To make it possible for earlier-level systems to coexist with z/OS, you install coexistence service (PTFs) on the earlier-level systems.

Attention: Complete the migration of all earlier-level coexisting systems as soon as you can.

Keep in mind that the objective of coexistence PTFs is to allow existing functions to continue to be used on the earlier level systems when run in a mixed environment that includes later-level systems. Coexistence PTFs do not enable functions in later releases to work on earlier-level systems.

Attention: Using z/OS 2.1 with earlier releases of JES2 and JES3 is no longer supported.

Rolling IPLs

Rolling z/OS across a multisystem configuration is referred to as *rolling IPLs*. A rolling IPL is the IPL of one system at a time in a multisystem configuration. You can stage the IPLs over a few hours or a few weeks. Using rolling IPLs enables you to migrate each z/OS system to a later release, one at a time, with continuous availability of the application. For example, data sharing applications offer continuous availability in a Parallel Sysplex configuration by treating each z/OS system as a resource for processing the workload.

The use of rolling IPLs allows z/OS systems that are running these applications to be IPLed one at a time to migrate to a release of z/OS, while the applications continue to be processed by the other z/OS systems that support the workload.

By using logical partition (LPAR) technology, you can use rolling IPLs to upgrade your systems without losing either availability or capacity.

You can use rolling IPLs when *both* of the following conditions are true:

- The release to which you are migrating is supported for coexistence, fallback, and migration with the releases that are running on the other systems.
- The appropriate coexistence PTFs are installed on the other systems in the multisystem configuration.

Even when you are using applications that do not support data sharing, rolling IPLs often make it easier to schedule z/OS software upgrades. It can be difficult to schedule a time when all applications running on all systems in a multisystem configuration can be taken down to allow for a complex-wide or IBM Parallel Sysplex-wide IPL. The use of rolling IPLs enables continuous availability from a user application point of view and eliminates the work that is associated with migrating all z/OS systems in a multisystem configuration at the same time.

Fallback considerations

Fallback (backout) is a return to the prior level of a system. Fallback can be appropriate if you migrate to z/OS 2.1 and, during testing, encounter problems that can be resolved by backing out the release. By applying fallback PTFs to the prior release system before you migrate, the prior release system can tolerate changes that were made by the system during testing.

Fallback is relevant in all types of configurations, that is, single-system or multi-system, with or without resource sharing. As an example of fallback, consider a single system that shares data or data structures, such as user catalogs, as you shift the system image from production (on the old release) to test (on the release) and back again (to the old release). The later-level test release might make changes that are incompatible with the earlier-level production release. Fallback PTFs on the earlier-level release can allow it to tolerate changes that are made by the later-level release.

As a reminder: Always plan to have a backout path when installing software by identifying and installing any service that is required to support backout.

Fallback is at a system level, rather than an element or feature level, except for JES2 and JES3. That is, except for JES2 and JES3, you cannot back out an element or feature; you can back out only the entire product. JES2 and JES3 fallback can be done separately if the JES level is supported by the z/OS release and any necessary fallback PTFs are installed.

Remember: Using z/OS 2.1 with earlier releases of JES2 and JES3 is not supported.

Fallback and coexistence are alike in that the PTFs that ensure coexistence are the same ones that ensure fallback.

Note: Keep in mind that functions can require that all systems be at z/OS 2.1 level before the functions can be used. Therefore, be careful not to use functions until you are fairly confident that you will not need to back out your z/OS 2.1 systems, because fallback maintenance is not available in these cases. Consult the appropriate element or feature documentation to determine the requirements for using a particular function.

1.1.5 Installation considerations

Consider the conditions cited for the following elements:

- BCP:
  - The z/OS 2.1 product number has changed from 5694-A01 to 5650-ZOS, and it can be shared with z/OS 1.13. You must update the IFAPRDxx member of PARMLIB.
  - Health Checker now starts automatically.
- JES/SDSF:
  - Starting with 2.1, the staged migration path with old levels of JES2, JES3, and SDSF is no longer supported.
  - Only the latest levels of JES2, JES3, and SDSF are delivered in ServerPac and SystemPac.
- SMP/E:
  - The Global Zone entry in the CSI adds the parameter **`WORK=2M`** for IEBCOPY. Without adding this value, it is likely to receive a failure during the SMP/E RECEIVE for some of the z/OS 2.1 FMIDs.
- z/OS Font Collection (element):
  - It is necessary to move from previous fonts to the font collection.
  - Previous font programs can no longer be ordered.

1.2 Installation of z/OS 2.1

The program number to order z/OS 2.1 is 5650-ZOS. It became generally available on September 30, 2013 via ServerPac, CBPDO, and SystemPac.

When ordering, be sure that you order all of the optional features that you were licensed for in previous releases of z/OS.

There are several IBM packages available for installing z/OS. Some are entitled as part of your z/OS license, at no additional charge, and others are available for an additional fee.

You might find that sharing system libraries or cloning an already installed z/OS system is faster and easier than installing z/OS with an IBM installation package such as ServerPac. You do not have to wait for the order to be processed nor to receive the package, and you do not have to use the CustomPac Installation dialog twice. Sharing the system libraries (logical SYSRES volume) might also save DASD and support costs because you need to install service (or other products) only one time.

However, before sharing or cloning z/OS, you must have a license for each z/OS operating system that you run. If you do not have the appropriate license or licenses, you must contact IBM. Any sharing or cloning of z/OS without the appropriate licenses is not an authorized use.

1.2.1 Installing with ServerPac

ServerPac is an entitled software delivery package of products and services for which IBM has performed the System Modification Program/Extended (SMP/E) installation steps and some of the post-SMP/E installation steps. To install the package on your system and complete the installation of the software it includes, you use the CustomPac Installation dialog. Both z/OS and the products that run on it are available through ServerPac.

CustomPac dialog

The CustomPac Installation dialog generates tailored installation jobs and saves detailed definitions of volume, catalog, and data set configurations, which can be tailored, saved, and merged to install subsequent ServerPacs. The CustomPac Installation dialog is the same dialog that is used for all the CustomPac offerings, including SystemPac (dump-by-data-set format), IBM ProductPac®, and RefreshPac. For more information about CustomPac fee offerings, see the IBM CustomPac services web page:

http://www.ibm.com/services/custompac/

If you plan to install a ServerPac, review the ServerPac PSP bucket.

Installing z/OS 2.1 by using ServerPac includes the following documentation with the ServerPac order:

- *ServerPac: Using the Installation dialog*, SA23-2278
- The custom-built installation guide, *ServerPac: Installing Your Order (no order number; custom-built to your order)*.

1.2.2 Installing with CBPDO

If you plan to install the CBPDO deliverable for z/OS 2.1, you must delete the element that is withdrawn as of z/OS 2.1 from the target system as described in the *z/OS V2R1 Program Directory,* GI11-9848.

The sample job, CLNOS390, was not updated to delete HSAP360. Therefore, a DUMMY FUNCTION delete is required to remove the Integrated Call Level Interface (ICLI) component of z/OS UNIX, (HSAP360). The CLNOS390 job deletes withdrawn elements. The *z/OS V2R1 Program Directory,* GI11-9848 provides instructions on running the sample job CLNOS390.

The obsolete libraries, paths, and DDDEFs for the deleted elements must be removed after the withdrawn elements have been deleted by running the CLNOS390 sample job. The *z/OS Migration,* GA32-0889 identifies the deleted libraries, paths, and DDDEFs.

If you plan to install z/OS 2.1 by using CBPDO, review the *z/OS Program Directory,* GI11-9848.

1.3 Driving system requirements for z/OS 2.1

The *driving* system is the system image (hardware and software) that you use to install the target system. The *target* system is the system software libraries and other data sets that you are installing. You log on to the driving system and run jobs there to create or update the target system. After the target system is built, it can be IPLed on the same hardware (same LPAR or same processor) or different hardware than that used for the driving system.

If your driving system will share resources with your target system after the target system has been IPLed, be sure to install applicable coexistence service on the driving system before you IPL the target system.

1.3.1 Customized Offerings Driver V3 (5751-COD)

The Customized Offerings Driver V3 (5751-COD) is an entitled driving system, you can use it under the following conditions:

- You do not have an existing system to use as a driving system.
- Your existing system does not meet driving system requirements and you do not want to upgrade it to meet those requirements. This driver is a subset of a z/OS 1.12 system.

The Customized Offerings Driver is in DFSMSdss dump/restore format and supports 3390 triple-density or higher DASD devices. The Customized Offerings Driver requires a locally attached non-SNA terminal and a system console from the IBM (or equivalent) family of supported terminal types: 317x, 327x, 319x, or 348x. An IBM (or equivalent) supported tape drive is also required to restore the driver.

The Customized Offerings Driver is intended to run in single-system image and monoplex modes only. Its use in multisystem configurations is not supported. The Customized Offerings Driver is intended to be used only to install levels of z/OS by using ServerPac or a Custom-built Product Delivery Option (CBPDO) and to install service on the software until a copy (clone) of the system can be made. The use of the Customized Offerings Driver for other purposes is not supported.

Note: IBM does not support the use of the Customized Offerings Driver to run any production workload.

The Customized Offerings Driver includes a hierarchical file system and the necessary function to use Communications Server (IP Services), Security Server, and the system-managed storage (SMS) facility of DFSMSdfp, however, these elements are not customized. However, existing environments can be connected to, and used from, the Customized Offerings Driver system.

1.3.2 Identifying driving system requirements for z/OS 2.1 by using ServerPac

The driving system requires the following elements to install z/OS by using ServerPac or dump-by-data-set SystemPac:

Operating system Use either of the following:

z/OS 1.12 or later
The Customized Offerings Driver V3 (5751-COD)

TSO/E session A Time Sharing Option Extensions (TSO/E) session on the IPLed system must be established by using a locally attached or network-attached terminal.

Proper authority Use the RACFDRV installation job as a sample of the security system definitions that are required so that you can perform the installation tasks.

Proper security To install the UNIX files, the following are required:

The user ID that you use must either be a superuser (UID=0) or have Read access to the **BPX.SUPERUSER** resource in the IBM Resource Access Control Facility (RACF) FACILITY class.

The user ID that you use must have Read access to these FACILITY class resources: **BPX.FILEATTR.APF**, **BPX.FILEATTR.PROGCTL**, and **BPX.FILEATTR.SHARELIB** (or **BPX.FILEATTR.*** if you choose to use a generic name for these resources). The commands to define these FACILITY class resources are in SYS1.SAMPLIB member BPXISEC1. Group IDs uucpg and TTY and user ID uucp must be defined in your security database. These IDs must contain OMVS segments with a GID value for each group and a UID value for the user ID. (For ease of use and manageability, define the names in uppercase.)

OMVS A/S active

For ServerPac only (not SystemPac), an activated OMVS address space with z/OS UNIX kernel services operating in full function mode is required.

SMS active

The Storage Management Subsystem (SMS) must be active to allocate z/OS UNIX file systems (HFS or zFS) and PDSE data sets, whether they are SMS-managed or non-SMS-managed. Also, the use of z/OS UNIX file systems (HFS or zFS) is supported only when SMS is active in at least a null configuration, even when the data sets are not SMS-managed. Do either of the following tasks:

- To allocate non-SMS-managed z/OS UNIX file systems (HFS or zFS) and PDSE data sets, you must activate SMS on the driving system in at least a null configuration. You must also activate SMS on the target system.
- To allocate SMS-managed z/OS UNIX file systems (HFS or zFS) and PDSE data sets, you must activate SMS on the driving system in at least a minimal configuration. Then, you must define a storage group, create SMS-managed volumes, and write, translate, and activate a storage class automatic class selection (ACS) routine that allows the allocation of PDSE and z/OS UNIX file systems (HFS or zFS) with the names in the ALLOCDS job. You must also activate SMS on the target system.

LE

The CustomPac Installation dialog uses the IBM Language Environment® runtime library, SCEERUN. If SCEERUN is not in the link list on the driving system, you must edit the ServerPac installation jobs to add it to the JOBLIB or STEPLIB DD statements. Do not specify the following Language Environment runtime options as not over-rideable (NONOVR) in the CEEDOPT CSECT: ALL31, ANYHEAP, BELOWHEAP, DEPTHCONDLIMIT, ERRCOUNT, HEAP, HEAPCHK, HEAPPOOLS, INTERRUPT, LIBSTACK, PLITASKCOUNT, STACK, STORAGE, THREADHEAP, THREADSTACK.

CustomPac

If you are installing a ServerPac or dump-by-data set SystemPac for the first time, you need to install the CustomPac Installation dialog on your driving system. See *ServerPac: Using the Installation dialog,* SA23-2278 or *SystemPac: CustomPac dialog Reference* for instructions. For subsequent orders, you do not need to reinstall the dialog. IBM includes dialog updates with each order. Check the PSP bucket for possible updates to the CustomPac Installation dialog. For ServerPac, the upgrade is ZOS2.1 and the subset is SERVERPAC.

For SystemPac dump-by-data set orders, the upgrade is CUSTOMPAC and the subset is SYSPAC/DBD.

Service level

To install service on the target system that you are building, your driving system must meet the minimal driving system requirements for CBPDO Wave 1 and must have the current (latest) levels of the program management binder, SMP/E, and HLASM. The service jobs that are generated by the CustomPac Installation dialog use the target system's (and therefore current) level of the program management binder, SMP/E, and HLASM.

If you choose to use your own jobs, model them after the jobs provided by ServerPac or dump-by-data-set SystemPac by adding STEPLIB DD statements to access MIGLIB (for the program management binder and SMP/E) and SASMMOD1 (for HLASM). Be sure that the SASMMOD1 and SYS1.MIGLIB data sets are APF-authorized.

Another way to install service is from a copy of your target system.

SMP/E

For the SMP/E ++JAR support, if your ServerPac order contains any product that uses the ++JARUPD support that was introduced in SMP/E V3R2 (which is the SMP/E in z/OS V1R5), your driving system requires IBM SDK for z/OS, Java 2 Technology Edition, V1 (5655-I56) at SDK 1.4 or later.

zFS

The z/OS file system (zFS) must be configured properly. If you will use zFS for installation, you must be sure that zFS is installed and configured as described in *z/OS Distributed File Service zFS Administration,* SC23-6887.

Internet delivery

If you intend to receive your ServerPac order on the Internet, you need the following capabilities:

SMP/E V3R6, which is in Versions 1.12 and 1.13

ICSF configured and active so that it can calculate SHA-1 hash values to verify the integrity of data that is being transmitted. If ICSF is not configured and active, SMP/E calculates the SHA-1 hash values by using an SMP/E Java application class if IBM SDK for z/OS, Java 2 Technology Edition, V1 (5655-I56) or later is installed. We recommend the ICSF method because it is likely to perform better than the SMP/E method. (For how to configure and activate ICSF, see *z/OS Cryptographic Services ICSF System Programmer's Guide*, SA22-7520. For the required SMP/E setup, see *SMP/E User's Guide*, SA23-2277.)

A download file system. Your order is provided in a compressed format and is saved in a download file system. The size of this file system should be approximately twice the compressed size of your order to accommodate the order and workspace to process it.

Firewall configuration. If your enterprise requires specific commands to allow the download of your order by using FTP through a local firewall, you must identify these commands for later use in the CustomPac Installation dialog, which manages the download of your order.

Proper dialog level. If you are using a CustomPac Installation dialog with a Package Version that is less than 17.00.00, you must migrate the dialog to this level or later. You can determine whether you have the correct dialog level by looking for the text *that says* "*This dialog supports electronic delivery"* at the bottom of the CPPPPOLI panel. If

your dialog is not at the minimum level, follow the migration scenarios and steps that are described in *ServerPac: Using the Installation dialog*, SA23-2278.

If you choose to download your ServerPac order to a workstation and from there to z/OS, you need the following capabilities in addition to the requirements listed before:

Download Director: This is a Java applet that is used to transfer IBM software to your workstation.

The ServerPac order accessible to the host. To make the ServerPac order (files) accessible to z/OS, you can use either of the following methods:

- Configure the workstation as an FTP server. After you download the order to your workstation, the dialogs that are used to install ServerPac can point to a network location (in this case, your workstation) to access the order. Consult the documentation for your workstation operating system to determine whether this FTP capability is provided or you must install software. Commercial, shareware, and freeware applications are available to provide this support. However, IBM does not recommend nor endorse any specific application. This option requires the use of ICSF.
- Use network drives that are mounted to z/OS. The mounting can be accomplished by using the NFS base element, server message block (SMB) support that is provided by the Distributed File Service base element, or the Distributed FileManager component of the DFSMSdfp base element. The package is received from the file system that is defined as your SMPNTS.

CD write capability. If you specified that 100% electronic delivery is required, there might be CD images with your order. The images are delivered in ISO9660 format and are packaged in compressed files (with a .zip extension). These files require your workstation to have CD write capability and you might have to acquire software to support this capability.

DASD storage requirements for z/OS 2.1

If you are migrating from z/OS 1.13 to z/OS 2.1, you will have a different product set than your previous release, so you might see an increased need for DASD. How much more depends on what levels of products you are running.

Note: The DASD that is required for your z/OS system includes ALL elements, ALL features that support dynamic enablement, regardless of your order, and ALL unpriced features that you ordered. This storage is in addition to the storage that is required by other products that you have installed. All sizes include 15% free space to accommodate the installation of maintenance.

The total storage (3390 Cylinders) required for z/OS data sets is listed in the space tables in the *z/OS Program Directory, GI11-9848*.

Table 1-1 on page 12 lists the approximate total storage that is required for all the system data sets.

Table 1-1 System data sets - approximate total storage required

	z/OS 2.1	z/OS 1.13	Delta
Target Libraries	10842	5868	4977
Distribution Libraries	17657	8941	8716
Root	3309 (ZFS) 3293 (HFS)	3309(ZFS) 3293(HFS)	No delta No delta
Fonts (new)	2048 (ZFS) 2028 (HFS)	- -	Full delta Full delta

1.3.3 Identifying driving system requirements and installing a CBPDO

After you have your z/OS 2.1 system installed by using ServerPac, you need to install service on the z/OS 2.1 system. The driving system to install service on your z/OS 2.1 ServerPac system is the same as to install with the CBPDO method. For servicing your ServerPac system, your driving system must meet the requirements for Wave 1. When you use the CBPDO method of installing z/OS, you install in three stages called *waves*. (To be more manageable, Wave 1 is divided into several tasks called *ripples*.) This section describes the driving system requirements for each wave.

z/OS 2.1 CBPDO: Driving system Wave 0

In Wave 0, you install the program management binder (part of the BCP), the HLASM base element, and the SMP/E base element. These items must be installed on (available from) the driving system for subsequent wave installations. You can use either of the following as the driving system for installing Wave 0:

- z/OS 1.12 or later release.
- The Customized Offerings Driver V3 (5751-COD)

Wave 0 requires that the target system file system be mounted on the driving system because the code for the program management binder and SMP/E installs into that file system.

z/OS 2.1 CBPDO: Driving system Wave 1

In Wave 1, you install most of the elements and features of z/OS. These are the driving system requirements for installing Wave 1:

Operating system Use either of the following systems:

z/OS 1.12 or later, except that the program management binder, HLASM, and SMP/E must be at the current (latest) levels. To satisfy the program management binder, HLASM, and SMP/E requirements, you can use a STEPLIB DD statement to access the z/OS 2.1 binder, HLASM, and SMP/E in the Wave 0 target system's SYS1.MIGLIB and ASM.SASMMOD1 data sets. Ensure that the target system's SYS1.MIGLIB and ASM.SASMMOD1 data sets are APF-authorized on the driving system.

The Customized Offerings Driver V3 (5751-COD)

Proper security To install the UNIX files, the security definitions on page 9, under "Identifying driving system requirements for z/OS 2.1 by using ServerPac", are required.

SMP/E The SMP/E UTILITY must have an entry for the binder. You can specify any of these program names in the UTILITY entry: IEWBLINK,

HEWL, IEWL, LINKEDIT, or HEWLH096. (The linkage editor, which uses the names HEWLKED, HEWLF064, IEWLF440, IEWLF880, and IEWLF128, cannot be used.)

LE — You must add SCEERUN (the runtime library that is provided by Language Environment) to your program search order because many elements and features require it. If you want, add SCEERUN to your LNKLST concatenation. If you do not add SCEERUN to your LNKLST, you access SCEERUN by using STEPLIB DD statements in the individual element and feature procedures that require them. The BCP is one element that requires access to SCEERUN; its program management binder is the source of this requirement since OS/390 Version 2 Release 10 (2.10). This means that you must make SCEERUN available (by LNKLST concatenation or STEPLIB DD statement, for instance) to any JCL and procedures (such as SMP/E procedures) that start the binder. This ensures that processing, such as conversion of long names to short names, is successful.

OMVS — Before you install the elements and features in Wave 1, you must activate the OMVS address space in full function mode on the driving system. To activate OMVS, complete the required customization (for example, SMS and RACF setup) as described in *z/OS UNIX System Services Planning*, GA32-0884.

File system — The target system's file system must be mounted.

z/OS 2.1 CBPDO: Driving system Wave 2

In Wave 2, you install the z/OS 2.1 level of JES2 or JES3 and the z/OS 2.1 System Display and Search Facility (SDSF). Wave 2 is optional and can be combined with Wave 1. The driving system requirement for Wave 2 is the same as for Wave 1.

CBPDO installation considerations

- The font collection requires samples to be run to create a file system, directory entries, DDDEFS, and allocations.
- The Unicode size has increased significantly (see the sample jobs).
- The BCP sample jobs BPXISDDD and BPXMKDIR were updated for two DDDEFs and paths that are required for installation of HOT7790.

1.4 Elements changed in z/OS 2.1

The following elements and features are changed in z/OS 2.1:

- Base Control Program (BCP)
  - Base, BCP Unicode, Program Management Binder, BCP Capacity
- Provisioning
- Common Information Model (CIM)
- Communications Server
  - IP Services, SNA Services
- Cryptographic Services
  - ICSF, PKI, System SSL
- DFSMSdfp, DSFSMdss, DFSMShsm, DFSMSrmm, DFSMStvs
- DFSORT
- Distributed File Service (DFS) and z/OS file system (zFS)
- Hardware configuration definition (HCD)

- Hardware Configuration Manager (HCM)
- IBM Tivoli® Directory Server
- IBM InfoPrint Server
- Integrated Security Services, Network Authentication Service
- Interactive System Productivity Facility (ISPF)
- JES2
- JES3
- Language Environment
- Library Server
- Network File System (NFS)
- IBM Resource Access Control Facility (RMF™)
- Runtime Library Extensions
- System Display and Search Facility (SDSF)
- Security Server RACF
- Time Sharing Option Extensions (TSO/E)
- XL C/C++
- z/OS Security Level 3
  - System SSL Security Level 3
  - Network Authentication Srv Level 3
  - IBM Tivoli Directory Server for z/OS Security Level 3
- z/OS Font Collection (new)
- z/OS UNIX Application Services

The following element is withdrawn in z/OS 2.1:

- ICLI component of z/OS UNIX (HSAP360)

The following elements will be withdrawn after z/OS 2.1:

- Book Manager Build
- IBM HTTP Server

There are two elements in this version:

- z/OS Font Collection:
  - HFNT110 (z/OS Font Collection) Base
  - HFNT11J (z/OS Font Collection: Chinese, Japanese, Korean)
- BCP FMID HOT7790

1.4.1 Cryptographic Services

The Cryptographic Services ISCF is changed in z/OS 2.1, and the ICSF FMID is HCR77A0.

The FMID HCR77A0 is also available as Cryptographic Support for z/OS 1.12 and 1.13, which you can download from the “z/OS downloads” web page:

`http://www.ibm.com/systems/z/os/zos/downloads/`

1.5 Preventive Service Planning buckets

z/OS and most products that run on it provide files that contain information that becomes available after the product information is published. These files are kept in the IBM RETAIN® system and are also available by using IBMLink ServiceLink. These files are called *Preventive Service Planning (PSP) buckets*, or just *PSPs*. They are available to IBM clients on the Preventive Service Planning buckets website:

http://www14.software.ibm.com/webapp/set2/psp/srchBroker

The PSP buckets are identified by an upgrade identifier, and specific parts of them are called subsets. Each upgrade contains information about a product. Subsets contain information about specific parts of a product. For example, the z/OS PSP bucket has subsets for the z/OS elements, such as BCP, JES2, ServerPac, and others.

Software upgrades

For software upgrades for ServerPac and CBPDO installations, see the *z/OS Program Directory,* GI11-9848, which is available to download from the z/OS V2R1 Elements and Features Library:

http://www.ibm.com/systems/z/os/zos/library/bkserv/v2r1pdf/

For software upgrades for SystemPac installations, the upgrade is CUSTOMPAC and the subsets are SYSPAC/FVD (for full volume dump format) and SYSPAC/DBD (for dump-by-data set format).

At the beginning of each PSP bucket is a change index. For each subset, the change index identifies the date of the latest entries in each section. You can quickly determine whether there are entries you must read by checking the change index.

The upgrade for the z/OS 2.1 PSP bucket is ZOSV2R1. z/OS uses descriptive element names rather than FMIDs for the subsets. This reduces the number of PSP buckets that must be reviewed, because most elements consist of multiple FMIDs. There are subsets in the ZOSV2R1 upgrade for general topics (ZOSGEN) and for the ServerPac deliverable (SERVERPAC) that should be reviewed also. All PSP upgrades and subset IDs for z/OS 2.1 are listed in the *z/OS Program Directory*, GI11-9848. However, for the non-exclusive elements, the program upgrade and subsets are used.

Hardware PSP upgrade identifiers

Hardware Preventive Service Planning (PSP) bucket upgrade IDs are in the form xxxxDEVICE. They contain the latest software dependencies for the hardware and the PTFs and APARs that are required for specific processor models. These are the PSP hardware upgrade identifiers:

- 2827DEVICE for the zEC12 server
- 2817DEVICE for the z196 server
- 2097DEVICE for the z10 EC server
- 2098DEVICE for the z10 BC server
- 2094DEVICE for the z9 EC server
- 2096DEVICE for the z9 BC server

1.6 Remove deleted data sets, paths, and references

Use the table in *z/OS Migration,* GA32-0889 as a guide and remove deleted data sets, paths and references to the deleted data sets and paths and determine whether references to the data sets must be deleted from the following places:

- Parmlib
- Proclib
- Logon procedures
- Catalogs
- Security definitions, including program control definitions
- DFSMS ACS routines

- /etc/profile
- SMP/E DDDEF entry
- Any backup and recovery procedures.

If the catalog is shared with earlier z/OS releases, do not remove references to the data sets that are obsolete in z/OS 2.1 if the data sets are still referenced in the earlier releases.

The following elements are withdrawn as of z/OS 2.1; therefore, the target and distribution data sets and the file system paths that were associated with the withdrawn elements are deleted:

- DCE
- Integrated Security Services DCE
- z/OS UNIX Connection Manager
- Process Manager

In addition, the DFS element has several data sets that are deleted as of z/OS VR13.

The Table 1-2 lists the data sets and file system paths that are deleted as of z/OS 2.1:

Table 1-2 Data sets and file system paths that are not included in z/OS 2.1

DDEF	Data set	Element
AFOMDATA	SYS1.AFOMDATA	ICLI component of z/OS UNIX (HSAP360) removed from z/OS
SFOMDATA	SYS1.SFOMDATA	ICLI component of z/OS UNIX (HSAP360) removed from z/OS
SFOMEHFS	/usr/lpp/icli/IBM	ICLI component of z/OS UNIX (HSAP360) removed from z/OS
SFOMESBN	/usr/lpp/icli/sbin/IBM	ICLI component of z/OS UNIX (HSAP360) removed from z/OS

1.7 Documentation and references

For more information, see the follow IBM publications and the “Related publications” on page 537:

- *z/OS Introduction and Release guide,* GA32-0887

  http://www.ibm.com/support/docview.wss?uid=pub1ga32088700
- *z/OS MVS Initialization and Tuning Reference,* SA23-1380

  https://ibm.biz/BdRWn3
- *z/OS UNIX System Services Planning,* GA32-0884

  https://ibm.biz/BdFEbq
- *z/OS V2R1 Planning for Installation,* GA32-0890

  http://www.ibm.com/support/docview.wss?uid=pub1ga32089000

2

IBM Health Checker for z/OS

This chapter describes the following changes that were introduced in the Health Checker in IBM z/OS Version 2 Release 1 (2.1). It includes the following sections:

- 2.1, “The auto-start feature” on page 18
- 2.2, “HZSPRINT PARM longer than 100 characters” on page 21
- 2.3, “Refreshed METAL C sample health checks” on page 22
- 2.4, “System name in message buffer” on page 22
- 2.5, “The and updated checks of IBM Health Checker” on page 23
- 2.6, “Documentation and reference” on page 29

© Copyright IBM Corp. 2014. All rights reserved.

2.1 The auto-start feature

In previous z/OS releases the Health Checker needed to be configured before use, missing many opportunities to prevent system problems.

In z/OS 2.1, some of the manual configuration steps that are required have been automated, and a large subset of the Health Checker functions are active by default.

This change helps ensure that you receive early warnings, so you can prevent serious system problems.

2.1.1 Invocation and use

The Health Checker address space is now started automatically at IPL time, via the HZSPROC procedure. HZSPROC is now in the PROCLIB, rather than in the SAMPLIB.

Note: If you have not used Health Checker before, be prepared to initially handle several check “exceptions” that are classified as HZS0001I (low severity), HZS0002E (medium), and HZS0003E (high).

- Refer to the individual check message and the details in the check's message buffer on how to “fix” those exceptions.
- Customize the check behavior, via HZSPRMxx PARMLIB members, to avoid exception messages in the future.

A system parameter, HZS, can be specified in IEASYSxx as an alternative. It is the preferred way to identify the HZSPRMxx PARMLIB members to be used when Health Checker starts.

This release introduces values for the HZSPRM parameter of the HZSPROC procedure. You can use those values in this command to manually start or restart the Health Checker:

```
S HZSPROC,HZSPRM=SYSPARM|PREV|NONE
```

Where:

SYSPARM	Use suffixes specified via system parameter HZS (can be empty).
PREV	Use suffixes that were in use by the previous instance of Health Checker. Also behaves like SYSPARM for the first start of Health Checker during an IPL. This is the recommended value.
NONE	To explicitly run without any HZSPRMxx members

You can also specify suffixes in the form, `HZSPRM={x1|(x2,...,xn)}`, as before. This action ignores the system parameter, **HZS**.

Table 2 in the *IBM Health Checker for z/OS: User’s Guide*, SC23-6843 provides details about the interaction of HZSPRMxx settings that are specified in HZSPROC and the HZS keyword that is specified in IEASYSxx. This table indicates which, if any, HXSPRMxx PARMLIB members are used.

2.1.2 Installation

Installation is necessary only if you have not used Health Checker in previous releases. If you have been using it in a previous release, follow the procedures in the migration section.

First, make sure that the HZSPROC procedure is available in your PROCLIB.

There are several additional setup steps that are described in the *Health Checker User's Guide*, but most of them are optional. Many of the steps that follow are highly recommended:

1. Update the HZSPROC procedure to specify a persistent data set, as Example 2-1shows, so that health checks preserve data across IPLs.

   Example 2-1 HZSPROC procedure

   ```
   //HZSPROC PROC HZSPRM='PREV'
   //HZSSTEP EXEC PGM=HZSINIT,REGION=OK,TIME=NOLIMIT,
   // PARM='SET PARMLIB=&HZSPRM'
   //HZSPDATA DD DSN=SYS1.&SYSNAME..HZSPDATA,DISP=OLD
   // PEND
   // EXEC HZSPROC
   ```

2. Use the member HZSALLCP in SYS1.SAMPLIB for an example of the data set requirements of the data set that is specified by the ddname HZSPDATA. The HZSPDATA must be allocated with a required format as shown in Example 2-2.

   Example 2-2 HZSPDATA required format

   ```
   //HZSPDATA DD  DSN=SYS1.system_name.HZSPDATA,DISP=(NEW,CATLG),
   //         SPACE=(4096,(100,400)),UNIT=SYSDA,
   //         DCB=(DSORG=PS,RECFM=FB,LRECL=4096)
   ```

   If you do not specify an HZSPDATA data set, the system issues this message: **HZS0013A SPECIFY THE NAME OF AN EMPTY HZSPDATA data set**.

3. Define in RACF or in your security product a user ID for the Health Checker and associate this user with the HZSPROC address space. The Health checker user must have an OMVS segment with UID(0) or BPX.SUPERUSER permissions, as shown in Figure 2-1. Make sure that the Health Checker user ID has access to your persistent data set and optionally to other resources. See the *Health Checker For z/OS: User's Guide*,23-6843 for more information.

```
ADDUSER hcid OMVS(UID(yy) HOME('/') PROGRAM('/bin/sh'))
NOPASSWORD
ADDGROUP OMVSGRP OMVS(GID(xx))
CONNECT hcid GROUP(OMVSGRP)
PERMIT BPX.SUPERUSER CLASS(FACILITY) ID(hcid) ACCESS(READ)

RDEFINE STARTED HZSPROC.* STDATA(USER(hcid) GROUP(OMVSGRP))
```

Figure 2-1 Define the Health Checker user

4. Put any optional Health Checker customization such as health check POLICYs, LOGSTREAM connects into the HZSPRMxx PARMLIB members.
5. Update IEASYSxx with the suffix of the HZSPRM member that is used in your installation, and specify the HZS parameter, for example **HZS=(aa,...,zz)**. Or reply to the IEA101A message at IPL. The procedure HZSPROC specifies HZSPRM=PREV by default.

Managing exceptions

The first time that the Health Checker is started, many exceptions might be generated. You can take the following actions to manage these exceptions:

1. Health checks can be adjusted to your preferred practices easily by using check parameters. This initial adjustment is preserved by HZSPRMxx PARMLIB members.
2. If you must, mark “incurable” checks INACTIVE by using HZSPRMxx, but let the rest continue to monitor the health of your system.
3. Compare the ADD POLICY UPDATE statement as discussed in the Health Checker User's Guide.
4. Fix the rest to prevent future system problems.
5. If high-severity check exception messages are flooding your console, consider these options:
   - Consider the CONTROL command, for example: **K S,DEL=R**
   - Lower the visibility and put this into your HZSPRMxx, temporarily. See the Example 2-3.

Example 2-3 HZSPRMxx

```
ADDREPLACE POLICY(HCONLY)
UPDATE CHECK(*,*) WTOTYPE(HARDCOPY)
REASON=('STOP RED MESSAGES')
DATE=(20130408)
ACTIVATE POLICY(HCONLY)
```

If you must disable the auto-start of Health Checker, alter the HZSPROC as follows. This prevents the automatic start of Health Checker but allows a manual start later.

```
//         EXEC HZSPROC=*NONE
```

2.1.3 Migration and coexistence considerations

If you used Health Checker in previous releases, follow these recommended actions:

1. Remove any manual Health Checker start commands. Typically found in COMMNDxx as **START HZSPROC**

   If they are not removed, a second start attempt will be rejected and one of the following warning message will be issued:
   - when the IPL-time instance is already up and running

     HZSO101I “...HEALTH CHECKER... IS ALREADY ACTIVE”
   - When the IPL-time instance is still initializing

     HZSO116I – “...HEALTH CHECKER... START PENDING’
2. If you specified the HZSPRM procedure parameter for the actual start command, for example **START HZSPROC,HZSPRM='01**', take these actions:
   - Move the HZSPRM value to the system parameter HZS and specify it in IEASYSxx.
   - Update your procedure to specify HZSPRM=PREV or HZSPRM=SYSPARM so the system will use the HZS system parameter.
   - You could specify the literal HZSPRM value in your procedure, but **HZSPRM=PREV** in the procedure is suggested.

3. If you renamed the procedure that is used to start Health Checker, specify the parameter **HZSPROC=***myhcproc* in IEASYSxx. Just renaming it back to HZSPROC might work, but remember that there likely is a user ID associated with the Health Checker address space through the procedure name. You must update that association also, for example by issuing **RDEFINE STARTED HZSPROC.* STDATA(USER(hcid) GROUP(hcidgrp))**
4. To keep your previous Health Checker procedure, do not copy the IBM HZSPROC from the IBM PROCLIB during system upgrade, because it might overwrite your existing HZSPROC or it might “hide” your renamed procedure (if you copy it to a PROCLIB that is listed earlier in the PROCLIB concatenation), because the system looks for the HZSPROC procedure by default.
5. Remember to keep the IBM PROCLIB towards the end of your PROCLIB concatenation, otherwise your HZSPROC might be hidden by the IBM HZSPROC.
6. Consider updating your existing procedure to take advantage of the special values for the procedure parameter HZSPRM. We suggest that you use **HZSPRM=PREV** in combination with UZS system value, rather than the previously used literal list of HZSPRMxx suffixes. The HZS system value does not have a default. The old HZSPROC coded **HZSPRM=“00”** as default.

2.2 HZSPRINT PARM longer than 100 characters

HZSPRINT is the tool to write check message (buffer) content to a data set and filter the output by certain criteria. z/OS V1R10 introduced several filter parameters, in particular TIMERANGE, and the total theoretical parameter length grew to over 100 characters (approximately 180).

In z/OS 2.1, HZSPRINT supports parameter strings longer than 100 characters.

2.2.1 Invocation and use

In z/OS 2.1, HZSPRINT uses the JCL PARMDD support. Use of the PARMDD statement allows up to 256 characters of parameter data to be passed, trailing blanks per input line do not count. The HZSPRINT member in the SAMPLIB was updated with the parameter PARMDD. See Figure 2-2 for an example.

```
//HZSPRINT EXEC PGM=HZSPRNT,TIME=1440,REGION=0M,PARMDD=SYSIN
//SYSIN DD *,DLM='@@'
CHECK(*,*)
,EXCEPTIONS
@@
```

Figure 2-2 The parameter PARMDD in HZSPRINT member

No rollback is planned for this HZSPRINT enhancement due to the PARMDD dependency.

2.2.2 Migration and coexistence considerations

No rollback is planned for this HZSPRINT enhancement because of the PARMDD dependency.

You can maximize the 100 characters for PARM in previous z/OS releases by how you write your code:

- Use wildcards in the check name filter parameter, for example instead of **(IBMPFA,PFA_ENQUEUE_REQUEST_RATE)** use **(*,PFA_E*)**.
- Avoid having trailing blanks, because that wastes space. Use JCL SET, as shown in Example 2-4.

Example 2-4 How to use JCL SET

```
//HZSPRINT JOB
// SET PARM1='CHECK(IBMASM,ASM_LOCAL_SLOT_USAGE)'
// SET PARM2=',LOGSTREAM(HZS.HEALTH.CHECKER.LOG)'
// SET PARM3=',EXCEPTIONS'
//HZSPRINT EXEC PGM=HZSPRNT,TIME=1440,REGION=0M,
// PARM='&PARM1.&PARM2.&PARM3.'
//SYSOUT DD SYSOUT=A,DCB=(LRECL=256)
```

2.3 Refreshed METAL C sample health checks

Some samples of METAL C checks that were introduced in z/OS V1R12 have been updated.

2.3.1 Invocation and use

You can see the updated samples in **/usr/lpp/bcp/samples**. These are the updated samples: **hzscadd.c**, **hzscchkp.c**, **hzscchkr.c**, **hzscrchk.c**, and **hzssmake.mk**.

The members HZSHCONS and HZSHPQE, provided in the SYS1.SIEAHDR.H, were also updated.

2.4 System name in message buffer

A health check's message buffer now contains the name of the system the check ran on. This makes it easier to associate the check output with a particular system.

2.4.1 Invocation and use

In the CK panel in the SDSF, if you select a check when the SDSF multi-system support is enabled, you can see the system where that check ran. See Example 2-5.

Example 2-5 A check with system name

```
CHECK(IBMCATALOG,CATALOG_RNLS)
SYSPLEX: PLEX1 SYSTEM: SY39
START TIME: 02/19/2013 12:16:40.224036
CHECK DATE: 20120827 CHECK SEVERITY: LOW

* Low Severity Exception *

IGGHC111E CHECK(IBMCATALOG,CATALOG_RNLS) found that the Catalog/DADSM
resources do not conform to IBM recommendations...
```

Note: That the SDSF display contains the system name also, but only on the main panel and much farther to the right in the standard setup.

2.5 The and updated checks of IBM Health Checker

More IBM Health Checker were introduced In z/OS 2.1, and others were changed.

Table 2-1 is a summary of the and the updated checks. The updated ones are marked with an asterisk (*).

Table 2-1 and updated checks in z/OS 2.1

Subsystem or component	or updated check
DFSMS OPEN/CLOSE/EOV	(IBMOCE,OCE_XTIOT_CHECK)
Access Control	(IBMRACF,RACF_AIM_STAGE) (IBMRACF,RACF_UNIX_ID) (IBMRACF,ZOSMIGV2R1_DEFAULT_UNIX_ID) (IBMRACF,RACF_SENSITIVE_RESOURCES)* (IBMRACF,RACF_CERTIFICATE_EXPIRATION)
Supervisor	(IBMSUP,SUP_SYSTEM_SYMBOL_TABLE_SIZE)
System trace	(IBMSYSTRACE,SYSTRACE_BRANCH) (IBMSYSTRACE,SYSTRACE_MODE)
Serviceability	(IBMSLIP,SLIP_PER)
VLF	(IBMVLF,VLF_MAXVIRT)
IOS	(IBMIOS,IOS_IORATE_MONITOR) (IBMIOS,IOS_FABRIC_MONITOR)
Communications Server	(IBMCS,ZOSMIGV2R1_CS_GATEWAY) (IBMCS,ZOSMIGV1R11_CS_DNSBIND9)*
Catalog	(IBMCATALOG,CATALOG_RNLS)
USS (z/OS UNIX)	(IBMUSS,USS_KERNEL_PVTSTG_THRESHOLD) (IBMUSS,USS_KERNEL_STACKS_THRESHOLD)
Storage Management	(IBMASM,ASM_PLPA_COMMON_SIZE)* (IBMASM,ASM_PLPA_COMMON_USAGE)* (IBMASM,ASM_LOCAL_SLOT_USAGE)* (IBMVSM,VSM_CSA_THRESHOLD)* (IBMVSM,VSM_SQA_THRESHOLD)*

2.5.1 or updated checks by component or subsystem

Details of the checks are described *by component* or *subsystem* in following subsections.

DFSM, owner (IBMOCE)

- The **OCE_XTIOT_CHECK** checks whether XTIOTs (TIOT, for Task I/O table, 16 MB or more) are enabled for non-VSAM data sets. Enabling XTIOTs for non-VSAM data sets

decreases the chances of running out of virtual storage when allocating and concurrently opening many sequential and partitioned data sets.

You can enable XTIOTs for non-VSAM data by specifying **NON_VSAM_XTIOT=YES** in the DEVSUPxx PARMLIB member.

`INTERVAL(ONETIME) - SEVERITY(LOW)`

Access control, owner (IBMRACF)

- The **RACF_AIM_STAGE** check examines the RACF database Application Identity Mapping (AIM) to see whether it is at AIM stage 3, which is recommended. AIM stage 3 allows RACF to more efficiently handle authentication and authorization requests from applications such as z/OS UNIX. Stage 3 is required to use some RACF function, for example to automatically assign unique UNIX UIDs and GIDs at the time they are needed.

  You can convert your RACF database to AIM stage 3 using the `IRRIRA00` conversion utility.

  `INTERVAL(ONETIME) - SEVERITY(MEDIUM)`

- The **RACF_UNIX_ID** check detects whether RACF is enabled to automatically assigning unique UNIX identities when users without OMVS segments access the system to use UNIX services. It also compares RACF profiles BPX.UNIQUE.USER and BPX.DEFAULT.USER of the FACILITY class. Using BPX.DEFAULT.USER without BPX.UNIQUE.USER is not recommended.

  z/OS V1.13 is the last release that supports default UNIX identities that are implemented by using the BPX.DEFAULT.USER profile.

  `INTERVAL(ONETIME) - SEVERITY(MEDIUM)`

- The **ZOSMIGV2R1_DEFAULT_UNIX_ID** determines whether a client is relying on RACF to assign default z/OS UNIX identities for users without OMVS segments who are accessing UNIX services. Assign a unique UNIX UID to each user and a unique GID to each group that needs access to z/OS UNIX functions and resources.

  Starting with z/OS 2.1, support for the default UNIX identity, which was implemented by using the BPX.DEFAULT.USER profile in the FACILITY class, is no longer available. A migration action might be required if you are using it. The need for a migration action is based on whether the BPX.UNIQUE.USER and BPX.DEFAULT.USER profiles are defined in the FACILITY class.

  `SEVERITY(MEDIUM)`

- The updated **RACF_SENSITIVE_RESOURCES** check examines the security characteristics of several system-critical data sets and general resources other than data sets. For example, for system-critical data sets, it checks whether the data sets exist on the expected volume and whether a resource has baseline protection.

  The following resources, in the indicated classes, are added to the existing RACF_SENSITIVE_RESOURCE check:

  - BPX.DEBUG/FACILITY
  - BPX.WLMSERVER/FACILITY
  - SUPERUSER.PROCESS.GETPSENT/UNIXPRIV
  - SUPERUSER.PROCESS.KILL/UNIXPRIV
  - SUPERUSER.PROCESS.PTRACE/UNIXPRIV
  - MVS.SLIP/OPERCMDS
  - IEAABD.DMPAUTH/FACILITY
  - IEAABD.DMPAKEY/FACILITY

- CSVAPF.data_set_name/FACILITY (not including CSVAPF.MVS.SETPROG.FORMAT.DYNAMIC/FACILITY)
- CSVDYLPA.ADD.module_name/FACILITY
- CSVDYLPA.DELETE.module_name/FACILITY
- CSVDYNEX.exit_name.DEFINE/FACILITY
- CSVDYNEX.exit_name.modname/FACILITY
- CSVDYNL.linklistname.DEFINE/FACILITY
- CSVDYNL.linklistname.ADD/FACILITY
- CSVDYNL.linklistname.DELETE/FACILITY
- CSVDYNL.linklistname.UNDEFINE/FACILITY

The **RACF_CERTIFICATE_EXPIRATION** check extracts certificates from the RACF database and examines the ending date of the certificate. It then lists the certificate in the check output if the ending date is equal to or less than the warning date.

If such a certificate is either TRUST or HIGHTRUST, it is marked as an exception.

The warning date is the current date that is adjusted by the “warning period” that the installation specified as a parameter for the check: **PARM('DAYS(nnn)')**, where “nnn” is 0 - 366, with a default of 60 days.

```
INTERVAL(ONETIME) - SEVERITY(MEDIUM)
```

Supervisor, owner (IBMSUP)

- The **SUP_SYSTEM_SYMBOL_TABLE_SIZE** checks whether the used size of the static system symbol table exceeds a specified threshold. The table capacity is > 800, approximately 32k.

  You can use the Check parameter **LIMIT(n|p%)**, with a default of 85%, to adjust the threshold by up to 32512 bytes or by a percentage.

  ```
  INTERVAL(ONETIME) - SEVERITY(LOW)
  ```

System trace owner (IBMSYSTRACE)

- The **SYSTRACE_BRANCH** checks whether system trace is using branch tracing and has been active longer than a specified amount of time. For example, using **TRACE BR=ON** for certain branch instructions.

  Leaving branch tracing **ON** can affect system performance. This is recommended only for short periods of time to solve a specific problem and is not recommended as a default for system tracing.

  Use the default of 7 days for the **TIME(DAYS,dd)** check parameter to adjust the time threshold.

  ```
  INTERVAL(4:00) - SEVERITY(LOW)
  ```

- The **SYSTRACE_MODE** checks whether system trace is using mode tracing (**TRACE MODE=ON**, for **AMODE 64 ON/OFF**) and has been active for longer than a specified amount of time.

  Leaving mode tracing **ON** can affect system performance. It is recommended only for short periods of time to solve a specific problem and is not recommended as a default for system tracing.

  The check parameter **TIME(DAYS,dd)**, with a default of 7 days, allows you to adjust the time threshold.

  ```
  INTERVAL(4:00) - SEVERITY(LOW)
  ```

Serviceability, owner (IBMSLIP)

- The **SLIP_PER** check checks whether a PER (program event record) trap has been active for longer than a specified amount of time. An active but not needed SLIP PER trap can cause degraded system performance.

  The check parameter **TIME({DAYS|HOURS},nn)**, with a default of 30 days, allows you to adjust the time threshold.

  ```
  INTERVAL(4:00) - SEVERITY(LOW)
  ```

Virtual lookaside facility, owner (IBMVLF)

- The **VLF_MAXVIRT** check examines whether VLF is trimming recently added objects while making room for objects. If so, the MAXVIRT setting, for at least one VLF class, might be too small for VLF to provide a good performance benefit.

  The **MAXVIRT** is used to define the data space size for each VLF class in the COFVLFxx PARMLIB member.

  Before this check, it was necessary to have the SMF 41(subtype 3) formatted to analyze the data spaces use.

  Use the **ALERTAGE(class_name1,alert_age1,class_name2,alert_age2,...)** check parameter to define an alert age for objects that are being cached in VLF. When an object younger than the alert value is trimmed, an exception is raised. The class name allows wildcards, for example (*,60). Therefore, the default of ALERTAGE (*,60) can be used to set an alert age of 60 seconds. This wildcard and value would then apply to all classes that do not have an ALERTAGE defined using COFVLFxx nor by the check parameter. The ALERTAGE values that are defined in the check override the ones that are specified in the COFVLFxx.

  ```
  INTERVAL(1:00) - SEVERITY(LOW)
  ```

Input/Output Supervisor, - owner (IBMIOS)

- The **IOS_FABRIC_MONITOR** check examines the health of the switch fabric and reports unusual conditions within the fabric.

  Devices can be connected to their processors via a switch fabric, which consists of one or more IBM FICON® directors in the path from the processor (channels) to the device (control unit). The directors can be a long distance from each other and connected by one or more inter-switch links (ISLs). When a performance problem occurs, it is often difficult to determine the cause. This check automates that task.

  Check parameters **LOG(NO|YES)** and **SHOW(LATEST|ALL)** control whether a diagnostic log is generated and whether all or only the one most recent exception queued for each route is reported.

  ```
  INTERVAL(0:10) - SEVERITY(MEDIUM)
  ```

- The **IOS_IORATE_MONITOR** detects if any control units in the system are reporting inconsistent I/O rates for their attached channel paths. It is available only for IBM zEnterprise EC 12 (zEC12) or later systems.

  Typically, I/Os are distributed equally across all paths for a control unit. Inconsistent I/O rates are the result of corrective action when the system directs I/Os away from a path, trying to correct performance problems.

  An exception notice is issued if the total I/O rate across all channel paths of a control unit exceeds the **THRESHOLD** check parameter value (default: 100 I/Os per second), and The ratio of the I/O rates for two channel paths for a control unit is higher than allowed by the **RATIO** check parameter value (default: 2).

Additional check parameters, **XTYPE(devtype)** and **XCU(cu1,...,cuN)**, can be used to exclude certain control unit device types **(DASD|TAPE)** or control units (identified by number) from checking and reporting.

INTERVAL(0:05) - SEVERITY(MEDIUM)

Communications Server, owner (IBMCS)

- The **ZOSMIGV2R1_CS_GATEWAY** checks whether the **GATEWAY** statement is in use in a TCP/IP profile on the system. Support for the **GATEWAY** statement will be removed in a future release of IBM z/OS Communications Server.

  We suggest that IBM clients who use the **GATEWAY** statement migrate to the **BEGINROUTES/ENDROUTES** configuration block.

  INTERVAL(24:00) - SEVERITY(LOW)

- The previous **ZOSMIGV1R11_CS_DNSBIND9** check has been removed in z/OS 2.1. It used to check whether a BIND9 DNS server is in use on a system. IBM has indicated in statements of direction that support for BIND9 DNS server functions on z/OS will not be available in IBM z/OS Communications Server releases after z/OS V1.13.

Catalog, owner (IBMCATALOG)

- The **CATALOG_RNLS** check makes specific recommendations about what to specify in global resource serialization resource name lists **GRS RNLs** to prevent catalog related deadlocks when using shared volumes and catalogs.

  The check signals an exception when **GRS RNLs** do not match IBM recommendations for reserves for **SYSIGGV2**, **SYSZVVDS** and **SYSVTOC**. Convert them all to **SYSTEMS ENQUEUEs** unless the user shares DASD outside of the Parallel Sysplex.

  INTERVAL(336:00) - SEVERITY(LOW)

z/OS UNIX System Service, owner (IBMUSS)

- The **USS_KERNEL_PVTSTG_THRESHOLD** check monitors availability of private storage in the kernel. Some z/OS UNIX syscalls fail if insufficient storage is available. That can result in an outage.

  Check parameter **PVTSTG(nn%)** controls when the check signals an exception about private storage being used up to a certain percentage. The default is 85%.

  INTERVAL(0:02) - SEVERITY(LOW-DYNAMIC)

- The **USS_KERNEL_STACKS_THRESHOLD** check monitors the kernel supply of stack (autodata) cell pool cells. Some z/OS UNIX syscalls fail if insufficient storage is available, which can result in an outage.

  Check parameter **STACKS(nn%)** controls when the check signals an exception about private storage being used up to a certain percentage. The default is 85%.

  INTERVAL(0:02) - SEVERITY(LOW-DYNAMIC)

Storage Management, owner (IBMASM)

- With the following updated checks, you now have the option to set them to support dynamic severity rather than fixed default severity for check exception notices:
  - ASM_PLPA_COMMON_SIZE
  - ASM_PLPA_COMMON_USAGE
  - ASM_LOCAL_SLOT_USAGE
  - VSM_CSA_THRESHOLD
  - VSM_SQA_THRESHOLD

For example, the VSM_SQA_THRESHOLD check has a default severity of MEDIUM for when its SQA or ESQA allocation thresholds are reached. Rather than check **SQA(nn%)** parameters, **ESQA(nn%)** you can now define multiple thresholds, as Example 2-6 shows:

Example 2-6 Defining multiple thresholds

```
SQA_HIGH(85%),SQA_MED(80%),SQA_LOW(70%)
ESQA_HIGH(90%),ESQA_MED(80%),ESQA_LOW(70%)
```

2.5.2 Use of the system health checks

Most checks are delivered ACTIVE and do not require manual interaction to run. To see the checks, select the SDSF **CK** option.

The migration checks are delivered INACTIVE. To display the status of a check, use the **DISPLAY** command as shown in Example 2-7.

Example 2-7 Display CHECKs status

```
-MODIFY HZSPROC,DISPLAY,CHECK(*,ZOSMIG*)
 HZS0200I 11.43.37 CHECK SUMMARY     653
 CHECK OWNER      CHECK NAME                       STATE STATUS
 IBMZFS           ZOSMIGREC_ZFS_RM_MULTIFS          IE   INACTIVE
 IBMCS            ZOSMIGV2R1_CS_GATEWAY             IE   INACTIVE
 IBMUSS           ZOSMIGREC_ROOT_FS_SIZE            IE   INACTIVE
 IBMTIMER         ZOSMIGREC_SUP_TIMER_INUSE         IE   INACTIVE
   A - ACTIVE          I - INACTIVE
   E - ENABLED         D - DISABLED
   G - GLOBAL CHECK    + - CHECK ERROR MESSAGES ISSUED
```

You can activate them by using the command that is shown in Example 2-8.

Example 2-8 Activate CHECK command

```
-MODIFY HZSPROC,UPDATE,CHECK(*,ZOSMIG*),ACTIVE
 HZS0400I CHECK(*,ZOSMIG*):
 UPDATE PROCESSING HAS BEEN COMPLETED
 HZS0001I CHECK(IBMCS,ZOSMIGV2R1_CS_GATEWAY):
 ISTM014E GATEWAYS statements are in use on this system during this IPL.
 HZS1002E CHECK(IBMUSS,ZOSMIGREC_ROOT_FS_SIZE):
 AN ERROR OCCURRED, DIAG: 00000000_00000008
```

Issue the display command again and check STATE and STATUS columns. See Example 2-9.

Example 2-9 Display CHECKs status

```
-MODIFY HZSPROC,DISPLAY,CHECK(*,ZOSMIG*)
 HZS0200I 12.03.35 CHECK SUMMARY     821
 CHECK OWNER      CHECK NAME                       STATE STATUS
 IBMZFS           ZOSMIGREC_ZFS_RM_MULTIFS          AE   SUCCESSFUL
 IBMCS            ZOSMIGV2R1_CS_GATEWAY             AE   EXCEPTION-LOW
 IBMUSS           ZOSMIGREC_ROOT_FS_SIZE            AD   UNEXP ERROR
 IBMTIMER         ZOSMIGREC_SUP_TIMER_INUSE         AE   SUCCESSFUL
   A - ACTIVE          I - INACTIVE
   E - ENABLED         D - DISABLED
   G - GLOBAL CHECK    + - CHECK ERROR MESSAGES ISSUED
```

A small set of non-migration checks are INACTIVE by default also. Typically, those are INACTIVE because their check parameters need individual adjustments that differ from installation to installation. Find those INACTIVE checks and decide whether to change them and to what parameters to run within your installation.

2.5.3 Installation

These IBM health checks are automatically installed with their associated product or component, or the are included with the base operation system for the release. Some checks, such as migration checks, have APARs available for previous releases. See the next section for more information.

2.5.4 Migration and coexistence considerations

Some CHECKs are available for previous z/OS releases, see the following APARs:

- OA37164 - (IBMRACF,RACF_AIM_STAGE)
- OA37164 - (IBMRACF,RACF_UNIX_ID)
- OA37164 - (IBMRACF,ZOSMIGV2R1_DEFAULT_UNIX_ID)
- OA40548, OA40037 – (IBMIOS,IOS_FABRIC_MONITOR) and (IBMIOS,ISO_IORATE_MONITOR)

Some checks were updated with DATEs. You must reevaluate any POLICY statements, in HZSPRMxx PARMLIB members, to make sure that those policies continue to get applied. Look for dates for:

- (IBMRACF,RACF_SENSITIVE_RESOURCES)
- (IBMASM,ASM_PLPA_COMMON_SIZE)
- (IBMASM,ASM_PLPA_COMMON_USAGE)
- (IBMASM,ASM_LOCAL_SLOT_USAGE)
- (IBMVSM,VSM_CSA_THRESHOLD)
- (IBMVSM,VSM_SQA_THRESHOLD)

2.6 Documentation and reference

For more information, see these IBM publications:

- *IBM Health Checker for z/OS: User's Guide*, SC23-6843

  https://ibm.biz/BdFEba
- *Exploiting the IBM Health Checker for z/OS Infrastructure*, REDP-4590

  http://www.redbooks.ibm.com/abstracts/redp4590.html?Open

3

Virtual lookaside facility

This chapter describes the enhancements, functions, and features of the virtual lookaside facility (VLF) that are introduced with z/OS Version 2 Release 1 (2.1). It includes the following sections:

- 3.1, “MODIFY VLF command” on page 32
- 3.2, “VLF PARMLIB concatenation” on page 32
- 3.3, “The VLF Health Check” on page 33
- 3.4, “Documentation and reference” on page 34

The VLF is a set of IBM MVS services that provide a high-performance, alternative path of retrieving named objects from DASD for many users. VLF uses data spaces to hold data objects in virtual storage as an alternative to repeatedly retrieving the data from DASD. Therefore, it improves the response time.

Certain products or components, such as library lookaside (LLA), IBM Time Sharing Option Extensions (TSO/E), coordinating address space (CAS), and IBM Resource Access Control Facility (RACF) use VLF data spaces as an alternative way to access data.

A VLF *class* is a group of related objects that are made available to users of an application or component.

© Copyright IBM Corp. 2014. All rights reserved.

3.1 MODIFY VLF command

The **MODIFY VLF,REPLACE,NN=** command was introduced in z/OS 2.1 to address the needs of installations where the VLF configuration needs to be changed without having to restart VLF, because restarting causes the loss of the current VLF cache and its performance benefit.

By using the **MODIFY** command, the VLF configuration can be changed without restarting VLF, and the existing VLF cache remains.

3.1.1 Invocation and use

The **MODIFY VLF,REPLACE,NN=** command replaces the current VLF configuration with a one (see Example 3-1). The objects that are cached in VLF remain cached if that is still appropriate for a configuration.

Example 3-1 MODIFY VLF command

```
-MODIFY VLF,REPLACE,NN=01
 COF026I MODIFY VLF PROCESSING IS COMPLETE.
```

The **replace** command supports a concatenation of up to 16 COFVLFxx PARMLIB members.

3.2 VLF PARMLIB concatenation

z/OS 2.1 allows up to 16 COFVLFxx PARMLIB members to be concatenated to form one VLF configuration when VLF is started. A concatenation of VLF configuration options allows more complex VLF configuration options.

This is a true concatenation because class definitions can span from one member to the next.

3.2.1 Invocation and use

Up to 16 COFVLFxx PARMLIB members may be concatenated. VLF treats the entire concatenation of PARMLIB members as one configuration. Duplicate definitions are rejected with the appropriate existing error messages. When more than one COFVLFxx PARMLIB member is specified, the set of members must be enclosed in parenthesis and separated by commas, as Example 3-2 shows.

Example 3-2 Start a concatenated VLF

```
-S VLF,SUB=MSTR,NN=(00,01)
...
COF011I VLF INITIALIZATION IS IN PROGRESS.
...
COF105I COFVLF01, RECORD 28, CSVLLA IS A DUPLICATE CLASS DEFINITION.
COF105I COFVLF01, RECORD 33, IKJEXEC IS A DUPLICATE CLASS DEFINITION.
COF105I COFVLF01, RECORD 41, IGGCAS IS A DUPLICATE CLASS DEFINITION.
COF025I VLF INITIALIZATION IS COMPLETE.
```

Possible COFVLFxx PARMLIB changes

The following considerations are relevant to the use of COFVLFxx concatenation:

- Classes may be added or deleted. When a class is deleted, any programs that are currently using it receive existing failure return codes. The cache for the class is purged.
- Major names (**EMAJ** or **EDSN**) may be added to or deleted from an existing class. When a major name is deleted, any programs that are currently using it receive existing failure return codes. The objects in the cache that is associated with the deleted major names are purged.
- The **MAXVIRT** parameter can be specified, raised, or lowered for an existing class. A decrease in **MAXVIRT** might cause VLF trimming of the oldest objects to reduce the cache to the size.
- The **ALERTAGE** parameter can be specified, raised, or lowered for an existing class. An **ALERTAGE** check parameter that is specified for the VLF Health Check overrides this value, even if it is changed by using a **MODIFY** command.

3.2.2 Migration and coexistence considerations

There is no authorized program analysis report (APAR) for pre-z/OS 2.1 releases for a concatenation of COFVLFxx PARMLIB members.

3.3 The VLF Health Check

The z/OS 2.1 introduces a check called **VLF_MAXVIRT** for the Health Checker. It is owned by IBMVLF.

3.3.1 Invocation and use

This **VLF_MAXVIRT** check monitors VLF trimming activity to detect when objects are being trimmed that are younger than a specified age threshold. That can be an indication of thrashing, because **MAXVIRT** is too low. The **MAXVIRT** parameter specifies the data space size and **ALERTAGE** specifies an age threshold.

The **ALERTAGE** is an optional class parameter in COFVLFxx. It has a default value of 60 seconds. The **ALERTAGE** can also be specified as a check parameter for the **VLF_MAXVIRT** check. When it is specified as a check parameter, it overrides the value that is specified in COFVLFxx.

Example 3-3 shows a successful **VLF_MAXVIRT**check.

Example 3-3 The VLF_MAXVIRT check

```
CHECK(IBMVLF,VLF_MAXVIRT)
SYSPLEX:    PLEX75    SYSTEM: SC74
START TIME: 05/01/2013 12:46:54.456162
CHECK DATE: 20110802  CHECK SEVERITY: LOW

COFVLH01I For all classes, VLF is trimming objects within the goals set
for this check.

END TIME: 05/01/2013 12:46:54.456316  STATUS: SUCCESSFUL
```

If the **ALERTAGE** parameter for class CSVLLA is set (to 60 seconds, for example), but VLF finds that it has trimmed an object for CSVLLA that was cached for only 45 seconds, the Health Check raises an alert to recommend that the **MAXVIRT** parameter be increased to provide more cache space for the CSVLLA class.

For more **VLF_MAXVIRT** check details, see *Chapter 2, "IBM Health Checker for z/OS" on page 17*.

3.4 Documentation and reference

For more information, see the following IBM publications in the z/OS V2R1 Elements and Features Library (PDF files):

http://www.ibm.com/systems/z/os/zos/library/bkserv/v2r1pdf/

- *z/OS V2R1.0 MVS Authorized Assembler Services Guide*, SA23-1371
- *z/OS V2R1.0 MVS Initialization and Tuning Reference*, SA23-1379
- *z/OS V2R1.0 MVS System Commands*, SA38-0666

4

System Modification Program/Extended

This chapter describes the changes to System Modification Program/Extended (SMP/E) that were introduced in IBM z/OS Version 2 Release 1.

SMP/E is a tool that manages the installation of software products on a z/OS system and tracks modifications of those products.

This chapter includes the following topics:

- 4.1, “SMP/E ISPF dialog split screen display” on page 36
- 4.2, “Documentation and reference” on page 38

© Copyright IBM Corp. 2014. All rights reserved.

4.1 SMP/E ISPF dialog split screen display

SMP/E ISPF now allows you to open multiple dialogs windows at one time, using split-screens displays.

Before this was possible, it was difficult to correlate information from multiple entries or more than one zone. Also, you had to exit the dialog and choose another entry or zone, so you lost your place while working on a task.

Figure 4-1 shows the SMP/E ISPF error message from the previous z/OS version.

```
SC58 GIMQUPO                QUERY SELECTION MENU
===>

    1  CSI QUERY           - Display SMPCSI entries
    2  CROSS-ZONE QUERY    - Display status of an entry in
                             all zones
    3  SOURCEID QUERY      - Display SOURCEIDs for specified zone

    D  DESCRIBE            - Overview of using QUERY

    T  TUTORIAL            - Information on using QUERY

To return to the SMP/E primary option menu, enter  END .

.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
SC58 GIMISCVE            SMP/E INITIALIZATION ERROR
===>

*****************************************************************************
*                                                                           *
* The following error was detected while attempting to                      *
* initialize the SMP/E dialogs. The initialization of                       *
* these dialogs has not completed.                                          *
*                                                                           *
*                                                                           *
*                                                                           *
*                                                                           *
* The message has been logged in your ISPF log.  You may                    *
* wish to print your log and give it to the system                          *
* programmer responsible for maintaining this system.                       *
*                                                                           *
* To return to ISPF, enter END .                                            *
*                                                                           *
*****************************************************************************

     +-------------------------------------------------------------+
     | GIMIS093 Split-screen mode is not allowed for the SMP/E dialogs |
 F1= +-------------------------------------------------------------+ NGE
 F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

Figure 4-1 Previous z/OS SMP/E split-screen mode error notice

With the SMP/E ISPF dialog split-screen support, you can view and more easily correlate information from multiple entries and multiple zones.

4.1.1 Installation

The support is supplied as an SPE for SMP/E V3.6 (z/OS V1.13 and z/OS V2.1) with APAR: IO18093 (FMID: HMP1J00).

4.1.2 Invocation and use

The SMP/E ISPF dialog split-screen support allows multiple dialogs to be open at one time, with the following conditions:

- CSI data sets may be opened for read in multiple dialogs at once.
- A CSI data set may be open for update in only one dialog. Accessing the SMP/E Administration dialog and the Order Management dialog puts the SMP/E session in Update mode.

See Figure 4-2 for an example of multiple SMP/E dialog windows open in an ISPF session.

```
#@$A GIMQIT28              CSI QUERY - HFS      ENTRY         Row 1 to 2 of 2
===>                                                        SCROLL ===> PAGE

 To return to previous panel, enter END .

 Primary Command: FIND

 Entry Type:  HFS                                    Zone Name: DB8QT
 Entry Name:  DDASQLJS                               Zone Type: TARGET

 FMID    : HDDA211       DISTLIB : ADDABIN  LASTUPD: HDDA211  TYPE=ADD
 RMID    : HDDA211       SYSLIB  : SDDABIN  TEXT
 SHSCRIPT:

         ---------------------------------------------------------------

LINK     '../sqlj'
PARM     PATHMODE(0,7,5,5)
******************************* Bottom of data ********************************
.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
#@$A GIMQITPT            CSI QUERY - PATH DDDEF ENTRY         Row 1 to 1 of 1
===>                                                        SCROLL ===> PAGE

 To return to the previous panel, enter END .

Primary Command: FIND

 Entry Type:  DDDEF                                Zone Name: DB8QT
 Entry Name:  SDDABIN                              Zone Type: TARGET

      ---------------------------------------------------------------------
PATH: '/service_dsn910/usr/lpp/jcct4v3/bin/IBM/'
******************************* Bottom of data ********************************
```

Figure 4-2 Two SMP/E dialogs open in a split screen

Remember that SMPCSI data sets may be open for update in only one dialog. If an update function is attempted from multiple SMP/E dialogs, a message similar to the one shown in Figure 4-3 is displayed.

```
#@$A     TARGET/DLIB ZONE DEFINITION - ENTRY TYPE SELECTION
===>

Global CSI: DSN910.GLOBAL.CSI
Zone name:  DB8QT

Select one of the following:

   1  ZONE            - create or modify the zone definition
   2  DDDEF           - create or modify the DDDEF entries
   3  ZONEDESCRIPTION - create or modify the ZONEDESCRIPTION



.  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
#@$A GIMAD            ADMINISTRATION SELECTION MENU
===>

Select one of the following:

   1  DEFINITION - Define, update, delete, or display zone
                   definition entries and ZONEDESCRIPTION
                   subentries, and OPTIONS, UTILITY, DDDEF,
                   FMIDSET, and ZONESET entries.

   D  DESCRIBE   - An overview on the use of SMP/E
                   Administration dialogs

   T  TUTORIAL   - Detailed information on the use of SMP/E
                   Administration dialogs

   NOTE: For information about the use of PRIMARY and LINE
         commands, including the "Locate" and "Select"
         c +-----------------------------------------------------+
           | GIMXX055 SMPCSI data set DSN910.GLOBAL.CSI is busy |
5694-A01 5 +-----------------------------------------------------+
```

Figure 4-3 Error when trying to perform update action from multiple SMP/E dialogs

4.2 Documentation and reference

For more information, see the following publications and web pages:

- z/OS V2R1 Information Center, SMP/E Version 3 (documentation)
  https://ibm.biz/BdR7CU
- IBM Knowledge Center: Changes made in SMP/E Version 3, Release 4
  https://ibm.biz/BdR7CN

- IBM Knowledge Center: SMP/E Version 3, where you can download the following publications:

  https://ibm.biz/BdR7CW

  - *SMP/E User's Guide*, SA23-2277
  - *SMP/E Commands*, SA23-2275
  - *SMP/E Reference*, SA23-2276
  - *SMP/E Messages, Codes, and Diagnosis*, GA32-0883

5

Dynamic Channel Path Management

This chapter includes the following topics and describes the enhancements, functions, and features for Dynamic Channel Path Management that were introduced with IBM z/OS Version 2.1:

- 5.1, “DCM support for cascaded FICON switches” on page 42
- 5.2, “Documentation and reference” on page 46

© Copyright IBM Corp. 2014. All rights reserved.

5.1 DCM support for cascaded FICON switches

Dynamic Channel Path Management (DCM) was initially included in the first z/OS release. At that time, FICON native channels were not supported, and DCM supported only IBM Enterprise Systems Connection (IBM ESCON) and Fibre Channel connection (FICON) Bridge channels. DCM and FICON DCM allow z/OS to manage channel paths (FICON and ESCON) dynamically. You need to identify the channels and control units that need to be managed.

Defining an I/O configuration to maximize availability and performance is complex and cumbersome. Many clients tend to over-configure the I/O configuration to manage performance peaks.

The DCM that is provided allows z/OS 2.1 to dynamically manage FICON channel paths and control unit ports in response to changing workload demands. FICON DCM support is extended to allow cascaded or multiswitch connections for managed channels. This enhancement simplifies I/O configuration definition, improves I/O performance, permits a more efficient use of hardware resources, dynamically balances I/O resources based on workload demand, and can enhance availability by dynamically adding channel paths for certain error conditions.

This enhancement eliminates the requirement of only one switch between the processor and control unit. It uses 2-byte link addresses for cascaded connections and can have a mixture of managed cascaded channel paths and managed non-cascaded channel paths.

5.1.1 Invocation and use

To enable DCM, follow these steps:

1. Define managed control units in the input/output definition file (IODF) by specifying at least one static channel (two are better for availability) and specifying one or more asterisks to indicate that managed channels can be assigned. Static channels must be defined as shared or spanned and switch-attached. The control unit must be accessible to all systems in the logical partition (LPAR) cluster.
2. Define managed channels in the IODF that are attached to the switches for the managed control units. The I/O (LPAR) cluster name must be the name of the IBM Parallel Sysplex where the managed channels will be used. If your LPAR cluster name spans multiple logical channel subsystems (LCSSes), managed channels must be defined for each LCSS.
3. Define switch devices in the IODF and vary them online to z/OS. It is important to remember that a non-managed channel must be defined to access the control unit port (CUP). The CUP is used by DCM to retrieve switch topology information.
4. Enable at least one partition in the I/O or LPAR cluster to make dynamic I/O configuration changes. This is in the LPAR image profile on the HMC.

Note: The **SETIOS DCM=ON** command can be used after IPL. DCM can be enabled or disabled at any time by using the **SETIOS DCM** command or activating an IODF with or without managed resources defined. It is possible to display the status of DCM by using the **DISPLAY IOS,DCM** command.

Examples of HCD panels

The following sequence of figures shows HCD panels to illustrate the steps to enable DCM.

When you define the managed control unit, as shown in Figure 5-1, type an asterisk as a placeholder for the channel path and leave the link address field blank.

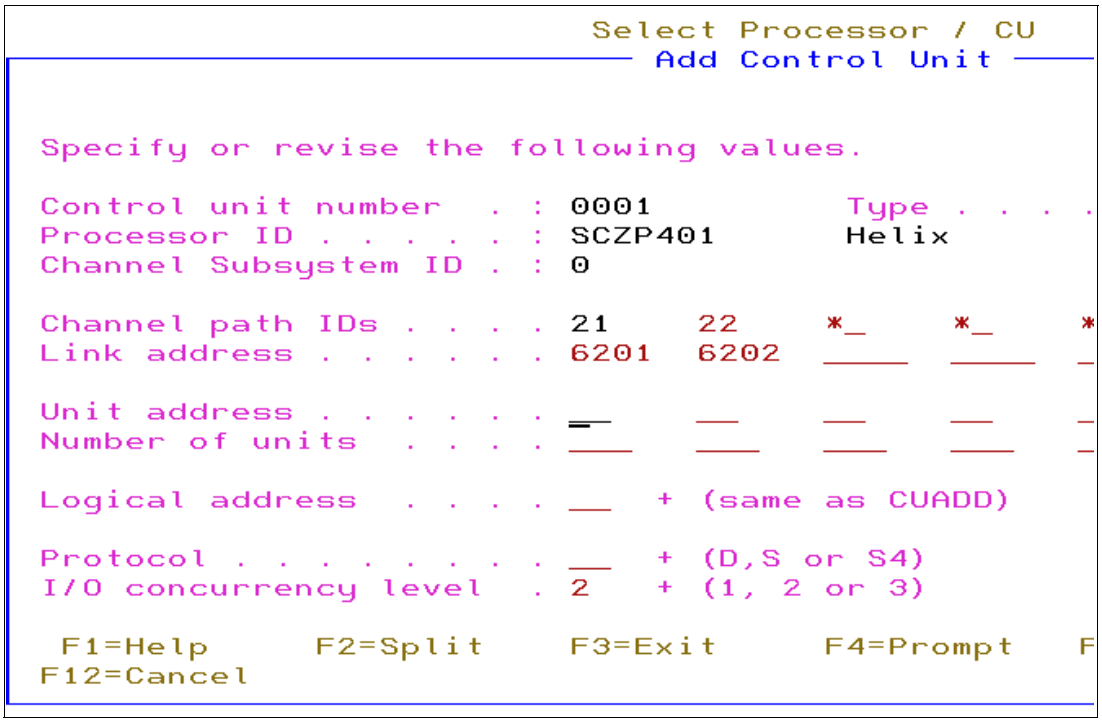


Figure 5-1 Add Control Unit panel, adding a managed control unit

In Figure 5-2 shows the control unit definition where you can see the asterisk in the channel path ID column.

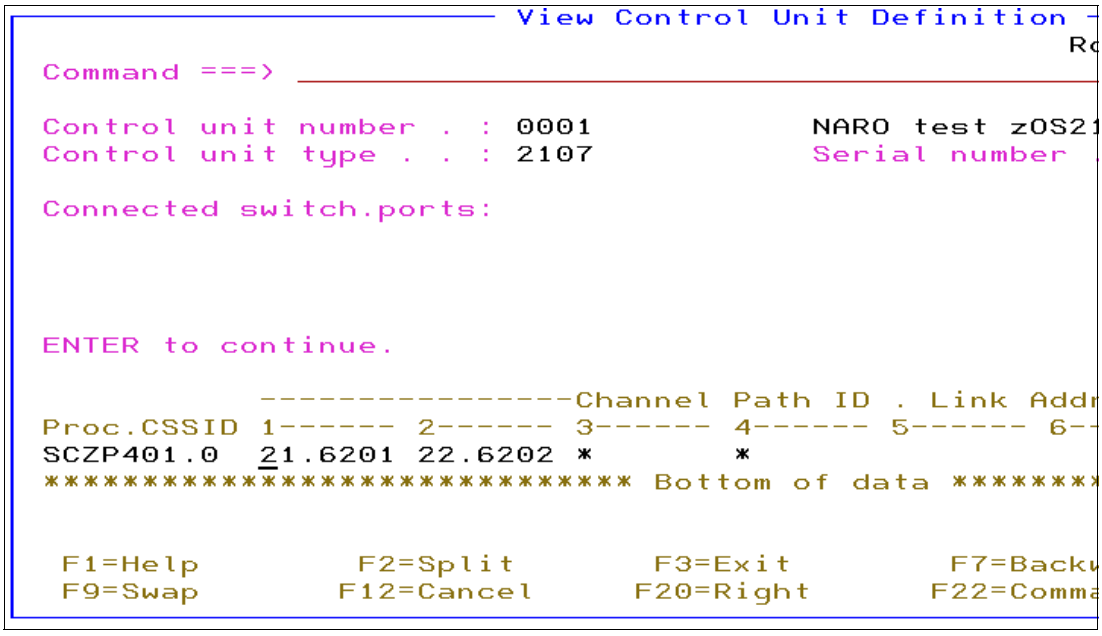


Figure 5-2 View Control Unit Definition panel

When adding a managed channel path, use the existing “Managed” field to designate whether the channel path will be managed by DCM or not. In an LPAR environment, such a channel path must be shared (SHR). And because the scope of management for DCM is at the I/O cluster level, you must define it as shown in Figure 5-3.

```
                              Add Channel Path


Specify or revise the following values.

Processor ID . . . . : SCZP401      Helix
Configuration mode . : LPAR
Channel Subsystem ID : 0

Channel path ID  . . . . 03    +             PCHID . . . ___
Number of CHPIDs . . . . 1
Channel path type  . . . FC_   +
Operation mode . . . . . SHR   +
Managed  . . . . . . . . YES  (Yes or No)    I/O Cluster SC75
Description  . . . . . . Managed chpid_____________________

Specify the following values only if connected to a switch:
Dynamic entry switch ID  62  + (00 - FF)
Entry switch ID  . . . . 62  +
Entry port . . . . . . . 11  +
```

Figure 5-3 Add Channel Path panel

In the channel path list, the existing “Mng” column shows which channel paths are managed (Yes) and which channels are not managed (No), as shown in Figure 5-4.

```
                                Channel Path List
Command ===> _____________________________________________

Select one or more channel paths, then press En

Processor ID . . . . : SCZP401       Helix
Configuration mode . : LPAR
Channel Subsystem ID : 0

          PCHID                  Dyn Entry +
/ CHPID AID/P Type+ Mode+ Sw+ Sw Port Con Mng De
_ 00    574   OSD   SPAN  __  __ __         No  E>
_ 01    54C   OSC   SPAN  __  __ __         No  E>
_ 02    _____ FC    SHR   62  62 11         Yes Ma
_ 06    50C   OSD   SPAN  __  __ __         No  E>
_ 07    570   OSD   SPAN  __  __ __         No  E>
_ 0C    5B4   OSD   SPAN  __  __ __         No  E>
_ 0D    5C8   OSC   SPAN  __  __ __         No  E>
_ 0E    544   OSD   SPAN  __  __ __         No  E>
_ 0F    578   OSD   SPAN  __  __ __         No  E>
_ 12    5C4   OSD   SPAN  __  __ __         No  E>
```

Figure 5-4 Channel Path List panel, checking whether the channels are managed or not

5.1.2 Installation

Complete the following steps to install FICON DCM:

1. Create a plan to explore DCM. It is best that you “start small” with the installation. To determine the number of managed channels and control unit interfaces, analyze the workload.
2. Ensure that at least one system in the LPAR cluster is enabled to make dynamic configuration changes.
3. Ensure that the CUP feature is installed on switches that will be connected to managed channels.
4. Change the I/O configuration (IODF):
   a. Define managed control units.
   b. Define managed channels.
   c. Define switch devices.
5. Activate the I/O configuration.
6. Ensure that switch devices are brought online to z/OS.

Evaluate DCM after installation

To evaluate the environment and identify problems that could occur after the DCM implementation, these two suggestions are helpful:

- Enable DCM Component Tracing for IOS (update IECIOSxx and CTIIOSxx).
- Use IBM Resource Measurement Facility™ (RMF) reports (channel, device, IOQ, and ESS) with intervals not greater than 15 minutes.

5.1.3 Interactions and dependencies

There is no software dependency consideration.

From a hardware perspective, consider the following points:

- Processor: All currently supported processors
- Channels: All currently supported FICON channels
- Coupling facility: Required if running multisystem
- Switches:
  - Must have control unit port (CUP) function
  - CUP must be installed on entry switch, exit switch, and intermediary switches for cascade connections
  - Must be defined in the IODF

5.1.4 Migration and coexistence considerations

For migration and coexistence, there are no compatibility program temporary fixes (PTFs) required, and the existing ESCON DCM function is not affected by the change that is introduced for FICON DCM.

z/OS V2.1 does not have to be installed on all systems in the LPAR cluster to use DCM for cascaded fabrics. However, only DCM at the z/OS 2.1 level can make changes in a cascaded configuration, where the control unit is attached only to the exit switch. DCM at earlier levels marks the control unit as ineligible for DCM because the control unit is not attached to an entry switch.

5.2 Documentation and reference

For more information, see the following publications and web pages:

- *z/OS MVS System Commands,* SA38-0666-01
  https://ibm.biz/BdRWnL
- *Dynamic Channel Path Management (DCM), z/OS HCD User's Guide,* SC33-7988-10
  https://ibm.biz/BdRWne
- *z/OS V2R1 Hardware Configuration Definition User's Guide*, SC34-2669
  https://ibm.biz/BdRWn8
- *z/OS HCD Reference Summary*, SX33-9032
  http://www.ibm.com/support/docview.wss?uid=pub1sx33903206
- *z/OS Intelligent Resource Director*, SG24-5952
  https://ibm.biz/BdRWnw
- *z/OS MVS System Messages, Vol 9 (IGF-IWM)*, SA38-0676
  https://ibm.biz/BdRWnQ
- *z/OS MVS Initialization and Tuning Reference*, SA23-1380
  https://ibm.biz/BdRWn3
- *z/OS Resource Measurement Facility User's Guide, Version 2 Release 1*, SC34-2664
  https://ibm.biz/BdRWnT

6

Hardware configuration and management console changes

This chapter describes the various enhancements, added functions, and features for hardware configuration definition (HCD) and hardware configuration management (HCM) that were introduced in IBM z/OS Version 2 Release 1 (V2.1).

- 6.1, “I/O discovery and autoconfiguration” on page 48
- 6.2, “Verify Configuration function with I/O autoconfiguration” on page 56
- 6.3, “HMC-wide activation of dynamic I/O changes” on page 60
- 6.4, “Unconditional generation of DR site OS configurations” on page 66
- 6.5, “HCD validation” on page 67
- 6.6, “HCD OS group change action for device groups” on page 67
- 6.7, “HCD batch utility” on page 68
- 6.8, “HCD support of PCIe functions” on page 69
- 6.9, “PCHID Summary Report” on page 76
- 6.10, “OFFLINE parameter in the MVS Device Detail Report” on page 77
- 6.11, “HCD CF Channel Path Connectivity List” on page 77
- 6.12, “HCD hardware support” on page 78
- 6.13, “HCM port number status line” on page 78
- 6.14, “Connecting CHPIDs to control units in HCM” on page 79
- 6.15, “HCM explanatory message on load production IODF” on page 79
- 6.16, “Documentation and reference” on page 80

© Copyright IBM Corp. 2014. All rights reserved.

6.1 I/O discovery and autoconfiguration

This section explains the enhancements to the I/O z/OS Discovery and Autoconfiguration (zDAC) function.

6.1.1 Discovery by serial number

With the ability to allow discovery by controller serial number, you can restrict discovery to a specific controller if it already has a control unit defined. HCD filters the discovered information and presents only the controllers that have matching serial numbers.

As Figure 6-1 shows, you can limit discovery to a controller with a specific serial number (generic specification is supported) and skip systems that are not able to perform discovery. Select “Scope of discovery” option **4, Controller with S/N**, to limit discovery.

Specifying a controller with a serial number indicates that discovery should return only the controllers with the fully or partially specified serial number. A serial number specification is required. It can be specified fully or only with a prefix (followed by an asterisk: *) or a suffix (preceded by an asterisk: *) of the serial number.

```
                      /OS V2.1 HCD
+------------------ Discovery and Autoconfiguration Options ------------------+
|                                                                             |
|                                                                             |
| Specify autoconfiguration options. Then, press Enter to start the           |
| discovery process.                                                          |
|                                                                             |
| Autoconfiguration is based on 2  1. Active IODF                             |
|                                  2. Currently accessed IODF                 |
|                                                                             |
| Scope of discovery  . . . . . 2  1. New controllers only                    |
|                                  2. All controllers                         |
|                                  3. Controller containing CU  ____     +    |
|                                  4. Controller with S/N       0L30002421__  |
|                                                                             |
| Show proposed definitions . . 1  1. Yes   2. No                             |
| Force full mode discovery . . 2  1. Yes   2. No                             |
| Tolerate incapable systems    1  1. Yes   2. No                             |
|                                                                             |
| Target IODF name  . . . 'KARAN.IODF16.WORK'                          +      |
|                                                                             |
|                                                                             |
|                                                                             |
+-----------------------------------------------------------------------------+
```

Figure 6-1 Discovery and Autoconfiguration Options in HCD

As Figure 6-2 shows, you have the equivalent processing options for HCM. Under “Scope of discovery,” click the **Controller with serial number** radio button and enter the serial number to limit discovery.

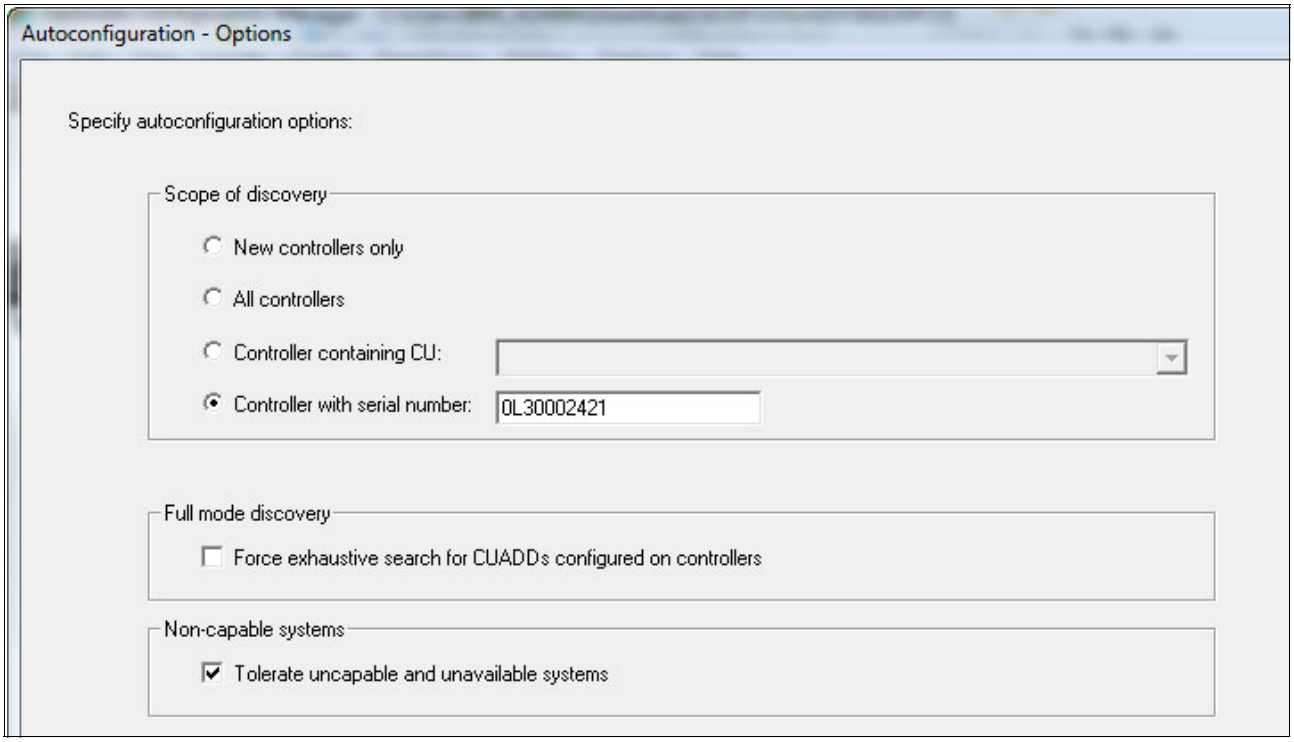


Figure 6-2 Autoconfiguration options for scope of discovery in HCM

6.1.2 Directly attached (non-switched) controllers in discovery

Previously, only switch-attached controllers were considered for proposal. The limitation of the FICON discovery process to switch-attached controllers has been removed. FICON point-to-point (dedicated connection) paths are included in the discovery process. With this change, I/O autoconfiguration (zDAC) discovers directly attached controllers and proposes point-to-point paths if they are available.

6.1.3 User-assigned control unit and device numbers

Before this enhancement, the device numbering policy options might not have worked well for every client. This change provides more control of the control unit and device numbering, so you can follow your installation conventions.

Another profile option allows controller discovery to occur without device numbering. The `AUTO_SS_DEVNUM_SCHEME` profiles in both HCD and HCM now accept a policy option of `NONE`. When this is in effect, auto discovery bypasses device and control unit number processing. HCD and HCM then present newly discovered control unit and devices, which allows you to manually enter values that are validated by HCD or HCM.

HCD and HCM users can request to provide the control unit and device numbers. This is indicated in `AUTO_SS_DEVNUM_SCHEME=NONE` autoconfiguration policy. If that is specified, Input/Output supervisor (IOS) skips the numbering phase for new control unit and devices.

In that case, HCD presents discovered control units with a CU number of 0000 in the Proposed Control Unit List as shown in Figure 6-3.

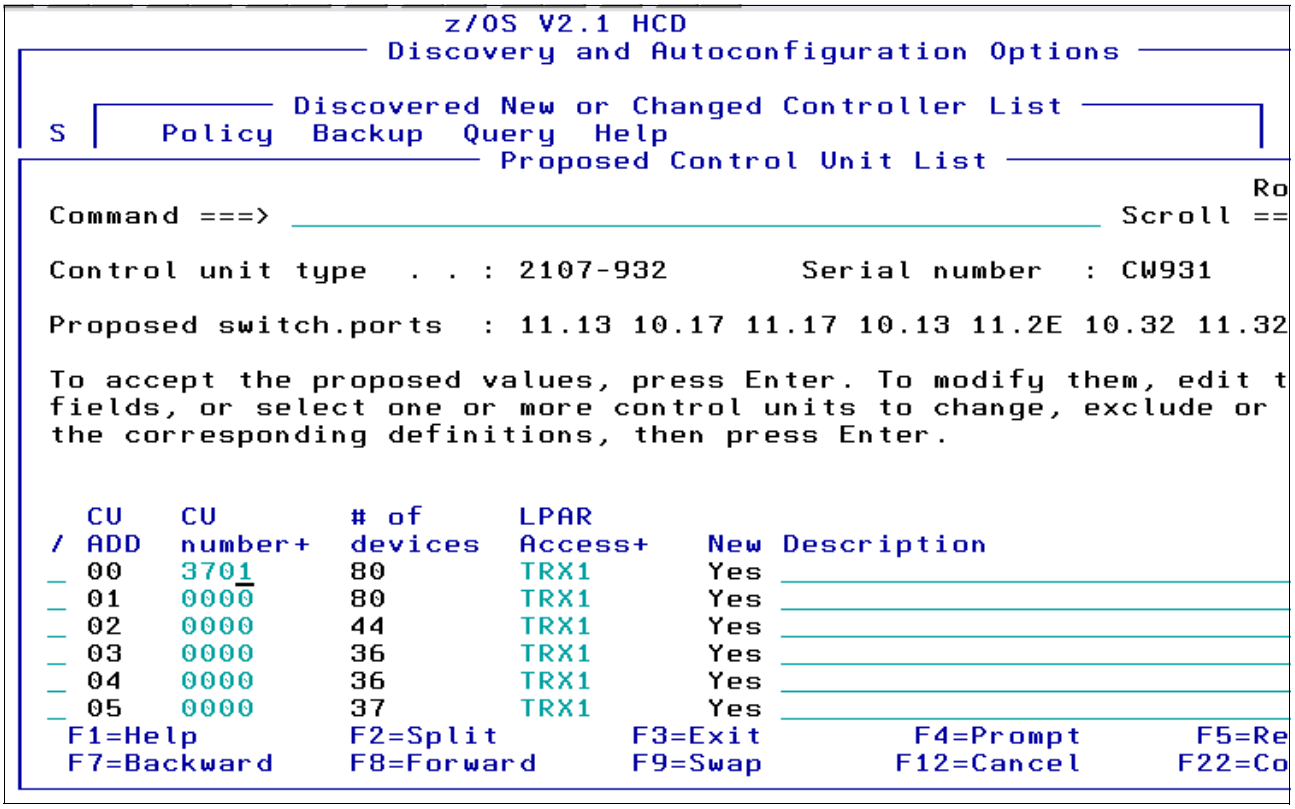


Figure 6-3 Proposed Control Unit List

HCD presents new discovered devices with a starting device number of 0000 in the Proposed Control Unit / Device List panel, as shown in Figure 6-4.

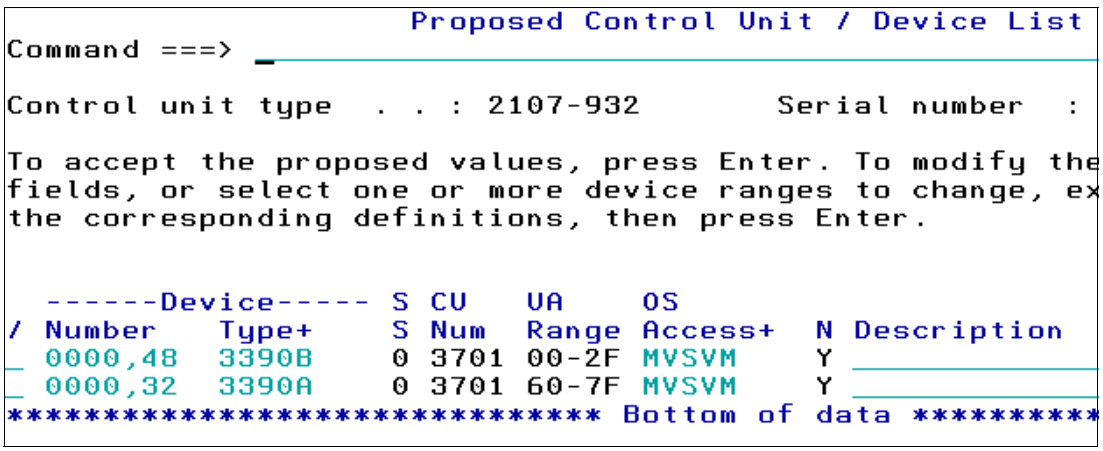


Figure 6-4 Proposed Control Unit / Device List

You can then assign CU numbers and starting device number ranges by editing the appropriate column.

Support for these enhancements is also included in HCM.

6.1.4 Channel path or switch inclusion and exclusion lists

When using IOS zDAC (HCD and HCM), you can now supply switch inclusion or exclusion maps. If they are provided, IOS discovery and autoconfiguration limits controllers returned to those that are accessible by using the supplied inclusion and exclusion criteria. Path selection is also bounded by the inclusion and exclusion criteria. These inclusion and exclusion maps do not affect any controller that is connected through point-to-point criteria. Switch inclusion or exclusion maps have no effect on point-to-point paths.

The HCD and HCM autoconfiguration policies are enhanced to allow the zDAC user to specify inclusion or exclusion lists for switch addresses. These lists are converted by HCD to a switch address inclusion map.

You can also supply CHPID inclusion and exclusion lists. If they are provided, IOS discovery and autoconfiguration limits controllers returned to those that are accessible by using the supplied inclusion or exclusion criteria. CHPID inclusion or exclusion maps must be provided for each target system. Path selection is also bounded by the inclusion and exclusion criteria.

You can now control which channel paths and switches to include for discovery and definition and which switches or channel paths to use or not to use for the discover and propose processes. Therefore, it is possible to restrict the discovery of new or changed controllers to only specific switches of the fabric or to use only specific channel paths for the discovery process and the proposed connectivity to new controllers. The specification is done in the HCD profile with Autoconfiguration Policies profile options.

The keywords are described in the subsections that follow.

AUTO_CHPID_EXCLUDE

This specifies a single CHPID number or a range of CHPID numbers that are excluded from being used for discovery or channel path assignment for a specific channel subsystem (by providing its ID) or all channel subsystems (by specifying with an asterisk: *) of a certain processor. The keyword can occur multiple times.

Figure 6-5 on page 52 specifies that CHPIDs 80, 8A - 8E, and 8F are not used for channel subsystem SCZP401.0. CHPID 8F is also not used for all other channel subsystems of processor SCZP401.

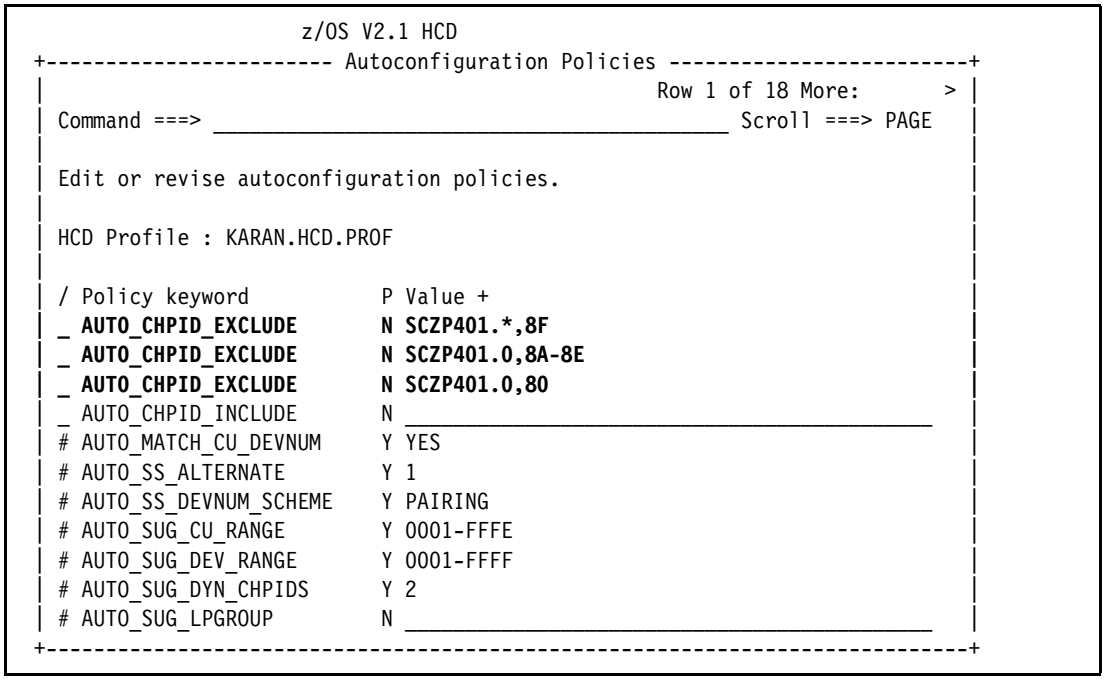


Figure 6-5 AUTO_CHPID_EXCLUDE

AUTO_CHPID_INCLUDE

This specifies a single CHPID number or a range of CHPID numbers to be considered for discovery and channel path assignment for a specific channel subsystem (by providing the ID) or all channel subsystems (by specifying with an asterisk: *) of a certain processor. The keyword can occur multiple times. If this option is specified, no other channel paths are used.

Figure 6-6 on page 53 specifies specify that CHPIDs with numbers 90, 9A-9E, or 9F are used for discovery and definition from the SCZP401.0 channel subsystem. CHPID 9F may also be used from all other channel subsystems of the PROCA processor. Other CHPIDs are not used.

```
                      z/OS V2.1 HCD
+------------------------ Autoconfiguration Policies -------------------------+
|                                                    Row 1 of 15 More:       > |
| Command ===> ___________________________________________ Scroll ===> PAGE    |
|                                                                              |
| Edit or revise autoconfiguration policies.                                   |
|                                                                              |
| HCD Profile : KARAN.HCD.PROF                                                 |
|                                                                              |
| / Policy keyword           P Value +                                         |
| _ AUTO_CHPID_EXCLUDE       N _____________________________________________   |
| _ AUTO_CHPID_INCLUDE       N SCZP401.*,9F                                    |
| _ AUTO_CHPID_INCLUDE       N SCZP401.0,9A-9E                                 |
| _ AUTO_CHPID_INCLUDE       N SCZP401.0,90                                    |
| # AUTO_MATCH_CU_DEVNUM     Y YES                                             |
| # AUTO_SS_ALTERNATE        Y 1                                               |
| # AUTO_SS_DEVNUM_SCHEME    Y PAIRING                                         |
| # AUTO_SUG_CU_RANGE        Y 0001-FFFE                                       |
| # AUTO_SUG_DEV_RANGE       Y 0001-FFFF                                       |
| # AUTO_SUG_DYN_CHPIDS      Y 2                                               |
| # AUTO_SUG_LPGROUP         N _____________________________________________   |
| # AUTO_SUG_OSGROUP         N _____________________________________________   |
| # AUTO_SUG_STAT_CHPIDS     Y 6                                               |
| _ AUTO_SWAD_EXCLUDE        N _____________________________________________   |
+------------------------------------------------------------------------------+
```

Figure 6-6 AUTO_CHPID_INCLUDE

AUTO_SWAD_EXCLUDE

This specifies a single switch address or a range of switch addresses that are excluded from being used for discovery or channel path assignment. (Switch addresses must be specified, rather than switch IDs.) The keyword can occur multiple times.

If no value is specified, no switch is excluded from the discovery and channel path proposal.

Figure 6-7 on page 54 specifies that switches with addresses 04 and 20 to 2F are not used for discovery and channel path assignments.

```
                      z/OS V2.1 HCD
+------------------------ Autoconfiguration Policies -------------------------+
|                                                    Row 1 of 14 More:       > |
| Command ===> ___________________________________________ Scroll ===> PAGE    |
|                                                                              |
| Edit or revise autoconfiguration policies.                                   |
|                                                                              |
| HCD Profile : KARAN.HCD.PROF                                                 |
|                                                                              |
| / Policy keyword           P Value +                                         |
| _ AUTO_CHPID_EXCLUDE       N _____________________________________________   |
| _ AUTO_CHPID_INCLUDE       N _____________________________________________   |
| # AUTO_MATCH_CU_DEVNUM     Y YES                                             |
| # AUTO_SS_ALTERNATE        Y 1                                               |
| # AUTO_SS_DEVNUM_SCHEME    Y PAIRING                                         |
| # AUTO_SUG_CU_RANGE        Y 0001-FFFE                                       |
| # AUTO_SUG_DEV_RANGE       Y 0001-FFFF                                       |
| # AUTO_SUG_DYN_CHPIDS      Y 2                                               |
| # AUTO_SUG_LPGROUP         N _____________________________________________   |
| # AUTO_SUG_OSGROUP         N _____________________________________________   |
| # AUTO_SUG_STAT_CHPIDS     Y 6                                               |
| _ AUTO_SWAD_EXCLUDE        N 04                                              |
| _ AUTO_SWAD_EXCLUDE        N 20-2F_______________________________________    |
| _ AUTO_SWAD_INCLUDE        N _____________________________________________   |
+------------------------------------------------------------------------------+
```

Figure 6-7 AUTO_SWAD_EXCLUDE

AUTO_SWAD_INCLUDE

This specifies a single switch address or a range of switch addresses that are to be used for discovery and channel path assignment. (Switch addresses must be specified, rather than switch IDs). The keyword can occur multiple times. If this option is specified, no other switches are used.

If no value is specified, all switch addresses in the range 00 to FF can be used for the discovery and channel path proposal.

Figure 6-8 on page 55 specifies that switches with addresses 04 or 20 - 2F can be used for discovery and channel path assignments. No other switches are used.

```
                      z/OS V2.1 HCD
+------------------------ Autoconfiguration Policies -------------------------+
|                                                    Row 1 of 14 More:      > |
| Command ===> _____________________________________________ Scroll ===> PAGE   |
|                                                                              |
| Edit or revise autoconfiguration policies.                                   |
|                                                                              |
| HCD Profile : KARAN.HCD.PROF                                                 |
|                                                                              |
| / Policy keyword           P Value +                                         |
| _ AUTO_CHPID_EXCLUDE       N _____________________________________________   |
| _ AUTO_CHPID_INCLUDE       N _____________________________________________   |
| # AUTO_MATCH_CU_DEVNUM     Y YES                                             |
| # AUTO_SS_ALTERNATE        Y 1                                               |
| # AUTO_SS_DEVNUM_SCHEME    Y PAIRING                                         |
| # AUTO_SUG_CU_RANGE        Y 0001-FFFE                                       |
| # AUTO_SUG_DEV_RANGE       Y 0001-FFFF                                       |
| # AUTO_SUG_DYN_CHPIDS      Y 2                                               |
| # AUTO_SUG_LPGROUP         N _____________________________________________   |
| # AUTO_SUG_OSGROUP         N _____________________________________________   |
| # AUTO_SUG_STAT_CHPIDS     Y 6                                               |
| _ AUTO_SWAD_EXCLUDE        N _____________________________________________   |
| _ AUTO_SWAD_INCLUDE        N 04                                              |
| _ AUTO_SWAD_INCLUDE        N 20-2F                                           |
+------------------------------------------------------------------------------+
```

Figure 6-8 AUTO_SWAD_INCLUDE

Note: Switches are specified by their switch addresses, not by their switch IDs.

These policy options are also available for HCM.

6.1.5 Policy change between controller discoveries

HCD and HCM now allow a zDAC user to change I/O autoconfiguration policies between two subsequent controller discoveries, without the need to make a new fabric discovery as in z/OS V1.12. This enables IOS to detect changes in policy for each new controller discovery.

In HCD, the POLICY has been added in the action bar of the Discovered New or Changed Controller List dialog. In HCM, on the Discovered Controller List page, the Policies button option can be used to respecify autoconfiguration policies.

You can change your defined autoconfiguration policies between two subsequent controller discoveries without the need for a new fabric discovery. For this purpose, in the Discovered New or Changed Controller List dialog, select the **Policy** action bar choice, and then select **1**. Change policy options to start the Autoconfiguration Policies dialog. You can now change the values of certain keywords that you want to apply on the subsequent controller discovery and control unit autoconfiguration. However, only changes for those keywords become effective immediately for which HCD sets column P to Y in the Autoconfiguration Policies, for example, AUTO_MATCH_CU_DEVNUM.

Changes of all policy keywords that are denoted with value N in column P are also possible between two controller discoveries. However, those require a new fabric discovery to become effective.

6.1.6 Toleration of incapable systems

A zDAC request that contains target systems that are a previous level or not capable, such an earlier release with lower maintenance, before the IBM zEnterprise 196 (z196) processor, failed with this error message: REASON=01E2 or REASON=01E3.

Now, HCD and HCM can request that the process continue with the remaining systems. IOS issues the existing messages, as usual (NOTE = 01E2 or NOTE = 01E3) but continues with the appropriate systems.

HCD and HCM need to indicate that unavailable or incapable systems can be ignored in the IOS zDAC invocation. It can limit the target systems to a subset of the LPAR group or Parallel Sysplex if it detects that systems are unavailable.

As shown in Figure 6-1 on page 48, you can skip systems that are not able to perform discovery. Select option **1 (YES)** for **Tolerate incapable systems** to skip systems.

If set to *Yes*, for each discovered controller, discovery continues processing if the active LP group contains systems that do not fulfill the conditions for autoconfiguration. Such systems are ignored and excluded from the LP group. If set to *No* (which is the default), processing stops if the active LP group contains partitions with systems that are not capable of autoconfiguration.

As shown in Figure 6-2 on page 49, you have the equivalent processing options on HCM. Select **Non-capable incapable systems > Tolerate uncapable and unavailable systems** to skip systems.

6.1.7 Save discovery and auto-definition results

HCD implements the **SAVE** command to allow the zDAC user to save the fabric and controller discovery list panels in a data set.

You can now use the **SAVE** command for the following I/O autoconfiguration lists:

- Discovered New or Changed Controller List
- Proposed Control Unit List
- Proposed Control Unit / Device List

From one of these list dialogs, enter the **SAVE** command on the HCD command line. The Save List dialog then prompts you to enter a name for the data set where you want to save the list.

6.2 Verify Configuration function with I/O autoconfiguration

This section explains the following enhancements to the Verify Configuration function (I/O path report):

- Verify an active or target configuration by using z/OS Discovery and Autoconfiguration (zDAC).

  Before this change, the option to verify configuration was available only if IBM Tivoli System Automation I/O operations was installed.

For processors that use zDAC, including IBM zEnterprise EC12, BC12, zEnterprise 196 (z196), and zEnterprise 114 (z114) systems, automated verification is used to discover the active configuration if Tivoli System Automation I/O not operational. The verification is limited to Fibre Channel connection (FICON)-attached storage devices.

- Provide single point of failure information.

  There was no information about a single point of failure (SPOF) for active paths within HCD. This change integrates information that is provided by the OSSPOF macro in the I/O path report so that you can easily identify single points of failure.

6.2.1 Invocation and use

The paths to start verifying a configuration by using zDAC are the same as starting the verification functions with Tivoli System Automation I/O operations before, either with option 2.6 against the local system or with 2.7 against a system in the Parallel Sysplex.

For option 2.6, use a production IODF, select option 2 to activate or process configuration data, and then select option 6 to activate or verify configuration. Now, as shown in Figure 6-9, select option 4 or 5, and then identify the target.

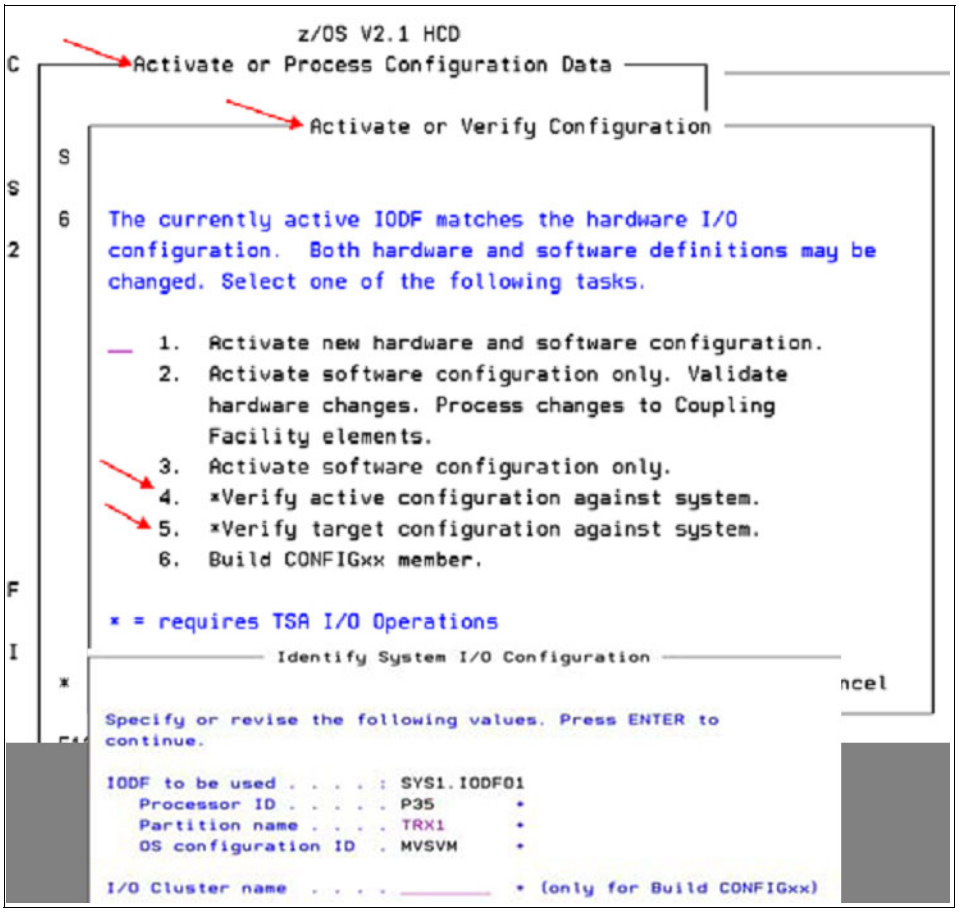


Figure 6-9 Using option 2.6 to verify configuration with zDAC

For option 2.7, use a production IODF, select option 2 to activate or process configuration data, and then select option 7 to activate configuration Parallel Sysplex-wide). Now. as shown in Figure 6-10 on page 58, select option 9 or 10 and identify the target.

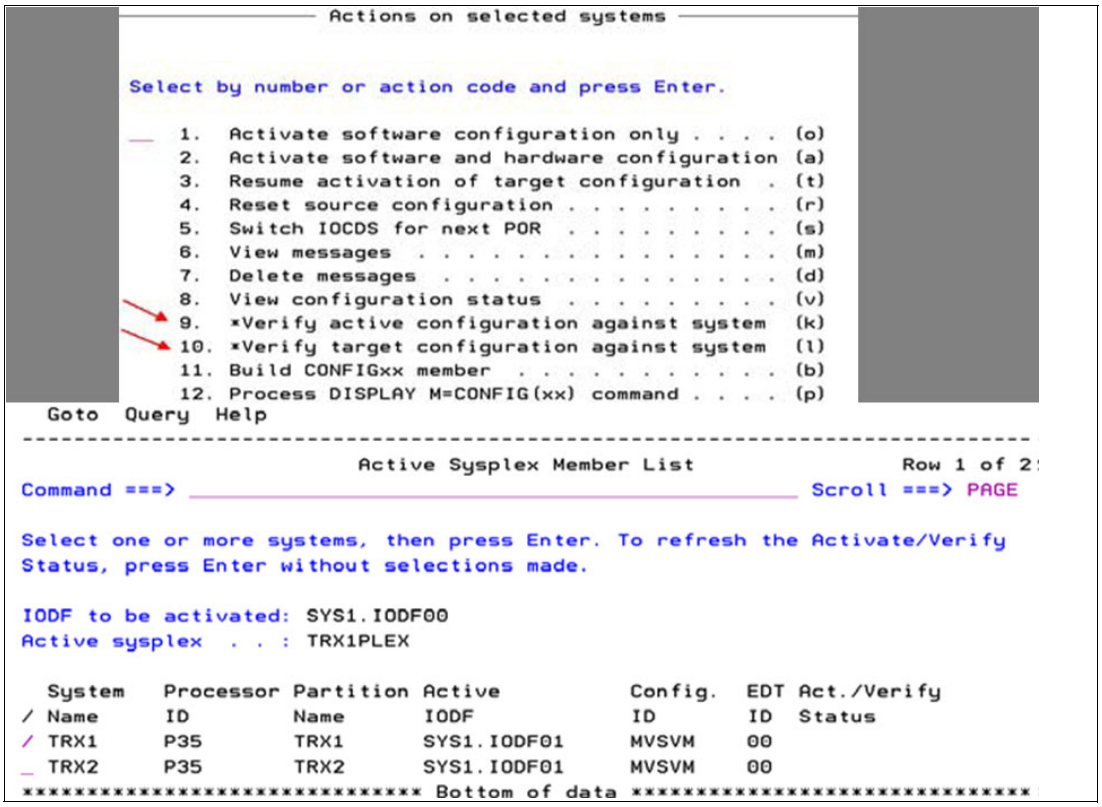


Figure 6-10 Using option 2.7 to verify configuration via zDAC

When Tivoli System Automation I/O operations are not available, HCD tries to retrieve the active configuration via zDAC. An informational message is issued, as shown in Figure 6-11.

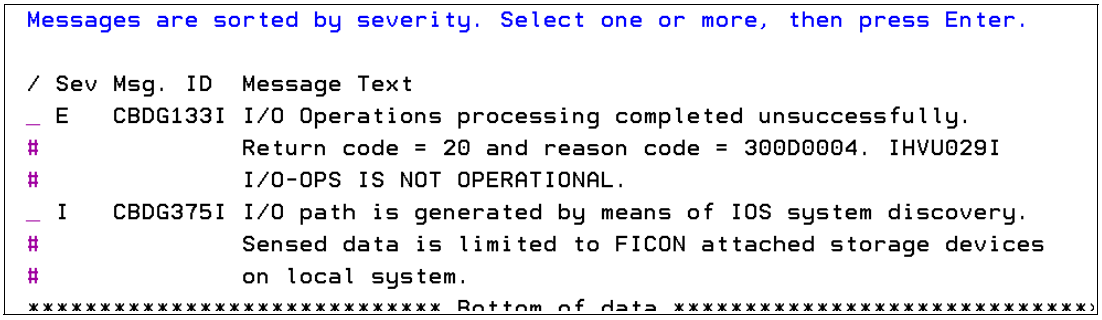


Figure 6-11 Informational message about zDAC being used

The I/O path report now has information that is provided by the **IOSSPOF** macro. The STAT column was changed and now indicates a possible Single Point of Failure (SPOF), as shown in Figure 6-12 on page 59.

```
  Goto  Filter  Backup  Query  Help
------------------------------------------------------------------------------
                              I/O Path List     Row 5 of 1515 More:       >
Command ===> ____________________________________________ Scroll ===> PAGE

Accessed IODF: SYS1.IODF01
Active IODF  : SYS1.IODF01                              System . : THIS-SYS
Processor . .: P35        Partition : TRX1              OS config: MVSVM

----I/O Path----- ----------Sensed Data--------- ----IODF Data---------- D
CHP CU   DEV      STAT CHT   CUTYPE   DEVTYPE  O CHT CUTYPE   DEVTYPE  S
0D  0621 0620,16                                OSM OSM       OSA-M      *
30  2000 2000,2   5    FC_S  2107-9A2 3390B      FC  2107      3390B
30  2000 2002,17       FC_S  2107-9A2 3390B    Y FC  2107      3390B
30  2000 2013     5    FC_S  2107-9A2 3390B      FC  2107      3390B
30  2000 2014,14       FC_S  2107-9A2 3390B    Y FC  2107      3390B
30  2000 2022,4   5    FC_S  2107-9A2 3390B      FC  2107      3390B
30  2000 2026,9        FC_S  2107-9A2 3390B    Y FC  2107      3390B
30  2000 202F     5    FC_S  2107-9A2 3390B      FC  2107      3390B
30  2000 2030,32       FC_S  2107-9A2 3390B    Y FC  2107      3390B
```

Figure 6-12 Changed STAT column showing SPOF

The information in the STAT column has up to four character positions:

1. Operational (blank).
2. If the report is requested for the local system, it contains the status that is returned from IOCINFO PATHMAP for the channel path. For all operational paths (either via IOCINFO or I/O Operations API), there is a check for single point of failure. A map of the four indicators is shown in Table 6-1.
3. Not operational (OFFL).
4. Unknown (UNKN).

Table 6-1 STAT column map for an operational path

Position 1	Position 2	Position 3	Position 4
BLANK or number between 3 and 8: 3 BOOK 4 CAGE 5 FAILOVER DOMAIN 6 FANOUT 7 DOMAIN 8 SECONDARY STI/STI	C If all CU interfaces share a single point of failure	P If the device has only one path online	S If all paths go through one switch

These are the descriptions for the CHT, CUTYPE, DEVTYPE, and O fields:

CHT	The channel path type returned from IOSCHPT. For discovery, this information is available only if the report is for the local system.
CUTYPE	If zDAC was used, the type is specified in the node descriptor of the controller.
DEVTYPE	If zDAC was used, the device type is the type found in the IODF. This field is filled only if the report is issued for the local system and if there is a token match with the active IODF.
O	If zDAC was used, the state is the state that is returned from UCBSCAN. This field is filled only if the report is issued for the local system and if there is a token match with the active IODF.

6.2.2 Interactions and dependencies

Verifying a configuration automatically requires a zDAC-capable server, such as the zEC12, zBC12, z196, or z114 servers.

6.3 HMC-wide activation of dynamic I/O changes

HCD provides a function called *HMC-wide activate* to manage dynamic I/O changes from a single managing system. It manages changes for the target systems that are running on central processor complexes (CPCs) that are controlled by the same HMC as the managing CPC. This capability provides a single point of management control for dynamic activations across all servers and logical partitions (LPARs) that are controlled by the same HMC.

With a managing HCD system, you can define, distribute, and activate the I/O configuration for all other systems that are controlled by the same HMC. The managing system handles the following functions:

- Lists all images on the HMC-controlled CPCs with the activation status (deactivated, activated, IPLed), system name, Parallel Sysplex name, operating system type, and version.
- Distributes a production IODF to selected target systems.
- Performs dynamic hardware or software changes remotely on selected target systems.
- Remotely operates the target system that is related to the activation (including issuing IBM MVS commands).

The function is integrated in the existing dialog, in the IBM System z cluster list, to manage the IOCDS and IPL parameters in an HMC-controlled CPC cluster. This works on both z/OS and IBM z/VM operating systems.

Actions can be initiated from the CPC Image List dialog, where image information is displayed. The image information of the systems is obtained from the HMC and SE by z/OS Base Control Program internal interface (BCPii) calls. To get the activation status of z/OS and z/VM systems, distributing IODFs or performing dynamic activates or other system commands, HCD makes remote calls to the HCD versions that are running on the target systems. The results of the actions are reported to the managing system.

6.3.1 Installation

HCD uses BPCii calls to query the image (partition) attributes from the Support Element (SE) of the selected CPC. To get the partition attributes, the following requirements must be met (see *MVS Programming: Callable Services for High-Level Languages*, SA23-1377 for details about setting up the BCPii requirements):

- The BCPii address space (HWIBCPII) must be active and ready to handle BCPii requests on the LPAR that is used as the HCD managing system. This is required because the HCD managing system issues BCPii calls that must be handled by the BCPii address space on the local system so that HCD can gather image information from local and remote CPCs. The BCPii address space does not have to be enabled on any remote systems to use the HMC-wide activate function.
- The local and remote support elements must be enabled for BCPii communication (cross-partition authority must be enabled for each image on the CPC that you want to grant BCPii access). HCD cannot retrieve image information through BCPii for those partitions where cross-partition authority is not enabled.
- The user of the HCD HMC-wide activate function must have the authority to perform the BCPii calls for the target CPCs and images.

The HMC-wide activate function uses TCP/IP to communicate with remote managed systems. This requires that a TCP/IP connection is available between the HCD managing system and the target systems.

An HCD dispatcher (the same as used for HCM) must be run on each of the target systems that allows directing incoming remote HCD requests to the local HCD versions.

The connection data is provided in a connection table. For HCD to establish communication with remote target systems, you must provide a connection table that contains the TCP/IP login data for the remote systems.

The data set name for the connection table is specified in the HCD profile keyword CONNECTION_TABLE. PassTicket support is available as option to avoid specifying sensitive data, such as a password, in a data set. Figure 6-13 shows an example of the syntax of the profile keyword CONNECTION_TABLE.

```
/* ************************************************************** */
/*                                                                */
/*    HCD Profile Section for Standard Profile Options            */
/*                                                                */
/* ************************************************************** */
CONNECTION_TABLE = KARAN.HCD.CONN
```

Figure 6-13 CONNECTION_TABLE example

CONNECTION_TABLE specifies the name of the data set or partitioned data set member of the TCP/IP connection table that is used for CPC-wide dynamic activate. The connection table is allocated with data definition (DD) name HCDCNTL. If the keyword is not specified and no data set is allocated with HCDCNTL, no connection table is used.

To establish connectivity to remote systems, connection data must be provided in a connection table that contains the following information:

- Processor SNA address with network name and CPC name, as defined for the corresponding processors in the IODF and as known in the HMC/SE.

- Partition name (image) as specified in the IODF and configured in the SE.
- IP address or host symbolic destination name for the TCP/IP target system where the HCD dispatcher program is running.
- IP port number that is used by the HCD dispatcher program at the remote site.
- User ID for the remote system.
- Password for the user ID on the remote system. If you do not provide a password, HCD uses a PassTicket for verifying the authorization for the user ID on the remote system. Corresponding SAF definitions must be in place for PassTicket support.

Figure 6-14 is a sample connection table.

```
*********************************************************************************************
* NETWORK NAME                  IMAGE        IPADDR                       PORT,USERID,PASSWORD
*********************************************************************************************
USIBMSC,SCZP401                 ,A08         ,WTSC80.ITSO.IBM.COM         ,51107,KARAN   ,xxxxxxxx
USIBMSC,SCZP401                 ,A19         ,WTSCTA.ITSO.IBM.COM         ,51107,KARAN   ,xxxxxxxx
```

Figure 6-14 Connection table example

In addition to the CONNECTION_TABLE keyword, the following profile options have been added:

RCALL_LOG — YES or NO. Use this option to activate logging of remote calls. The default is NO. Setting this option to YES allocates a new data set named HLQ.cbdqclnt.log, if it does not yet exist. Otherwise, an existing data set is used.

RCALL_TIMEOUT — Specifies the maximum number of seconds that a remote call waits for a response at logon time before a timeout occurs. The default value is 60. This default is also used if the value is set to 0.

Connections: Depending on the installation's topology, a connection might not be needed for every managed system. A connection to a partition on every CPC is not required, either. It is possible to use the HMC-wide activate function with only a single Parallel Sysplex connection. Hardware and software can be activated through a single connection to a member of a Parallel Sysplex.

For example, if the managing HCD system (SYSA) is on CPC1 and has a connection to the SYSX managed remote system (a member of PLEXA Parallel Sysplex) on CPC2, software can be activated for SYSX and hardware can be activated on CPC2 through SYSX.

Also, suppose that the PLEXA Parallel Sysplex consists of members named SYSY on CPC3 and SYSZ on CPC4. In this case, software can be activated through the connection to SYSX for SYSY and SYSZ. Hardware can be activated on CPC3 and CPC4 through the connection from SYSX. If the managing HCD system has a connection to one member of a remote Parallel Sysplex, the HCD on the remote sysplex can use the existing sysplex-wide activation feature.

6.3.2 Invocation and use

The HMC-wide activate dialogs are accessed from HCD option 2.6 (**Define, modify, or view configuration data → Build and manage System z cluster IOCDSs, IPL attributes, and dynamic I/O changes**). The System z Cluster List dialog is displayed from HCD option 2.6, as shown in Figure 6-15. A list of CPCs that are defined in the current IODF is displayed.

```
Goto  Query  Help
--------------------------------------------------------------------------------
                           System z Cluster List                  Row 1 of 4
Command ===> ____________________________________________ Scroll ===> PAGE

Select one or more CPCs, then press Enter.

  -------------CPC-------------   IODF
/ SNA Address        Type   Model   Processor ID
_ USIBMSC.SCZP101    2094   S18     SCZP101
_ USIBMSC.SCZP201    2097   E26     SCZP201
_ USIBMSC.SCZP301    2817   M32     SCZP301
_ USIBMSC.SCZP401    2827   H43     SCZP401
_
****************************** Bottom of data ********************************
```

Figure 6-15 System z Cluster List

Selecting line command option **v** or option **4** (Work with CPC images) from the menu initiates system discovery process.

After the system discovery process finishes, the CPC Image List panel is displayed. Figure 6-16 shows a partial and abbreviated display of the CPC Image list.

```
                              CPC Image List       Row 1 of 52 More:       >
Command ===> ____________________________________________________ Scroll ===> PAGE

Select one or more systems of the CPC, then press Enter. To refresh the
Activate/Verify Status, press Enter without selections made.

Processor ID : SCZP401        SNA Address  : USIBMSC.SCZP401
HSA Token  . : SCZP401  13-06-26 11:31:00 SYS6     IODF15

  -- Partition --- Co Sysplex   ------- System -------  ----- Activation -----
/ Name     ID Stat St Name      Name     Type   Level   HW SW Status
_ A0A      0A 0001    ITSOZVM   ITSOZVM3 VM     6.3
# A0B      0B 0002
_ A01      01 0001 S  PLEX75    SC74     MVS    2.1     Y  Y
_ A08      08 0001 Y  WTSCPLX8  SC80     MVS    2.1     Y  Y
_ A09      09 0001 S  WTSCPLX8  SC81     MVS    2.1     Y  Y
_ A19      19 0001 Y  #@$#PLEX  #@$A     MVS    2.1     Y  Y
```

Figure 6-16 CPC Image List page 1

Scrolling right displays additional information for each partition, as shown in Figure 6-17.

```
                              CPC Image List       Row 1 of 52 More: <
Command ===> ___________________________________________________ Scroll ===> PAGE

Select one or more systems of the CPC, then press Enter. To refresh the
Activate/Verify Status, press Enter without selections made.

Processor ID : SCZP401        SNA Address  : USIBMSC.SCZP401
HSA Token  . : SCZP401  13-06-26 11:31:00 SYS6     IODF15

  Partition  Active             Conf      EDT
/ Name       IODF               ID        ID
_ A0A
# A0B
_ A01        SYS6.IODF15        LO6RMVS1  01
_ A08        SYS6.IODF15        LO6RMVS1  01
_ A09        SYS6.IODF15        LO6RMVS1  01
_ A19        SYS6.IODF15        TRAINER   01
```

Figure 6-17 CPC Image List Page 2

The following header information is displayed on the CPC Image List:

Processor ID	ID of the processor associated with the SNA address of the CPC
SNA Address	Network name and CPC name of the CPC
HSA Token	Hardware configuration token in the HSA of this CPC

The following information is displayed for each partition on the CPC Image List:

Partition Name	Name of a partition in the processor
Partition ID	ID of the logical partition in the Support Element
Partition STAT	Operation status of the partition:
0001	Operational (activated and a system is IPLed)
0002	Activated but no system is IPLed
0008	Deactivated
0010	Exceptions
	Any other value shows a special condition for the LPAR that can be checked on the SE or HMC.
Co ST	Shows the connection status from the HCD of the local system to the HCD of the shown remote system:
BLANK	A connection does not exist between HCD of the local system to the displayed system.
Y	A connection exists between HCD of the local system to the displayed system because of an entry in the connection table.
S	A connection exists between HCD of the local system to the displayed system because they are either identical or in the same Parallel Sysplex, or another member of the Parallel Sysplex has a direct connection through an entry in the connection table.

N	A connection failed between the local HCD and the displayed system.
Sysplex Name	Name of the Parallel Sysplex to which the system belongs.
System Name	Name of the system running in the displayed partition.
System Type	Operating system type of the system.
System Level	Operating system release level of the system.
Activation HW	Whether a Hardware Activate is possible (Y) or not (N) or it is unknown (?). A hardware activation is not possible if the active IODF is not accessible, the IODF token does not match, or the HSA token does not match the processor token in the active IODF. The indication is shown only for systems that are connected or are in the same Parallel Sysplex as a connected system.
Activation SW	Whether a Software Activate is possible (Y) or not (N), or, if the software is not managed by HCD, a switch IODF is possible (S). A software activation is not possible if the active IODF cannot be accessed or the IODF token does not match. The indication is shown only for systems that are connected or which are in the same Parallel Sysplex as a connected system.
Activation Status	Indicates one of the following statuses:
Blanks	No dynamic I/O activate has been issued to the system and no messages are available.
Activating	A dynamic I/O activate command has been initiated to this system.
In Progress	A dynamic I/O activate is currently in progress, and no messages have been issued yet.
Executing	A system command has been initiated to this system.
Messages	A dynamic I/O activate or other request has been initiated and completed and has issued one or more messages.
Active IODF	Name of the currently active IODF.
Conf ID	ID of the active operating system configuration.
EDT ID	ID of the EDT that has active configuration data.

From the CPC Image List, the actions shown in Figure 6-18 on page 66 are available to be performed on selected images.

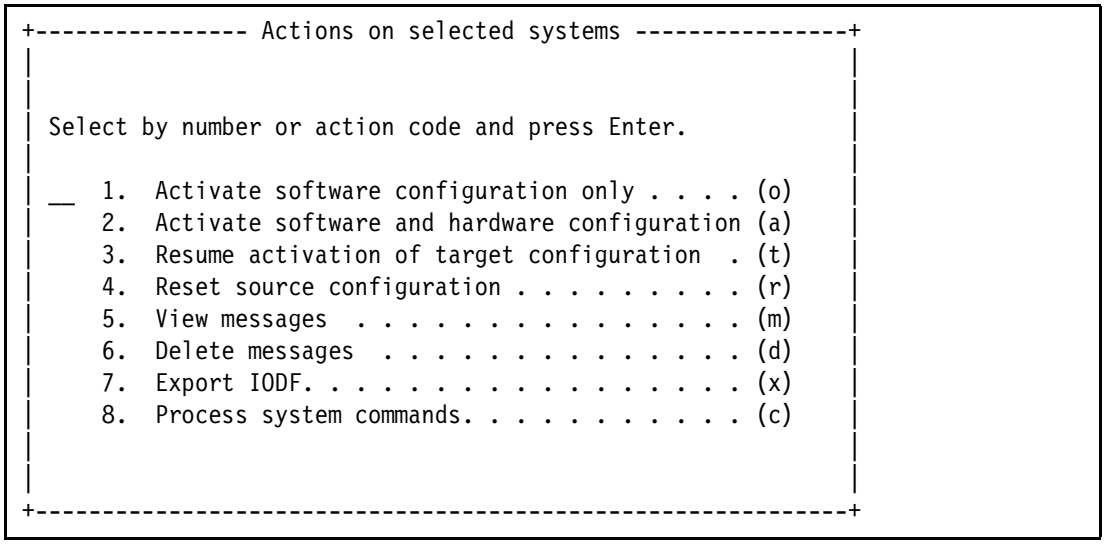


Figure 6-18 Available actions on selected systems

The HMC-wide activate function checks whether the IODF to be activated is available on the remote site. The result triggers these actions:

- If the IODF exists and the token of the IODF found on the remote system matches the token of the managing system's currently accessed IODF, the remote IODF is used for the activation.
- If IODF does not exist on the remote system or the token does not match that of the managing system's currently accessed IODF, the target production IODF is transferred to the remote system with an HLQ that matches the remote system's currently active IODF and allocated on the same volume as the remote systems's currently active IODF. The user is prompted for verification if these actions result in any data set deletions.

See *Hardware Configuration Definition User's Guide*, SC34-2669 for a full explanation of the HMC-wide activate options.

6.4 Unconditional generation of DR site OS configurations

If Peer-to-Peer Remote Copy (PPRC) devices are deleted, their connections are removed from both the primary and secondary (DR site) OS configurations. This causes the "generated" attribute of the secondary OS configuration to be deleted. Further updates to the primary OS configuration are not automatically reflected in the secondary OS configuration.

An added profile option enables you to generate a new DR site OS configuration at production IODF build time, regardless of whether the generated OS configuration has been changed or not. You do not have to manually delete the previously generated OS configuration to generate a new one when the production IODF is built.

6.4.1 Invocation and use

The HCD profile offers the UNCOND_GENERATE_DROS option, with two possible responses:

- If set to NO (the default), it preserves the previous behavior that causes the generation of a new DR site OS configuration only if the old one has not been modified.

- If set to YES, HCD regenerates DR site OS configurations whenever a new production IODF is built, independent of whether the configurations have been modified previously or not. This helps to avoid manual user interventions in cases where changes on the primary configuration are not automatically applied to the DR site OS configuration.

6.5 HCD validation

The subsections that follow describes the validation enhancements to the HCD:

6.5.1 Alert for ACTIVATE command with NOVALIDATE

When an **ACTIVATE SOFT=VALIDATE** command is issued, any changes in coupling facility connectivity are processed from the passed hardware changes. This is not done with the **NOVALIDATE** option.

Now, a CBDA854I message, as shown in Example 6-1, is issued when **ACTIVATE SOFT=NOVALIDATE** is used. That warns you that changes to CF elements are not processed.

Example 6-1 CBDA854I message

```
CBDA854I Changes to Coupling Facility elements are not processed.
```

6.5.2 Alert when CF CIB connection changes for connectivity updates

When removing the lowest CSS of a spanned CIB channel path, the connected target CIB channel path is implicitly changed. This requires an activation for the connected processor. If this is not done, the CIB channel path loses its connectivity.

When changing or editing the partition lists of a spanned CIB channel path so that the backward-referenced channel subsystem of the connected target CIB channel path is changed, HCD issues the CBDG422I message that is shown in Example 6-2. This message is also issued when a dynamic activation affects the target processor configuration.

Example 6-2 CBD422I

```
CBDG422I Changing CHPID <source CHPID> changes the lowest CSS to <CSS ID>,
affecting CF connection to <target CHID>. Consider activating the target
LPAR/processor.
```

6.6 HCD OS group change action for device groups

The OS group change action was available only on the I/O device list, which shows single devices, but not on the I/O device list, which shows device groups. Starting with this release, this action is also available from the I/O Device List that shows device groups.

6.6.1 Invocation and use

You can now select multiple device ranges for changing OS device attributes in one step, which is more efficient. The *OS group change* option is available as an action from I/O Device List panels that show device groups, as shown in Figure 6-19 on page 68.

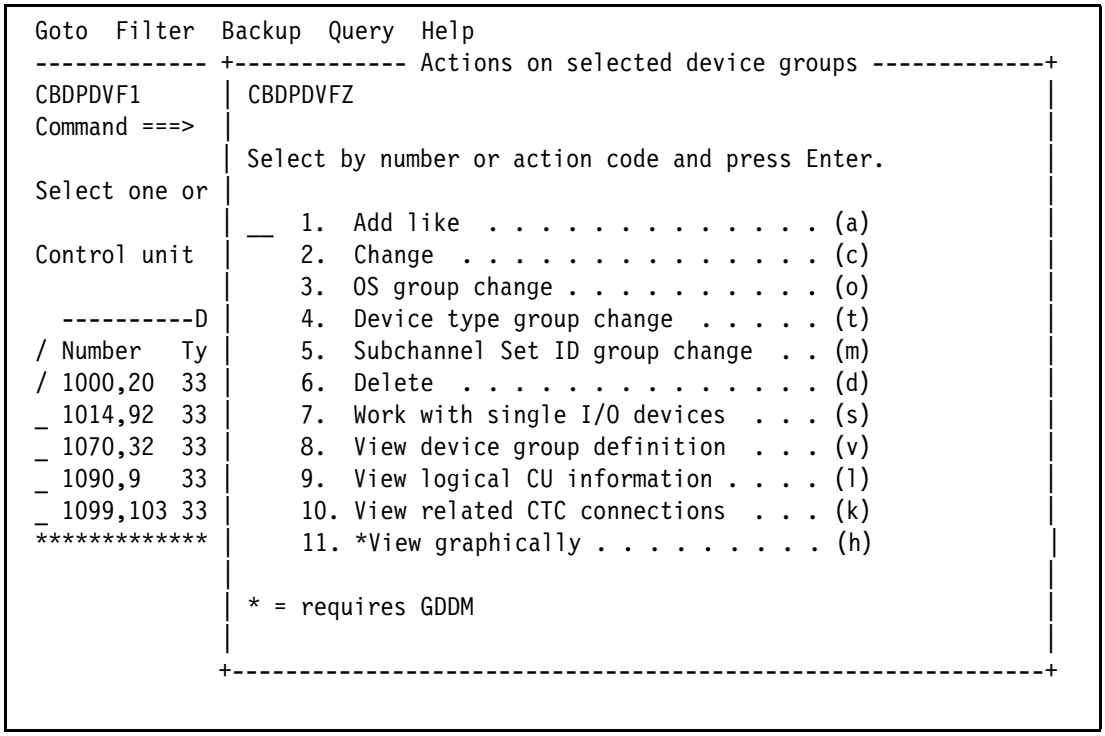


Figure 6-19 OS group change action from device group

6.7 HCD batch utility

The subsections that follow described the HCD batch utility enhancements that became available with z/OS 2.1.

6.7.1 Activate command now available as HCD batch command

Previously, it was not possible for HCD batch jobs to include **ACTIVATE** commands. Therefore, it was more difficult to automate a complete switch of configuration by batch. This change offers **ACTIVATE** as an HCD batch function. You can also write batch jobs with a series of HCD actions, including **ACTIVATE**, with common return code handling, message log, and so on.

This utility function activates an I/O configuration from an existing production IODF. Both the active IODF and the target IODF must be accessible.

This function is started by passing the **ACTIVATE** command in the parameter string (**PARM='...'**), as shown in Example 6-3 on page 69. The parameter string uses the same syntax for the **ACTIVATE** command that is described in *z/OS MVS System Commands*, SA38-0666-01:

https://ibm.biz/BdFc8y

The HCDMLOG data set must be allocated, and the HCDTRACE data set must be allocated if tracing is activated.

Example 6-3 ACTIVATE command by BATCH

```
//* ACTIVATE PRODUCTION IODF
//*
//WORK EXEC PGM=CBDMGHCP,PARM='ACTIVATE IODF=01,TEST'
//HCDMLOG DD DSN=BBEI.HCD.MSGLOG,DISP=OLD
//
```

6.7.2 Filter parameters for graphical reports that are created by a batch utility

Previously, when issuing a graphical control unit report in batch mode, there was no way to filter the report by control unit ranges. Therefore, it was necessary to use the HCD dialog to generate graphics for control unit connections.

Now, as with the HCD dialog for creating graphical reports, you can specify filter parameters when creating graphical reports with the batch utility function.

The added syntax for setting a filter for a graphical control unit report is shown in Figure 6-20.

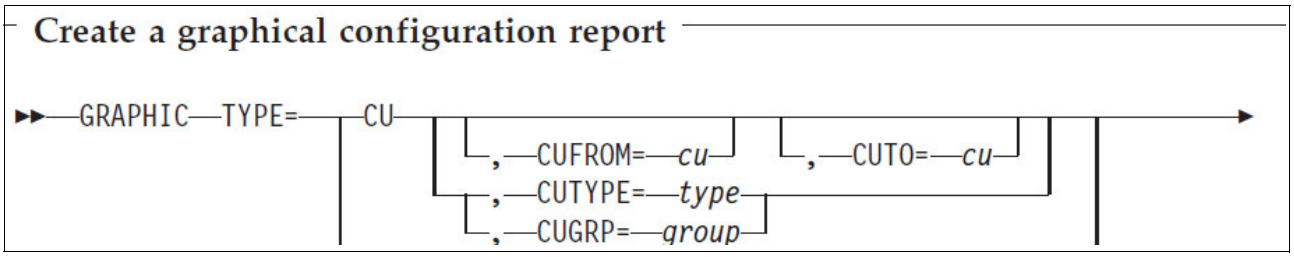


Figure 6-20 Graphic syntax addition

Example 6-4 shows the enhanced syntax in a JCL job.

Example 6-4 JCL

```
//* ------------------------------------------------------
//* Graphical Configuration Report
//* ------------------------------------------------------
//GCREP    EXEC PGM=CBDMGHCP,
//        PARM='GRAPHIC TYPE=CU,CUFROM=0200,CUTO=0480'
//HCDIODFS DD DSN=USER.IODF00.DBR4,DISP=SHR
//HCDRPT   DD DSN=USER.IODF00.DBR4.REPORT,
//            DCB=(RECFM=FBA,LRECL=200,BLKSIZE=6400),
//            SPACE=(TRK,(50,50)),DISP=(NEW,KEEP),UNIT=SYSALLDA
//HCDMLOG  DD DSN=USER.HCD.LOG,DISP=OLD
//HCDPROF  DD DSN=USER.HCD.PROF,DISP=SHR
```

6.8 HCD support of PCIe functions

HCD now has a dialog where you can define PCIe functions, assign them to LPARs, and activate them via IOCP or dynamically.

HCD also provides the following reports:

- The PCIe Function Summary Report displays the partitions in the access and a list of candidates that are entitled to access the available PCIe functions.

- The PCIe Function Compare Report shows the changes in the function and virtual numbers and attributes of PCIe functions between processors of two IODFs.

HCD supports the I/O configuration FUNCTION statement for defining and configuring PCIe functions.

OSD CHPID types and IQD channel definitions allow specification of a Physical Network ID.

Note: PCIe functions require z/OS V2.1 and applicable PTFs (for APAR OA39234).

6.8.1 PCIe function dialog

HCD now has a dialog where you can define PCIe functions, assign them to LPARs, and activate them via IOCP or dynamically. For a zEC12 or zBC12 processor type, HCD supports PCIe functions for the following PCIe hardware features:

- 10GbE RoCE (Remote Direct Memory Access over Converged Ethernet) Express feature
- IBM zEnterprise Data Compression (zEDC) Express feature

The PCIe Function List panel can be accessed from the Processor List panel (as action option **8** or line command character **f**), as shown in Figure 6-21.

```
Goto  Filter  Backup  Query  Help
------------ +-------------- Actions on selected processors ---------------+
             |                                                             |
Command ===> |                                                             |
             | Select by number or action code and press Enter.            |
Select one or|                                                             |
             | __  1.  Add like . . . . . . . . . . . . . . . . (a)        |
             |     2.  Repeat (Copy) processor configurations (r)          |
/ Proc. ID Ty|     3.  Change . . . . . . . . . . . . . . . . . (c)        |
_ ISGSYN   20|     4.  *Prime serial number . . . . . . . . . . (i)        |
_ ISGS11   20|     5.  Delete . . . . . . . . . . . . . . . . . (d)        |
_ SCZP101  20|     6.  View processor definition  . . . . . . . (v)        |
_ SCZP201  20|     7.  View related CTC connections . . . . . . (k)        |
_ SCZP301  28|     8.  Work with PCIe functions . . . . . . . . (f)        |
/ SCZP401  28|     9.  Work with partitions . . . . . . (SMP) (p)          |
************ |     10. Work with attached channel paths (SMP) (s)          |
             |     11. Work with attached devices . . . (SMP) (u)          |
             |     12. Copy to channel subsystem  . . . (SMP) (y)          |
             |     13. Work with channel subsystems . . (XMP) (p,s)        |
             |                                                             |
             | * = requires TSA I/O Operations                             |
             |                                                             |
             +-------------------------------------------------------------+
```

Figure 6-21 Define, Modify, or View Configuration Data

The PCIe Function List panel is shown in Figure 6-22 on page 71.

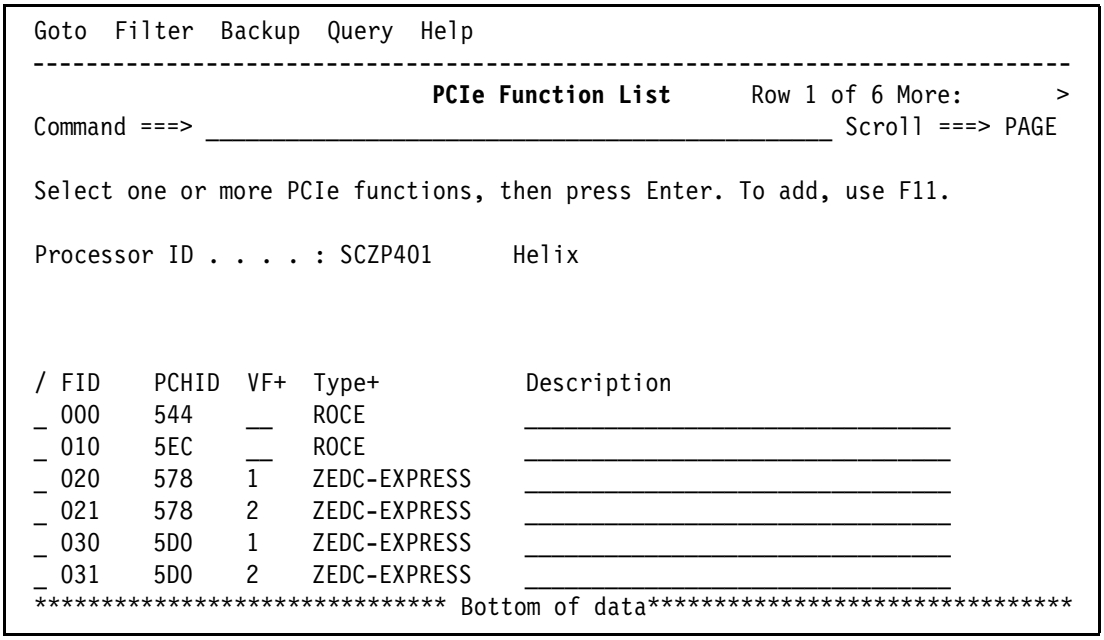


Figure 6-22 PCIe Function List panel

If you use F11 to add a PCIe function, the Add PCIe Function panel shown in Figure 6-23 is displayed.

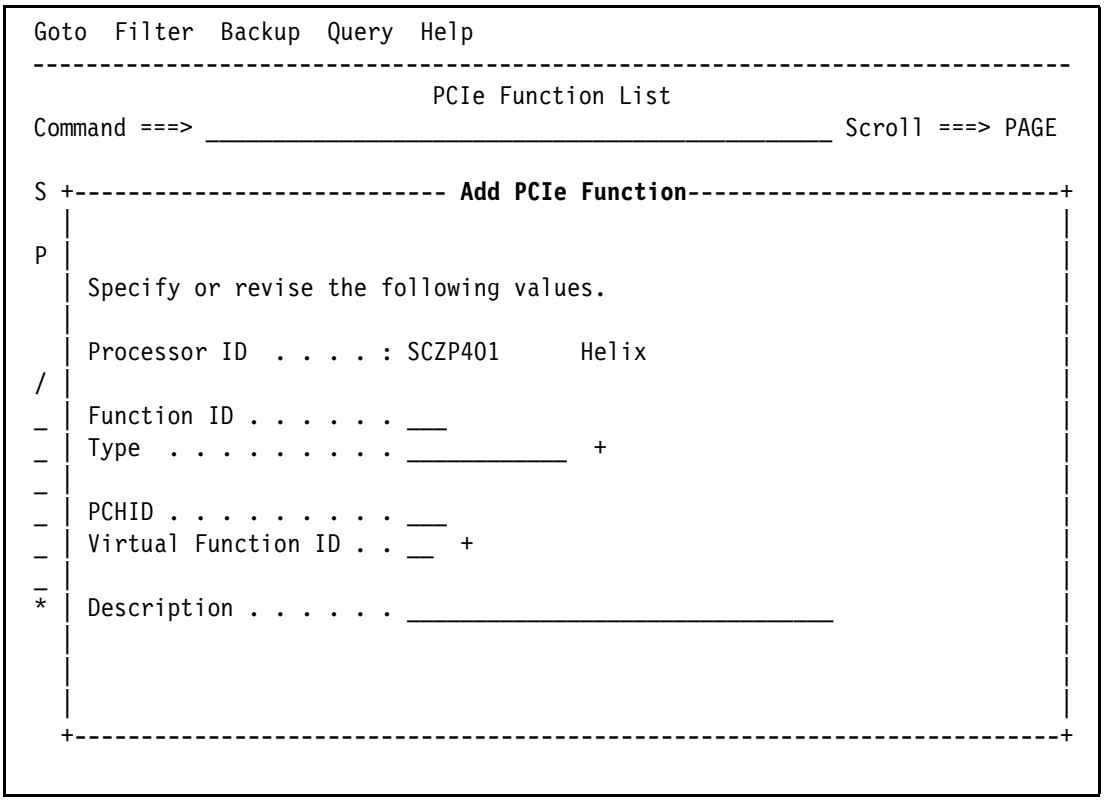


Figure 6-23 Add PCIe function panel

A PCIe function is defined uniquely, for each processor, by a function ID (FID), which can be specified in two ways:

- A hexadecimal value (three digits) which specifies the PCIe function. The valid range is from 00 - FF.
- The FID, which corresponds to a PCHID/VF definition, and must be unique in the configuration.

The following values are supported for the adapter type:

- ROCE
- zEDC-EXPRESS

The PCHID value follows the same rules as the PCHID value of CHPID. The PCHID values can be overgenned by specifying an asterisk (*) as the value. For the supported features, PCHID values are assigned based on the feature cards slot location in the PCIe drawer.

Some hardware features support a virtual function ID. This allows multiple functions to be assigned to the same PCHID if each of the functions is defined with a unique virtual function number.

Virtual function IDs are defined according to the following rules:

- A one-digit or two-digit decimal number from 1 - *n*. The valid ending number (*n*) depends on the specified function type.
- A virtual function ID must not be duplicated on the same PCHID specification.
- The zEDC supports a virtual function ID in the range of 1 - 15.

After the function ID, adapter type, PCHID, and virtual function ID are entered, card-specific panels are displayed. For example, the ROCE adapter type supports a Physical Network ID. An example of the Add/Modify Physical Network ID panel is shown in Figure 6-24.

```
 Goto  Filter  Backup  Query  Help
 - +--------------------- Add/Modify Physical Network ID----------------------+
   | |
 C |                                                                          |
   | If the PCHID is associated to one or more physical networks, specify     |
 S | each physical network ID corresponding to each applicable physical port. |
   |                                                                          |
 P | Physical network ID 1  . . ITSONET1________                              |
   | Physical network ID 2  . . ________________                              |
   | Physical network ID 3  . . ________________                              |
   | Physical network ID 4  . . ________________                              |
 / |                                                                          |
 _ |                                                                          |
 _ |                                                                          |
 _ |                                                                          |
 _ |                                                                          |
 _ +--------------------------------------------------------------------------+
 _ 031    5D0    2    ZEDC-EXPRESS    ____________________________________
 ****************************** Bottom of data ********************************
```

Figure 6-24 Add/Modify Physical Network ID

You can use the Add/Modify Physical Network IDs panel to specify up to four physical network IDs (PNET IDs) for corresponding physical adapter ports. The first PNET ID applies to the first physical port of the adapter, the second PNET ID to the second physical port, and so on. The following definition rules apply:

- Up to four physical network IDs can be specified. A physical network ID consists of an alphanumeric character name of up to 16 characters.
- PCIe functions with the same PCHID value defined must specify the same PNET IDs. Therefore, a change of the PNET IDs for a PCIe function results in an update of all functions with the same PCHID defined (except overgenned PCHID specified with *).

A PCIe function can be assigned to only one LPAR through the Define Access List dialog. But the PCIe function can be reassigned to another LPAR that was assigned to the PCIe function's candidate list. If PCIe function is defined with an empty access list, its candidate list must not be empty.

The following notes on PCIe functions also apply:

- The Repeat (Copy) processor configuration task also copies PCIe functions.
- The Repeat (Copy) partition task within the same processor also copies any PCIe functions.
- Copy CSS, copy partition to another processor and transfer partition does not copy PCIe functions to the new target.

The definition of a PCIe function can also be accomplished by using HCM.

6.8.2 FUNCTION IOCP Statement

The `FUNCTION` statement has been added to the Input/Output Configuration Program (IOCP) as a supported configuration statement. See *System z Input/Output Program User's Guide for ICP IOCP*, SB107037 for additional information.

Defining PCIe functions cause the generation of `FUNCTION` statements when creating the output for IOCP. Example 6-5 shows sample IOCP statements.

Example 6-5 FUNCTION IOCP statements

```
FUNCTION FID=000,PCHID=544,PNETID=ITSOPNET1,PART=((A15),(=))
FUNCTION FID=010,PCHID=5EC,PNETID=ITSOPNET1,PART=((A03),(=))
FUNCTION FID=020,VF=1,PCHID=578,PART=((A15),(=))
FUNCTION FID=021,VF=2,PCHID=578,PART=((A15),(=))
FUNCTION FID=030,VF=1,PCHID=5D0,PART=((A15),(=))
FUNCTION FID=031,VF=2,PCHID=5D0,PART=((A15),(=))
```

6.8.3 Physical Network ID keyword added for OSD and IQD CHPID types

Physical network IDs are used for network cards (10 Gb RoCE, OSD, IQD) to establish affinity with a particular physical network. When network connections require two interfaces to be coordinated, as with 10 Gb RoCE and an associated OSD, the physical network ID should match.

A Physical Network ID can now be specified for OSD and IQD CHIPD types.

For OSD type, up to four Physical Network IDs can be specified during CHPID add or change dialogs. The Add/Modify Physical Network IDs panel for a CHPID type OSD is shown in Figure 6-25 on page 74.

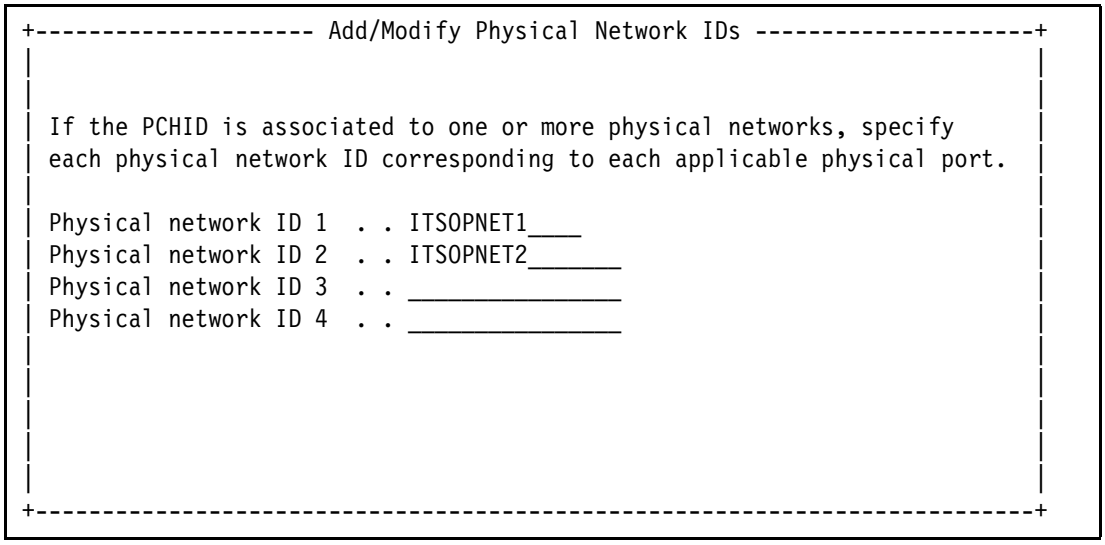


Figure 6-25 Physical Network IDs for the OSD CHPID type

For IQD type, one physical network ID can be specified during CHPID add or change dialogs. The Physical Network ID field is displayed with the Specify IQD Channel Parameters panel, as shown in Figure 6-26.

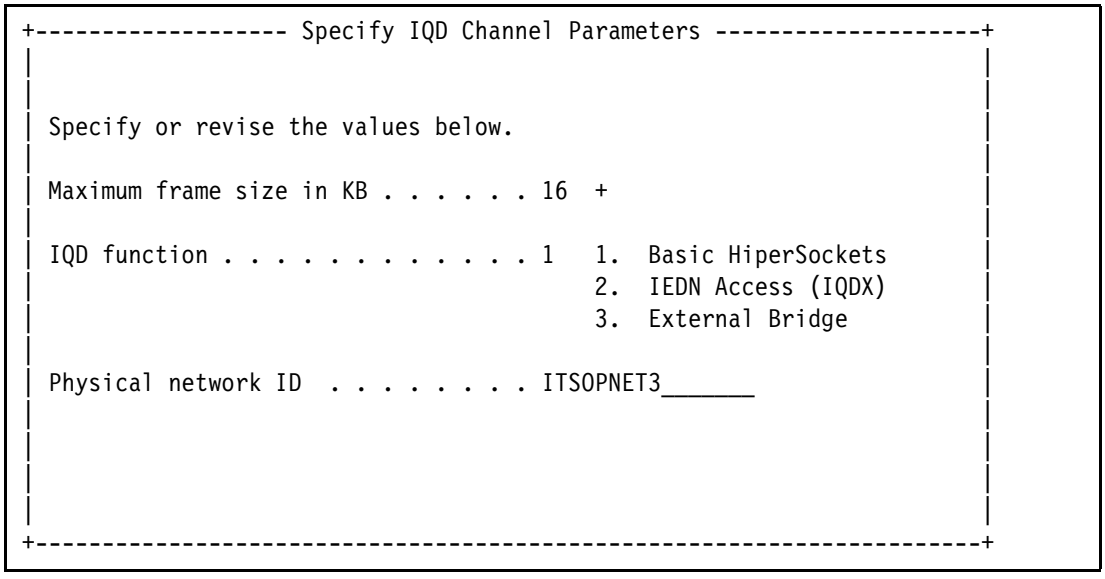


Figure 6-26 Physical network ID field for the IQD CHPID type

The physical network ID that is defined for IQD and OSD CHPIDS are represented in the IOCP configuration as a PNETID keyword, which is part of the CHPID statement.

6.8.4 PCIe Function Summary Report

The PCIe Function Summary Report displays the partitions in the access and candidate that are entitled to access the available PCIe functions. The report is generated as part of the Channel Subsystem Summary Reports. Figure 6-27 on page 75 shows an example.

```
                                    PCIE FUNCTION SUMMARY REPORT                TIME: 13:18 DATE:  2013-07-24  PAGE A-
 PROCESSOR ID  SCZP401  TYPE  2827     MODEL  H43      CONFIGURATION MODE: LPAR
                                              PARTITION NUMBERS
                          CSS0----------- CSS1----------- CSS2----------- CSS3-----------
FID VF PCHID TYPE         123456789ABCDEF 123456789ABCDEF 123456789ABCDEF 123456789ABCDEF   DESCRIPTION
___ __ _____ ___________ _______________ _______________ _______________ _______________   ____________________________
000    544   ROCE        --------------- ----A---------- --------------- ---------------
010    5EC   ROCE        --A------------ --------------- --------------- ---------------
020  1 578   ZEDC-EXPRESS --------------- ----A---------- --------------- ---------------
021  2 578   ZEDC-EXPRESS --------------- ----A---------- --------------- ---------------
030  1 5D0   ZEDC-EXPRESS --------------- ----A---------- --------------- ---------------
031  2 5D0   ZEDC-EXPRESS --------------- ----A---------- --------------- ---------------


    CSS0           CSS1           CSS2           CSS3
    PARTITION      PARTITION      PARTITION      PARTITION
    NUM  NAME      NUM  NAME      NUM  NAME      NUM  NAME
    ___  ________  ___  ________  ___  ________  ___  ________
     1   A01        1   A11        1   A21        1   A31
     2   A02        2   A12        2   A22        2   A32
     3   A03        3   A13        3   A23        3   A33
     4   A04        4   A14        4   A24        4   A34
     5   A05        5   A15        5   A25        5   *
     6   A06        6   A16        6   A26        6   *
     7   A07        7   A17        7   A27        7   *
     8   A08        8   A18        8   A28        8   *
     9   A09        9   A19        9   A29        9   *
     A   A0A        A   A1A        A   A2A        A   *
     B   A0B        B   A1B        B   A2B        B   *
     C   A0C        C   A1C        C   A2C        C   *
     D   A0D        D   A1D        D   A2D        D   A3D
     E   A0E        E   A1E        E   A2E        E   A3E
     F   A0F        F   A1F        F   A2F        F   A3F


    LEGEND FOR PARTITION NUMBERS FIELD:
    ___________________________________
     A     - PARTITION IS IN FUNCTION ACCESS LIST
     C     - PARTITION IS IN FUNCTION CANDIDATE LIST ONLY
     -     - PARTITION IS NOT IN FUNCTION ACCESS OR CANDIDATE LIST
```

Figure 6-27 PCIe Function Summary Report

6.8.5 PCIe Function Compare Report

An added report, the PCIe Function Compare Report, shows the changes in the function and virtual numbers and the attributes of PCIe functions between two IODFs.

The PCIe Function Compare Report can be generated when comparing IODFs. A sample report is shown in Figure 6-28.

```
 PCIe Function Compare Report                  TIME: 10:28  DATE: 2013-07-25  PAGE A -    1

              New IODF name:  SYS6.IODF15                          Old IODF name:  SYS6.IODF14


|------------------------------|----------------------|----------------------|-------------------------------------------------|
| PROC       FID               | New IODF             | Old IODF             | Description                                     |
|------------------------------|----------------------|----------------------|-------------------------------------------------|
| SCZP401    010               | Actual Data          | Old Data             |                                                 |
|                              |----------------------|----------------------|-------------------------------------------------|
|                              |  >> A03              |  >> new added        | Partition in Access List                        |
|                              |  >> old deleted      |  >> A15              | Partition in Access List                        |
|------------------------------|----------------------|----------------------|-------------------------------------------------|
```

Figure 6-28 PCIe Function Compare Report

6.9 PCHID Summary Report

The PCHID Summary Report is created as part of the CSS Summary Report. It lists all defined channel paths and PCIe functions, grouped by their defined PCHID values and HCA adapter IDs and port IDs, respectively.

6.9.1 Invocation and use

The PCHID Summary Report is generated as part of the CSS Summary Report. A truncated example is shown in Figure 6-29.

```
 PCHID SUMMARY REPORT                          TIME: 13:18 DATE:  2013-07-24  PAGE
 PROCESSOR ID  SCZP401  TYPE  2827     MODEL  H43      CONFIGURATION MODE: LPAR
             CHPID
PCHID  VF    FID   TYPE           SWITCH  CSS Numbers  PNET-1           PNET-2           PNET-3           PNET-4
_____  __  _____  ____________  ______  _____________ ________________ ________________ ________________ ________________
 594         59   FC                 61  |0|1|2|3|
 595         72   FC                     |0|1|2|3|
 598         5B   FC                 61  |0|1|2|3|
 599         6A   FC                 62  | |1|2| |
 59C         7E   FCP                    | |1| | |
 59D         62   FC                 62  |0|1|2|3|
 5A0         40   FC                 61  |0|1|2|3|
 5A1         78   FC                     |0|1|2|3|
 5A4         51   FC                 61  |0|1|2|3|
 5A5         70   FC                 61  |0|1|2|3|
 5A8         53   FC                 61  |0|1|2|3|
 5A9         68   FC                 61  | |1|2| |
 5B0         7C   FCP                    | |1| | |
 5B1         60   FC                 61  |0|1|2|3|
 5B4         0C   OSD                    |0|1|2|3|
 5B8         03   OSD                    | |1| | |
 5C4         12   OSD                    |0|1|2| |
 5C8         0D   OSC                    |0|1|2|3|
 5CC         41   FC                 62  |0|1|2|3|
 5CD         79   FC                     |0|1|2|3|
 5D0    1    030  ZEDC-EXPRESS           | |1| | |
 5D0    2    031  ZEDC-EXPRESS           | |1| | |
 5D4         55   FC                 62  |0|1|2|3|
 5D5         71   FC                 61  |0|1|2|3|
 5D8         57   FC                 62  |0|1|2|3|
 5D9         69   FC                 61  | |1|2| |
 5DC         7D   FCP                    | |1| | |
 5DD         61   FC                 61  |0|1|2|3|
 5E0         43   FC                 62  |0|1|2|3|
 5E1         7B   FCP                    |0| |2| |
 5E4         5D   FC                 62  |0|1|2|3|
 5E5         73   FC                     |0|1|2|3|
 5E8         5F   FC                 62  |0|1|2|3|
 5E9         6B   FC                 62  | |1|2| |
 5EC        010   ROCE                   |0| | | |  ITSOPNET1
 5F0         7F   FCP                    | |1| | |
 5F1         63   FC                 62  |0|1|2|3|
 5F4         0B   OSM                    | |1| | |
 5F8         13   OSD                    |0|1|2| |
09/1         80   CIB                    |0| | | |
09/2         A0   CIB                    | |1| | |
0A/1         81   CIB                    |0| | | |
0A/2         A1   CIB                    |0| |2| |
0A/2         B1   CIB                    | |1| | |
0A/2         B2   CIB                    |0| | | |
0B/1         82   CIB                    |0| | | |
0B/2         A2   CIB                    | |1| | |
```

Figure 6-29 PCHID summary report example

6.10 OFFLINE parameter in the MVS Device Detail Report

The MVS Device Detail Report previously showed the **OFFLINE** parameter for each device, even if it could not be specified by the user (for example a parallel access volume alias device). Now, the MVS Detail Report shows only the **OFFLINE** parameter for devices where the OFFLINE parameter can be set by the user.

6.10.1 Invocation and use

By default, the MVS Device Detail report now shows the OFFLINE parameter for devices where the parameter can be set by the user. A few lines from an MVS Device Detail Report are shown in Figure 6-30, where the OFFLINE parameter is not shown for parallel access volume alias devices.

```
                                      MVS DEVICE DETAIL REPORT
OPERATING SYSTEM CONFIGURATION ID:  LO6RMVS1
--- DEVICE ---      DEVICE
 NUMBER,RANGE    TYPE - MODEL    SS    PARAMETER
_____________   _____________   __    ____________________________________
   A500,92       3390A            1    WLMPAV=YES
-------------------------------------------------------------------------------
   A518,29       3390B                 OFFLINE=NO,DYNAMIC=YES,LOCANY=NO,
                                       WLMPAV=YES
-------------------------------------------------------------------------------
```

Figure 6-30 MVS Device Detail Report

6.11 HCD CF Channel Path Connectivity List

Previously, the CF Channel Path Connectivity List did not contain the PCHID or HCA and the port ID information for the CF CHPIDs. That required looking up the information in the Channel Path List.

The CF Channel Path Connectivity List is extended with the PCHID or HCA ID and the port number value for both the source and target CHPIDs of a CF connection.

6.11.1 Invocation and use

By default, the CF Channel Path Connectivity List displays the PCHID or HCD ID and the port number value for both the source and target CHPIDs of a CF connection. Figure 6-31 on page 78 shows an example.

```
CBDPCFFO             CF Channel Path Connectivity List             Row 1 of 22
Command ===> _____________________________________________ Scroll ===> PAGE

Select one or more channel paths, then press Enter.

Source processor ID . . . . . : SCZP401   Helix
Source channel subsystem ID . : 0
Source partition name . . . . : *

  ----------Source----------   -----------Destination-----------   -CU-  -#-
/ CHP PCHID CF Type Mode Occ   Proc.CSSID CHP PCHID CF Type Mode   Type  Dev
_ 80  09/1  N  CIB  SHR  N     SCZP201.0  90  09/1  Y  CIB  SHR    CFP   7
_ 81  0A/1  N  CIB  SHR  N     SCZP401.1  A0  09/2  Y  CIB  SHR    CFP   7
_ 82  0B/1  N  CIB  SHR  N     SCZP401.1  A2  0B/2  Y  CIB  SHR    CFP   7
_ 83  0C/1  N  CIB  SHR  N     SCZP401.1  A3  0C/2  Y  CIB  SHR    CFP   7
_ 84  19/1  N  CIB  SHR  N     SCZP201.0  98  0B/1  Y  CIB  SHR    CFP   7
_ 85  1A/1  N  CIB  SHR  N     SCZP401.1  A4  19/2  Y  CIB  SHR    CFP   7
_ 86  1B/1  N  CIB  SHR  N     SCZP401.1  A6  1B/2  Y  CIB  SHR    CFP   7
_ 87  1C/1  N  CIB  SHR  N     SCZP401.1  A7  1C/2  Y  CIB  SHR    CFP   7
_ A1  0A/2  Y  CIB  SPAN N     SCZP301.3  81  18/2  Y  CIB  SHR    CFP   7
_ A5  1A/2  Y  CIB  SPAN N     SCZP301.3  86  08/2  Y  CIB  SHR    CFP   7
_ B2  0A/2  Y  CIB  SHR  N     SCZP301.1  83  18/2  Y  CIB  SHR    CFP   7
_ B6  1A/2  Y  CIB  SHR  N     SCZP301.1  88  08/2  Y  CIB  SHR    CFP   7
```

Figure 6-31 CF Channel Path Connectivity List

6.12 HCD hardware support

HCD supports the IBM zEnterprise EC12 (zEC12) and zEnterprise BC12 (zBC12) processor types: 2827-H20, -H43, -H66, -H89, -HA1, and 2828-H06, -H13.

6.13 HCM port number status line

Before this change, the HCM status line had the information about the host name and Parallel Sysplex name of the connected z/OS system but not the IP port number. The HCM status line now shows the port number of the connected dispatcher HCM.

6.13.1 Invocation and use

The port information is automatically displayed in an HCM session. You can easily check the host name and the port number on the system to which the HCM dispatcher is connected. Figure 6-32 on page 79 shows an example of the port number display.

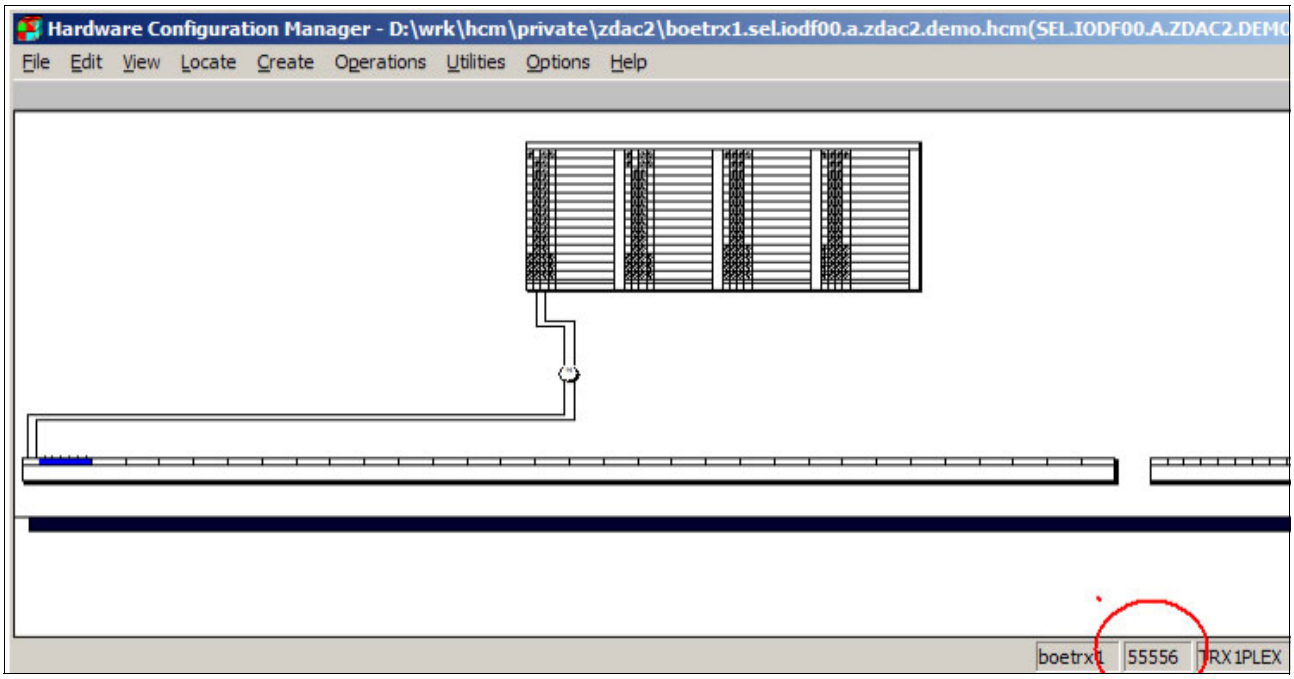


Figure 6-32 IP port number shown in HCM status line

6.14 Connecting CHPIDs to control units in HCM

In HCM, a performance degradation was observed when CHPIDs were connected to multiple control units via switches. A modification in z/OS 2.1 removes redundant processing in this task. The connection time of CHPIDs to control units is now similar with one or multiple control units.

6.15 HCM explanatory message on load production IODF

When a production IODF is loaded, HCM builds a *work* configuration file (with the `.hcm` extension), rather than a *production* configuration file with the `.hcr` extension). Also, if a production file is built, the configuration file opened in HCM is still a work configuration file.

This confused HCM users. Now, when loading a production IODF or running the **`Build`** production command, a notice says that the opened HCM file is a work configuration file so that the user is aware that HCM generates a work configuration file.

6.15.1 Invocation and use

When loading a production IODF, the message shown in Figure 6-33 displays.

Figure 6-33 Notice that HCM saves the file as a work configuration file

When running the **`File/Build production`** command on a work configuration file, the message shown in Figure 6-34 displays.

Figure 6-34 Notice that configuration opened in HCM is still a work configuration file

6.16 Documentation and reference

For more information, see the following publications and web pages:

- *z/OS Hardware Configuration Definition User's Guide, Version 2, Release1,* SC34-2669
  `https://ibm.biz/BdRWn8`
- z/OS and z/VM Hardware Configuration Manager User's Guide, SC34-2670
  `https://ibm.biz/BdFcUQ`
- *z/OS V2R1 HCD Planning*, GA32-0907
  `https://ibm.biz/BdFcUT`
- IBM Hardware Configuration Manager web page
  `https://ibm.biz/BdFBi2`
- IBM Hardware Configuration web page
  `http://www.ibm.com/systems/z/os/zos/features/hcm/`

7

Base Control Program Internal Interface (BCPii)

This chapter describes the functions and features of the Base Control Program Internal Interface (BCPii), which was introduced with IBM z/OS Version 2.1.

Within z/OS, authorized applications can query, change, and perform operational procedures against the installed IBM System z hardware base through a set of application program interfaces (APIs). BCPii allows authorized z/OS applications to have Hardware Management Console (HMC)-like control over systems in the HMC network. For example, by using BCPii, an authorized application can perform the following tasks:

- Monitor status or capacity changes
- Obtain configuration data that is related to the central processor complex (CPC) or image
- Re-IPL an image
- Change temporary capacity
- Query and update logical partition (LPAR) settings
- Set activation profiles

BCPii communication to the System z Support Element (SE) and the HMC is within the base operating system. Therefore, BCPii does not require communication on an IP network (intranet) for connectivity. This provides complete isolation of your System z hardware communication from any other network traffic within the intranet.

This chapter covers these topics:

- 7.1, “BCPii support for REXX” on page 82
- 7.2, “Setting up a BCPii REXX environment” on page 85
- 7.3, “BCPii performance improvement for HWILIST and HWIQUERY” on page 94
- 7.4, “Documentation and reference” on page 94

© Copyright IBM Corp. 2014. All rights reserved.

7.1 BCPii support for REXX

In previous releases, calls using the BCPii APIs can be made from the C or the Assembler programming languages. In addition to these existing programming languages, starting with z/OS 2.1, BCPii includes Restructured Extended Executor (REXX) language support for System REXX, Time Sharing Option/Extensions (TSO/E) REXX, and REXX environments from independent software vendors (ISVs).

The supported BCPii APIs for the REXX programming language are listed by environment in Table 7-1.

Table 7-1 z/OS BCPii APIs supported for the REXX programming language

Service	System REXX	TSO REXX	ISV REXX
HWICONN	X	X	X
HWIDISC	X	X	X
HWILIST	X	X	X
HWIQUERY	X	X	X
HWISET	X	X	X
HWIEVENT	X		
HWICMD	X		

REXX support provides the following benefits, among others:

- Uses the same SAF authorization requirements as existing BCPii applications

  No changes to the existing security setup for BCPii are required for use of the BCPii REXX APIs.

- Compatible with different REXX environments

  BCPii REXX programs are compatible with the different REXX environments, for the common services that are supported by BCPii in the different environments.

- Working with other C or assembly language BCPii applications from System REXX

  System REXX BCPii applications can work with other C or assembly language BCPii applications. The Connect Token includes connection information for CPC, image, capacity record, different types of activation profiles, or user-defined image groups. It can be passed to and from the System REXX exec and the other compiled BCPii applications. This is not supported in TSO or independent software vendor (ISV) REXX environments.

- Full support of BCPii API suite for System REXX

  Command and Event services require a non-REXX event exit and a program to wait on an ECB based on event activity. TSO and ISV REXX environments do not support the HWIEVENT and HWICMD service.

- Built-in return code (RC)

  Built-in RC return indicates whether BCPii processed the host command. The BCPii return code indicates whether the host command ended successfully or not. If the RC is zero, it is a successful completion, and BCPii checks the return code. If it is not zero, see the descriptions in *MVS Programming: Callable Services for High-Level Languages*, SA23-1377 (cited in 7.4, “Documentation and reference” on page 94).

- Address space or task affinity connections

  For System REXX, when the AXREXX macro invoker's address space terminates, BCPii disconnects all connections.

  For TSO and ISV REXX environments, BCPii implicitly disconnects all connections when the exec completes.

- TSO=YES and TSO=NO for environments that are supported for SYSTEM REXX
  - TSO=YES allows REXX to include (by using the **execio** command) and interpret the IBM-supplied REXX include file (HWICIREX), which defines constants.
  - TSO=NO requires the IBM-supplied include file to be copied to the exec.
  - TIMELIMIT keyword can throttle BCPii execution time for SYSTEM REXX

  The `TIMELIMIT` and `TIMEINT` parameters can be used to control a BCPii system exec's processing time. The default value of 30 seconds can be adjusted as required.

7.1.1 Invocation and use

The sections that follow describe methods of running REXX programs by using the BCPii API.

Running from System REXX

By using System REXX, you can write sophisticated BCPii applications by using REXX and other programming languages as part of a single application. There are two ways to run BCPii a SYSTEM REXX exec:

- Code an assembly language program to start the **AXREXX** macro.

  Specify the name of BCPii REXX exec and any of the **AXREXX** options.

- Use the BCPii **HWIREXX** program.

  Use the IBM-supplied helper program, **HWIREXX**, in SYS1.LINKLIB, which calls AXREXX to run the user REXX executable file. Authorized users can use **HWIREXX** to start their system REXX execs. Simple REXX can be invoked directly without the need to code the **AXREXX** assembly language macro. SAMPLIB JCL member **HWIXMRJL** provides a list of parameters **HWIREXX** takes as input. It also supports a subset of **AXREXX** options.

The sample JCL to start HWIREXX is included in SYS1.SAMPLIB(HWIXMRJL) as is shown in Example 7-1.

Example 7-1 HWIXMRJL

```
//STEP1   EXEC PGM=HWIREXX,REGION=1M,
//   PARM=('NAME=myexec',
//             'DSN=my.dsn.output',
//             'TSO=N',
//             'SYNC=Y',
//             'TIMELIM=Y',
//             'TIME=40')
//*
//STEPLIB  DD  DSN=SYS1.LINKLIB,DISP=SHR
//SYSPRINT DD  SYSOUT=*
```

The parameters can be specified according to the following guide:

NAME=<xxx>	Where xxx is a 1 to 8-character exec name to run.
TSO=<Y/N>	Y = TSO host command environment
	N = Standard MVS host environment
	Default = N
SYNC=<Y/N>	Y = request is synchronous
	N = request is asynchronous
	Default = Y
TIMELIM=<Y/N>	Y = time limit is applied
	N = no time limit is applied
	Default = Y
TIME=<xxx>	Where xxx is a number from 1 - 21474536 that represents the number of seconds to allow the exec to run.
	Default = 30
DSN=<xxx.xxx.xxx>	Where xxx.xxx.xxx is a 1 - 44 character PDS data set name where the REXX exec output is directed.
	Default = NO_REXXOUTDSN
	If the data set exists, it must be a partitioned data set. Otherwise, it is allocated by the System REXX services.

Note: When using **HWIREXX**, the data set where the **REXX** exec is to be run must be specified in the **AXRxx** member in **SYS1.PARMLIB**, and users of this program must have the authority to run programs that are in **SYS1.LINKLIB**.

The `TIMELIM=Y` argument with `TIME=`*nn* or `TIMELIM=`*N* can be used to prevent BCPii timeouts. `TSO=Y` can be used to enable more TSO functions, such as **execio**. Otherwise, no I/O capability is provided.

The partitioned data set for REXX exec output must be preallocated. The output data set member name has the same name as the called REXX exec.

Running from TSO/E REXX

TSO/E REXX supports all BCPii APIs except **HWIEVENT** and **HWICMD**. All connections created by the TSO/E REXX exec are automatically cleaned by BCPii when the exec completes. Therefore, connections cannot be shared with other BCPii applications or REXX execs.

Running from ISV REXX

All BCPii APIs except **HWIEVENT** and **HWICMD** are supported. All connections that are created by the ISV REXX exec are automatically cleaned by BCPii when the exec completes. Therefore, connections cannot be shared with other BCPii applications or REXX execs. Also, ISV REXX must be started from an authorized address space.

7.2 Setting up a BCPii REXX environment

To start REXX for BCPii, you need to set up BCPii. See “BCPii setup and installation” in *MVS Programming: Callable Services for High-Level Languages* (SA23-1377) in the IBM Knowledge Center. See 7.4, “Documentation and reference” on page 94 for a link.

Also follow these instructions:

- For TSO/E REXX for BCPii, allow BCPii APIs to run under TSO/E REXX by modifying the configuration of the TSO/E Commands and Programs PARMLIB member (**IKJTSOxx**).

  Add the BCPII REXX TSO command processor to the list of APF-authorized programs that can be called through the TSO Service Facility (**AUTHTSF**). Add this syntax to add BCPii to this list:

  ```
  AUTHTSF NAMES(HWIC1TRX)
  ```

  After this change is activated, the TSO/E user still needs SAF authorization to the correct BCPii profiles to perform the BCPii operation.

- For ISV REXX, to get the **bcpii** host command environment, the REXX exec must issue the following statement:

  ```
  rc = hwihost("ON")
  ```

  Also, ISV REXX must be started from an authorized address space, and the user must have SAF authorization to the correct BCPii profiles to perform the BCPii operation.

7.2.1 BCPii services that are available from REXX

The following BCPii services are available from REXX execs:

HWICONN Establish a logical connection between the application and a CPC, a CPC image (or LPAR), a capacity record, different types of activation profiles, or a user-defined image group.

HWIDISC Release the logical connection between the application and the identified CPC, image, capacity record, different types of activation profiles, or user-defined image groups.

HWILIST Retrieve Hardware Management Console (HMC) and BCPii configuration-related information. Depending on which information is requested, the data that is returned by this service can be used on subsequent BCPii service calls to take other actions.

HWIQUERY Retrieve information about objects that are managed by the Support Element (SE) or an HMC connected to CPCs, CPC images (LPARs), capacity records, different types of activation profiles, or user-defined image groups.

HWISET Change or set data for HMC-managed objects that are associated with CPCs, CPC images (LPARs), or activation profiles.

HWIEVENT Register an application and its connection to receive these notifications:

- One or more hardware or software events that are occurring on the connected CPC or image
- Communication errors between BCPii and the connected CPC or image

Or, delete the registration for one or more previously registered events.

HWICMD Issue a command for an HMC-managed object that is associated with CPCs and CPC images (LPARs). User-defined image groups can also be used to target multiple images with a single command.

Note: **HWIEVENT** and **HWICMD** can be started only from System REXX.

When coding REXX for BCPii, constants are used for any request. For example, the **HWI_LIST_CPCS** variable is used in the **HWILIST** command to request a list of CPCs that can be accessed, and **HWI_CPC** is used in **the HWICONN** command to request a connection to a target CPC that the application needs to communicate with.

Each variable and its constant value must be defined in the REXX exec. Constant values are described in *MVS Programming: Callable Services for High-Level Languages* (SA23-1377).

To define constants, define the required constants in the code explicitly, or read **SYS1.MACLIB** members **HWICIREX** and **HWIC2REX** into the REXX exec when it is running. A simple programming example that reads the values into the REXX exec by using the **EXECIO** function is provided in the **SYS1.SAMPLIB** member **HWIXMRS1**.

Note: When reading **SYS1.MACLIB** member **HWICIREX** and **HWIC2REX** into a REXX exec, you must specify `TSO=YES` to let the REXX exec run in a TSO host command environment that has I/O services enabled.

7.2.2 Sample codes

See the following code examples:

- REXX exec by using **the HWILIST** command for listing available CPCs in a network. (Example 7-2)
- REXX exec by using **the HWICONN** command for connecting to a specified CPC. (Example 7-3 on page 88)
- REXX exec by using **the HWIQUERY** command for querying information from a specified CPC. (Example 7-4 on page 89)
- REXX exec by using the **HWILIST**, **HWICONN**, and **HWIQUERY** commands for listing available CPCs in a network, connect to CPCs listed, and query information from them. (Example 7-5 on page 90)
- **HWIREXX** JCL starting REXX member that is named ‘HWILCQ’. (Example 7-6 on page 93)

For more samples, see **SYS1.SAMPLIB** members **HWIXMRJL** and **HWIXMRS1** and *MVS Programming: Callable Services for High-Level Languages* (SA23-1377) in the IBM Knowledge Center. See 7.4, “Documentation and reference” on page 94 for a link.

Example 7-2 Sample HWILIST command

```
/*REXX*/
rc = hwihost("ON")

 ListCPCs:

  /* ------------------------------------------------------------ */
  /* Report HWILIST CPCs is starting.                             */
  /* ------------------------------------------------------------ */
  say ' =>> List CPCs starts:'
```

```
/* ------------------------------------------------------------- */
/* Set HWILIST parameters.                                       */
/* ------------------------------------------------------------- */
HWI_LIST_CPCS = 1
ListType = HWI_LIST_CPCS

/* ------------------------------------------------------------- */
/* Initialize AnswerArea.                                        */
/* ------------------------------------------------------------- */
CPCList. = 'CPC names will be listed here'

/* ------------------------------------------------------------- */
/* Call HWILIST.                                                 */
/* Note: The input connect token is ignored for a List CPCs      */
/*       request. The parameter is not initialized.              */
/* ------------------------------------------------------------- */
address bcpii "hwilist ReturnCode ConnectToken ListType CPCList. DiagArea."

/* ------------------------------------------------------------- */
/* Save value returned as REXX special variable, RC.             */
/* ------------------------------------------------------------- */
REXXHostRc = RC

/* ------------------------------------------------------------- */
/* Report HWILIST results.                                       */
/* ------------------------------------------------------------- */
say '        REXXHostRc  = ' d2x(REXXHostRc)
say '        HWILIST rc  = ' d2x(ReturnCode)
say

/* ------------------------------------------------------------- */
/* If HWILIST fails, report diagnostic information.              */
/* ------------------------------------------------------------- */
If REXXHostRC <>0 | ReturnCode <> 0 Then
  Do
    say '        DiagArea.Diag_Index    = '  DiagArea.Diag_Index
    say '        DiagArea.Diag_Key      = '  DiagArea.Diag_Key
    say '        DiagArea.Diag_Actual   = '  DiagArea.Diag_Actual
    say '        DiagArea.Diag_Expected = '  DiagArea.Diag_Expected
    say '        DiagArea.Diag_CommErr  = '  DiagArea.Diag_CommErr
    say '        DiagArea.Diag_Text     = '  DiagArea.Diag_Text
    say
  End

/* ------------------------------------------------------------- */
/* Report the number of CPCs returned.                           */
/* ------------------------------------------------------------- */
say '        Number of CPCs returned = ' CPCList.0
say

/* ------------------------------------------------------------- */
/* Write the list of CPCs returned.                              */
/* ------------------------------------------------------------- */
Do i = 1 to CPCList.0
```

```
    say '        CPC '|| i ' = ' CPCList.i
  End
  say

 return /* end ListCPCs */
```

Example 7-3 Sample HWICONN command

```
/*REXX*/
rc = hwihost("ON")
  /* ------------------------------------------------------------- */
  /* Report HWICONN is starting.                                   */
  /* ------------------------------------------------------------- */
  say ' =>> Image Connect starts:'

  /* ------------------------------------------------------------- */
  /* Set HWICONN parameters.                                       */
  /* ------------------------------------------------------------- */
  HWI_CPC = 1
  ConnectType = HWI_CPC
  ConnectTypeValue = 'USIBMSC.SCZP401  '

  /* ------------------------------------------------------------- */
  /* Call HWICONN.                                                 */
  /* ------------------------------------------------------------- */
  say '     Connect to image '|| ConnectTypeValue

  address bcpii "hwiconn ReturnCode InConnectToken OutConnectToken",
                "ConnectType ConnectTypeValue DiagArea."

  /* ------------------------------------------------------------- */
  /* Save value returned as REXX special variable, RC.             */
  /* ------------------------------------------------------------- */
  REXXHostRc = RC
  /* ------------------------------------------------------------- */
  /* Report HWICONN results.                                       */
  /* ------------------------------------------------------------- */
  say '        REXXHostRc  = ' d2x(REXXHostRc)
  say '        HWICONN rc  = ' d2x(ReturnCode)
  say

  lastrc = rc
  say 'error  rc='lastrc 'bcpii retcode='returncode
  /* ------------------------------------------------------------- */
  /* If HWICONN fails, report diagnostic information.              */
  /* ------------------------------------------------------------- */
  If REXXHostRC <>0 | ReturnCode <> 0 Then
    Do
      say '        DiagArea.Diag_Index    = '  DiagArea.Diag_Index
      say '        DiagArea.Diag_Key      = '  DiagArea.Diag_Key
      say '        DiagArea.Diag_Actual   = '  DiagArea.Diag_Actual
      say '        DiagArea.Diag_Expected = '  DiagArea.Diag_Expected
      say '        DiagArea.Diag_CommErr  = '  DiagArea.Diag_CommErr
      say '        DiagArea.Diag_Text     = '  DiagArea.Diag_Text
```

```
      say
    End
  Else
    ImageConnectToken = OutConnectToken

 Return /* end ConnectToImage */
```

Example 7-4 Sample HWIQUERY command

```
/*REXX*/
  say ' =>> Query CPC attributes starts:'

  /* ---------------------------------------------------------------- */
  /* Build QueryParm structure to query CPC attributes.  QueryParm   */
  /* index 0 contains the number of attributes to be queried.        */
  /* ---------------------------------------------------------------- */
  QueryParm.0 = 2
  HWI_MMODEL = 25
  HWI_SNAADDR = 24
  QueryParm.1.ATTRIBUTEIDENTIFIER = HWI_MMODEL
  QueryParm.2.ATTRIBUTEIDENTIFIER = HWI_SNAADDR

  /* ---------------------------------------------------------------- */
  /* Call HWIQUERY.                                                  */
  /* ---------------------------------------------------------------- */
  address bcpii "hwiquery ReturnCode ConnectToken QueryParm. DiagArea."

  /* ---------------------------------------------------------------- */
  /* Save value returned as REXX special variable, RC.               */
  /* ---------------------------------------------------------------- */
  REXXHostRc = RC

  /* ---------------------------------------------------------------- */
  /* Report HWIQUERY results.                                        */
  /* ---------------------------------------------------------------- */
  say '        REXXHostRc  = ' d2x(REXXHostRc)
  say '        HWIQUERY rc = ' d2x(ReturnCode)
  say

  /* ---------------------------------------------------------------- */
  /* If HWIQUERY fails, report diagnostic information.               */
  /* ---------------------------------------------------------------- */
  If REXXHostRC <>0 | ReturnCode <> 0 Then
    Do
      say '        DiagArea.Diag_Index    = '  DiagArea.Diag_Index
      say '        DiagArea.Diag_Key      = '  DiagArea.Diag_Key
      say '        DiagArea.Diag_Actual   = '  DiagArea.Diag_Actual
      say '        DiagArea.Diag_Expected = '  DiagArea.Diag_Expected
      say '        DiagArea.Diag_CommErr  = '  DiagArea.Diag_CommErr
      say '        DiagArea.Diag_Text     = '  DiagArea.Diag_Text
      say
    End

  /* ---------------------------------------------------------------- */
```

```
 /* Report the returned attribute values.                          */
 /* -------------------------------------------------------------- */
 say '        MModel      = ' QueryParm.1.ATTRIBUTEVALUE
 say '        SNAAddr     = ' QueryParm.2.ATTRIBUTEVALUE
 say

return /* end QueryCPCAttr */
```

Example 7-5 Sample for listing all the CPCs defined in HMC, and Query information to each CPCs

```
/*REXX*/
rc = hwihost("ON")              /*Needed when invoking from ISV REXX*/

 /* ============================================================== */
 /* ListCPCs                                                       */
 /*  Call HWILIST to list all CPCs in an HMC network to which you  */
 /*  have authority. Save the returned list in List_of_CPCs array. */
 /* ============================================================== */
 ListCPCs:

  /* -------------------------------------------------------------- */
  /* Report HWILIST CPCs is starting.                               */
  /* -------------------------------------------------------------- */
  say ' =>> List CPCs starts:'

  /* -------------------------------------------------------------- */
  /* Set HWILIST constants and parameters.                          */
  /* -------------------------------------------------------------- */
  HWI_LIST_CPCS = 1
  ListType = HWI_LIST_CPCS

  /* -------------------------------------------------------------- */
  /* Initialize AnswerArea.                                         */
  /* -------------------------------------------------------------- */
  CPCList. = 'CPC names will be listed here'

  /* -------------------------------------------------------------- */
  /* Call HWILIST.                                                  */
  /* Note: The input connect token is ignored for a List CPCs       */
  /*       request. The parameter is not initialized.               */
  /* -------------------------------------------------------------- */
  address bcpii "hwilist ReturnCode ConnectToken ListType CPCList. DiagArea."

  /* -------------------------------------------------------------- */
  /* Save value returned as REXX special variable, RC.              */
  /* -------------------------------------------------------------- */
  REXXHostRc = RC

  /* -------------------------------------------------------------- */
  /* Report HWILIST results.                                        */
  /* -------------------------------------------------------------- */
  say '        REXXHostRc  = ' d2x(REXXHostRc)
  say '        HWILIST rc  = ' d2x(ReturnCode)
  say
```

```
/* ------------------------------------------------------------- */
/* If HWILIST fails, report diagnostic information.              */
/* ------------------------------------------------------------- */
If REXXHostRC <>0 | ReturnCode <> 0 Then
  Do
    say '        DiagArea.Diag_Index    = '  DiagArea.Diag_Index
    say '        DiagArea.Diag_Key      = '  DiagArea.Diag_Key
    say '        DiagArea.Diag_Actual   = '  DiagArea.Diag_Actual
    say '        DiagArea.Diag_Expected = '  DiagArea.Diag_Expected
    say '        DiagArea.Diag_CommErr  = '  DiagArea.Diag_CommErr
    say '        DiagArea.Diag_Text     = '  DiagArea.Diag_Text
    say
  End

/* ------------------------------------------------------------- */
/* Report the number of CPCs returned.                           */
/* ------------------------------------------------------------- */
say '        Number of CPCs returned = ' CPCList.0
say

/* ------------------------------------------------------------- */
/* Write the list of CPCs returned.                              */
/* ------------------------------------------------------------- */
Do i = 1 to CPCList.0
  say '        CPC '|| i ' = ' CPCList.i
End
say

/* ------------------------------------------------------------- */
/* End of HWILIST. Report HWICONN is starting.                   */
/* ------------------------------------------------------------- */
Do i = 1 to CPCList.0

say ' =>> CPC Connect to ' CPCList.i ' starts:'

/* ------------------------------------------------------------- */
/* Set HWICONN constants and parameters.                         */
/* ------------------------------------------------------------- */
HWI_CPC = 1
ConnectType = HWI_CPC
ConnectTypeValue = CPCList.i '  '

/* ------------------------------------------------------------- */
/* Call HWICONN.                                                 */
/* ------------------------------------------------------------- */
say '     Connect to CPC '|| ConnectTypeValue

address bcpii "hwiconn ReturnCode InConnectToken OutConnectToken",
              "ConnectType ConnectTypeValue DiagArea."

/* ------------------------------------------------------------- */
/* Save value returned as REXX special variable, RC.             */
/* ------------------------------------------------------------- */
REXXHostRc = RC
```

```
/* ------------------------------------------------------------- */
/* Report HWICONN results.                                       */
/* ------------------------------------------------------------- */
say '        REXXHostRc  = ' d2x(REXXHostRc)
say '        HWICONN rc  = ' d2x(ReturnCode)
say

/* ------------------------------------------------------------- */
/* If HWICONN fails, report diagnostic information.              */
/* ------------------------------------------------------------- */
If REXXHostRC <>0 | ReturnCode <> 0 Then
  Do
    say '        DiagArea.Diag_Index    = '  DiagArea.Diag_Index
    say '        DiagArea.Diag_Key      = '  DiagArea.Diag_Key
    say '        DiagArea.Diag_Actual   = '  DiagArea.Diag_Actual
    say '        DiagArea.Diag_Expected = '  DiagArea.Diag_Expected
    say '        DiagArea.Diag_CommErr  = '  DiagArea.Diag_CommErr
    say '        DiagArea.Diag_Text     = '  DiagArea.Diag_Text
    say
  End
Else
  ImageConnectToken = OutConnectToken

/* ------------------------------------------------------------- */
/* End of HWICONN. Report HWIQUERY is starting.                  */
/* ------------------------------------------------------------- */
say ' =>> Query CPC attributes starts:'

/* ------------------------------------------------------------- */
/* Build QueryParm structure to query CPC attributes.  QueryParm */
/* index 0 contains the number of attributes to be queried.      */
/* ------------------------------------------------------------- */
HWI_SNAADDR = 24
HWI_MMODEL = 25
HWI_MTYPE = 26
HWI_MSERIAL = 27
HWI_NUMGPP = 52

QueryParm.0 = 5
QueryParm.1.ATTRIBUTEIDENTIFIER = HWI_MMODEL
QueryParm.2.ATTRIBUTEIDENTIFIER = HWI_SNAADDR
QueryParm.3.ATTRIBUTEIDENTIFIER = HWI_MTYPE
QueryParm.4.ATTRIBUTEIDENTIFIER = HWI_MSERIAL
QueryParm.5.ATTRIBUTEIDENTIFIER = HWI_NUMGPP

ConnectToken = OutConnectToken
/* ------------------------------------------------------------- */
/* Call HWIQUERY.                                                */
/* ------------------------------------------------------------- */
address bcpii "hwiquery ReturnCode ConnectToken QueryParm. DiagArea."

/* ------------------------------------------------------------- */
/* Save value returned as REXX special variable, RC.             */
/* ------------------------------------------------------------- */
REXXHostRc = RC
```

```
 /* ------------------------------------------------------------- */
 /* Report HWIQUERY results.                                      */
 /* ------------------------------------------------------------- */
 say '        REXXHostRc  = ' d2x(REXXHostRc)
 say '        HWIQUERY rc = ' d2x(ReturnCode)
 say

 /* ------------------------------------------------------------- */
 /* If HWIQUERY fails, report diagnostic information.             */
 /* ------------------------------------------------------------- */
 If REXXHostRC <>0 | ReturnCode <> 0 Then
   Do
     say '        DiagArea.Diag_Index    = '  DiagArea.Diag_Index
     say '        DiagArea.Diag_Key      = '  DiagArea.Diag_Key
     say '        DiagArea.Diag_Actual   = '  DiagArea.Diag_Actual
     say '        DiagArea.Diag_Expected = '  DiagArea.Diag_Expected
     say '        DiagArea.Diag_CommErr  = '  DiagArea.Diag_CommErr
     say '        DiagArea.Diag_Text     = '  DiagArea.Diag_Text
     say
   End

 /* ------------------------------------------------------------- */
 /* Report the returned attribute values.                         */
 /* ------------------------------------------------------------- */
 say '        MModel      = ' QueryParm.1.ATTRIBUTEVALUE
 say '        SNAAddr     = ' QueryParm.2.ATTRIBUTEVALUE
 say '        HWModel     = ' QueryParm.3.ATTRIBUTEVALUE
 say '        MSerial     = ' QueryParm.4.ATTRIBUTEVALUE
 say '        GCPNumbers  = ' QueryParm.5.ATTRIBUTEVALUE
 say

End
return
```

Example 7-6 HWIREXX JCL invoking REXX member named 'HWILCQ'

```
//HWIXMRJL JOB NOTIFY=????????,
// MSGLEVEL=1,MSGCLASS=H
//STEP1   EXEC PGM=HWIREXX,REGION=1M,
//   PARM=('NAME=HWILCQ',
//             'DSN=ZOSUPD2.LOG',
//             'TSO=Y',
//             'SYNC=Y',
//             'TIMELIM=Y',
//             'TIME=60')
//*
//STEPLIB  DD  DSN=SYS1.LINKLIB,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//
```

7.3 BCPii performance improvement for HWILIST and HWIQUERY

In previous releases, BCPii retrieval requests can be slow, especially when multiple attributes are retrieved. That is because the connection between z/OS and the SE that z/OS BCPii uses (internal proprietary interface) has both high latency and low bandwidth.

Several enhancements in z/OS 2.1 improve performance significantly for BCPii **HWILIST** and **HWIQUERY**.

7.3.1 Invocation and use

In previous releases, single simple query requests can average between 0.3 and 0.5 seconds on the wall clock due to various factors. For example, **HWILIST** ListImage requires n+1 **HwmcaGet** requests to the SE, where n = number of LPARs on the CPC and **HWIQUERY** that specifies six attributes requires 6 **HwmcaGet** requests to the SE. **HwmcaGet** is one of the IBM System z APIs. It is used to a query or issue a **Get** request for a specified object or object attribute. See 7.4, “Documentation and reference” on page 94 for the link to *System z Application Programming Interfaces* for more information.

Starting with z/OS 2.1, **HwmcaGetBulk** call, which is also a System z API, became available for BCPii. **HwmcaGetBulk** is used to retrieve or get data that is associated with a series of object attributes with a single request. It minimizes the number of requests required to retrieve large amounts of object or object attribute data. This call can be viewed as performing a series of **HwmcaGetNext** calls with a single request. **HwmcaGetNext** is a call that is used to retrieve or get the data associated with the object attribute that occurs next in the lexical sequence of objects. With **the HwmcaGetBulk** call, **HWILIST** ListImage requires only one **HwmcaGetBulk** request to the SE, and **HWIQUERY**, which specifies six attributes, also requires only one **HwmcaGetBulk** request.

Note: If **HWIQUERY** requests were called separately for each attribute in the past, a modification to the application to combine the attribute queries in a single **HWIQUERY** call can improve performance significantly.

7.3.2 Interactions and dependencies

These enhancement require an IBM System z9 (running at the latest microcode level) or higher to take advantage of performance improvements.

7.4 Documentation and reference

See the following IBM publications for documentation:

- *MVS Programming: Callable Services for High-Level Languages*, SA23-1377

  https://ibm.biz/BdFdWM
- *System z Application Programming Interface*s, SB10-7030-16 (PDF file)

  https://ibm.biz/BdFxj3

8

SDSF: System Display and Search Facility

This chapter describes the enhancements, functions, and features that are introduced with the System Display and Search Facility (SDSF) for IBM z/OS Version 2.1:

- 8.1, “64-bit support” on page 96
- 8.2, “8-character job class” on page 96
- 8.3, “Print enhancements” on page 99
- 8.4, “Extended console name for the user log” on page 102
- 8.5, “Sort enhancement” on page 104
- 8.6, “SET DISPLAY enhancement” on page 107
- 8.7, “JESPlex scoping” on page 108
- 8.8, “System symbols and filtering support” on page 109
- 8.9, “REXX and Java enhancements (BROWSE and LOG)” on page 111
- 8.10, “Security Assist SECTRACE facility” on page 113
- 8.11, “Migration and coexistence considerations” on page 116
- 8.12, “Documentation and reference” on page 116

© Copyright IBM Corp. 2014. All rights reserved.

8.1 64-bit support

Because the numbers of rows for many display areas continues to increase as the number of jobs and output elements that are allowed increases. As a result, SDSF receives ABEND878 and other symptoms because tables get too large or storage gets fragmented.

Starting with z/OS 2.1, SDSF row data and ULOG buffers are moved to 64-bit storage, and 64-bit storage is requested on SSI 80 calls. This mitigates storage fragmentation, and you can now use larger tables.

8.1.1 Invocation and use

If SDSF cannot get 64-bit storage, an ISF121I dialog message is issued. This message is displayed once per session after the first failure. SDSF attempts to obtain 31-bit storage, instead. If 31-bit storage is also unavailable, the request results in ABEND if the storage request was unconditional, and you receive a message similar to this one:

```
ISF121I Module modname was unable to obtain nnnnnnnn_nnnnnnnn bytes of storage (nnn
segments). Check MEMLIM value.
```

8.2 8-character job class

As JES2 started supporting job class names up to 8 characters in z/OS 2.1, SDSF is enhanced to support eight character job class in JES2 environment. With this enhancement, longer job class names are supported in both JES2 and JES3 SDSF environment

8.2.1 Invocation and use

Columns and action characters are added to panels. Names in parenthesis are the titles displayed in the panel.

Job class panel

Columns:

GROUP(Group) Indicates job class group. Now applies to both JES2 and JES3. Overtypeable ($TJOBCLASS,GROUP=). By default, it is not at the end of the line but, instead, after the STATUS column (same location for both the JES2 and JES3 display).

See Figure 8-1 on page 97.

```
Display  Filter  View  Print  Options  Search  Help
-------------------------------------------------------------------------------
SDSF JOB CLASS DISPLAY ALL CLASSES                    LINE 1-26 (43)
COMMAND INPUT ===>                                      SCROLL ===> CSR
NP   CLASS    Status   Group    Mode Wait-Cnt Xeq-Cnt  Hold-Cnt ODisp         Q
     A        NOTHELD           JES         2        6         1 ()            N
     B        NOTHELD           WLM                              ()            N
     C        NOTHELD           JES                              ()            N
     CLASS1   NOTHELD  XYZ      JES                              ()            N
     CLASS2   NOTHELD  XYZ      JES                              ()            N
     CLASS3   NOTHELD  XYZ      JES                              ()            N
     CLASS4   NOTHELD  XYZ      JES                              ()            N
     CLASS5   NOTHELD  XYZ      JES                              ()            N
     D        NOTHELD           JES                              ()            N
     E        NOTHELD           JES                              ()            N
     F        NOTHELD           JES                              ()            N
     G        NOTHELD           JES                              ()            N
     H        NOTHELD           JES                              ()            N
     I        NOTHELD           JES                              ()            N
     J        NOTHELD           JES                              ()            N
     K        NOTHELD           JES                              ()            N
     L        NOTHELD           WLM                              ()            N
     M        NOTHELD           WLM                              ()            N
     N        NOTHELD           JES                              ()            N
     O        NOTHELD           JES                              ()            N
     P        NOTHELD           JES                              ()            N
     Q        NOTHELD           JES                              ()            N
     R        NOTHELD           JES                              ()            N
     S        NOTHELD           JES                              ()            N
     STC                                    28       42          ()
     T        NOTHELD           JES                              ()            N
```

Figure 8-1 SDSF job class panel

CLACTIVE(Active) Indicates whether job class is active in **YES** or **NO**. Active status also controls highlighting of row. This field can be overtyped. (issues $TJOBCLASS,ACTIVE= command internally).

See Figure 8-2 on page 98.

```
Display  Filter  View  Print  Options  Search  Help
-------------------------------------------------------------------------------
SDSF JOB CLASS DISPLAY ALL CLASSES                    LINE 1-26 (43)
COMMAND INPUT ===>                                      SCROLL ===> CSR
NP   CLASS    r MC Scheduling-Env   JesLog        XBMProc  DupJob JobRC  Active
     A                              (NOSPIN)               NO     MAXRC  YES
     B                              (NOSPIN)               NO     LASTRC YES
     C                              (NOSPIN)               NO     MAXRC  YES
     CLASS1                         (NOSPIN)               NO     MAXRC  YES
     CLASS2                         (NOSPIN)               NO     MAXRC  YES
     CLASS3                         (NOSPIN)               NO     MAXRC  YES
     CLASS4                         (NOSPIN)               NO     MAXRC  YES
     CLASS5                         (NOSPIN)               NO     MAXRC  YES
     D                              (NOSPIN)               NO     MAXRC  YES
     E                              (NOSPIN)               NO     MAXRC  YES
     F                              (NOSPIN)               NO     MAXRC  YES
     G                              (NOSPIN)               NO     MAXRC  YES
     H                              (NOSPIN)               NO     MAXRC  YES
     I                              (NOSPIN)               NO     MAXRC  YES
     J                              (NOSPIN)               NO     MAXRC  YES
     K                              (NOSPIN)               NO     MAXRC  YES
     L                              (NOSPIN)               NO     MAXRC  YES
     M                              (NOSPIN)               NO     MAXRC  YES
     N                              (NOSPIN)               NO     MAXRC  YES
     O                              (NOSPIN)               NO     MAXRC  YES
     P                              (NOSPIN)               NO     MAXRC  YES
     Q                              (NOSPIN)               NO     MAXRC  YES
     R                              (NOSPIN)               NO     MAXRC  YES
     S                              (NOSPIN)               NO     MAXRC  YES
     STC        S                   (NOSPIN)               NO     MAXRC
     T                              (NOSPIN)               NO     MAXRC  YES
```

Figure 8-2 SDSF job class panel 2

Action character:

ST This action character is not allowed for multi-character job classes other than STC and TSU due to ST command syntax restrictions.

DA, I, ST, O, and H panels

Column:

JCLASS(C) The job class column is now 8 characters under JES2.

INIT panel

Column:

JCLASS(C) The job class column is now 8 characters under JES2.

ICLASS(Classes) Represents list of JES2 initiator classes, only in JES2 environment. Overtypeable field and overtype extension (+) allow up to 36 classes. See Figure 8-3 on page 99.

```
 Overtype Extension

Column Classes
Maximum length 36

Type a value.
===> AB

F1=Help  F12=Cancel
```

Figure 8-3 Overtype Extension pop-up panel for Column Classes in the SDSF INIT panel

ICLASS1-8(Class1-8) Display first eight classes that are defined to the initiator. ICLASS1 is an overtypeable field. The overtype extension (+) panel allows up to 8 classes (or class groups). See Figure 8-4.

When multi-character job classes or groups are present in a class list, ICLASS column can no longer be overtyped for that initiator. And if any multi-character job classes are assigned, they are displayed as a “period” in ICLASS list.

```
      Overtype Extension

Column Class1
Maximum length 8

Type values or use blanks to
erase values.
===> A
===> B
===> C
===> D
===> E
===> F
===> G
===> 1

F1=Help  F12=Cancel
```

Figure 8-4 Overtype Extension pop-up for the Class1 column in the SDSF INIT panel

SO panel

Column:

SCLASS The same as ICLASS. See “INIT panel” on page 98

SCLASS1-8 The same as ICLASS1-8. See the INIT panel description.

8.3 Print enhancements

In previous releases, when printing to SYSOUT from SDSF, the SYSOUT data set was always allocated with RECVM=VBA, LRECL=240, and ASA was always used for carriage control. Also, there was no ability to specify a writer name when printing to SYSOUT.

Starting with z/OS 2.1, when printing to SYSOUT, you can request SDSF to use the attributes of the source data set for the SYSOUT data set characteristics. You can also specify more attributes, including writer name, record length, and record format. In addition, carriage control is chosen depending on RECFM of destination data set. With this enhancement, printing as-is is possible.

Note: RECFM ending in A (VBA, FA, and so on) receives ASA carriage control and is converted from machine carriage control or inserted, if necessary. RECFM ending in M (VBM, FM, and so on) receives machine carriage control and is converted from ASA or inserted if necessary. All others (VB, F, and so on) receive no carriage control. Machine or ASA carriage control is stripped.

8.3.1 Invocation and use

The topics in this section explain the following changes for this enhancement:

- Columns in SDSF panels added
- REXX variables added
- Java methods added
- Customized properties (PROPLIST) option

Columns in SDSF panels

The following options can be specified on the SDSF Open Print panel:

`Writer name`	Specifies the writer name to be associated with the SYSOUT.
`Record format`	Specifies the RECFM of the SYSOUT. Default is VBA.
`Record length`	Specifies the LRECL of the SYSOUT. Default is 240.
`Use source attributes`	Attempts to obtain the following attributes from the source data, such as class, forms, destination, FCB, UCS, writer, process mode, RECFM, and LRECL. If `YES` is specified, the panel is redisplayed with the values obtained. Some properties might be unavailable in some contexts. $LRECL$ means the maximum LRECL of all data sets selected.

REXX variables

REXX variables are available for REXX:

`isfprtwritername`	Specifies the writer name for a print to sysout request
`isfprtsourceatts`	Indicates whether source attributes are to be fetched on a print to SYSOUT. Specify `YES` or `NO`. Explicitly specified attributes are honored even if `YES` is specified for this variable.
`isfprtrecfm`	Specifies the record format for a print to sysout request.
`isfprtlrecl`	Specifies the logical record length for a print to sysout request.

For a sample REXX exec to print to SYSOUT, see Example 8-1 on page 101.

Example 8-1 Sample REXX to print to SYSOUT using source attributes

```
/* REXX */
rc=isfcalls('ON')
  /* Access the ST panel */
Address SDSF "ISFEXEC ST"
lrc=rc
call msgrtn
if lrc<>0 then
 exit 20
  /* Assign the special variables that correspond to */
  /* the attributes of the print file. Unassigned */
  /* variables will use defaults. */
isfprtsourceatts="YES"
isfprtclass="A"
isfprtcopies="3"
isfprtwriter="UCHIDA"
  /* Issue an XSC action against the row to be printed */
do ix=1 to JNAME.0
 if JNAME.ix = "ZOSUPD2Z" then
  do
   Address SDSF "ISFACT ST TOKEN('"TOKEN.ix"') PARM(NP XSC)"
   lrc=rc
   call msgrtn
   if lrc<>0 then
    exit 20
 end
end
exit
  /* Subroutine to list error messages */
msgrtn: procedure expose isfmsg isfmsg2.
  /* The isfmsg variable contains a short message */
if isfmsg<>"" then
 Say "isfmsg is:" isfmsg
  /* The isfmsg2 stem contains additional descriptive */
  /* error messages */
do ix=1 to isfmsg2.0
 Say "isfmsg2."ix "is:" isfmsg2.ix
end
return
```

Java methods

More methods are available for Java:

`addISFPrtWriter()`	Specifies the writer's name.
`removeISFPrtWriter()`	Removes the specified writer's name.
`addISFPrtSourceAttributes()`	Indicates whether source attributes are to be fetched during a print to SYSOUT. Specify `YES` or `NO`. Explicitly specified attributes are honored even if `YES` is specified for this variable.
`removeISFPrtSourceAttributes()`	Removes the specified source attributes.

Note: The principal source of information for using Java with SDSF is the Javadoc that is supplied with SDSF. To use the Javadoc, follow these steps:

1. Use FTP or a similar process to download this file, in binary mode, to an empty directory on your workstation:

   `/usr/include/java_classes/isfjcallDoc.jar`

2. If you have the Java SDK installed on your workstation, use this command:
   `jar -xf isfjcallDoc.jar`

   Otherwise, use another utility available on your workstation to extract the file.

3. Navigate to the **index.html** file and open it with a web browser. When the file content is displayed, you can use the links to navigate to specific classes or topics, such as these examples:

Overview	Display an overview to using SDSF with Java
Package	Display a list of classes
Tree	Display a hierarchical view of classes
Index	Display an index to the Javadoc

Customized properties (PROPLIST)

Customized properties define customized values for certain SDSF properties that can be specified in ISFPARMS.

For carriage control, the `Print.CCTL.AlwaysUseASA` custom property is provided. This property specifies how SDSF's print function handles carriage control. Either `TRUE` or `FALSE` can be specified:

- `TRUE` causes SDSF to always use ASA carriage control when printing, regardless of the RECFM of the output data set.
- `FALSE` causes SDSF to base the carriage control on the record format (RECFM value) of the output.

8.4 Extended console name for the user log

In previous releases, **`ULOG`** allocated an extended console for the user log, based on either a TSO user ID or a `SET CONSOLE` value. For this reason, multiple instances of SDSF (split screen or multiple logons) sent messages to the initial session's ULOG. `SET CONSOLE` can be changed manually for each, but this is an inadequate solution if it is needed frequently. This can also be an problem for REXX if **`isfulog`** is used.

In z/OS 2.1, if an extended console is in use, SDSF optionally attempts to use a different extended console name by appending one character to the original name. With this enhancement, unique ULOGs are provided for each session for split screens or multiple logons.

See Figure 8-5 on page 103 for a screen capture of the output.

```
 Display  Filter  View  Print  Options  Search  Help
 ------------------------------------------------------------------------------
 SDSF ULOG  CONSOLE ZOSUPD2                          LINE 0       COLUMNS 02- 81
 COMMAND INPUT ===>                                 SCROLL ===> CSR
******************************** TOP OF DATA *********************************
SC74      2013116  11:51:51.38            ISF031I CONSOLE ZOSUPD2 ACTIVATED
******************************** BOTTOM OF DATA ******************************









 .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
   Display  Filter  View  Print  Options  Search  Help
 ------------------------------------------------------------------------------
 SDSF ULOG  CONSOLE ZOSUPD2$                         LINE 0       COLUMNS 02- 81
 COMMAND INPUT ===>                                 SCROLL ===> CSR
******************************** TOP OF DATA *********************************
SC74      2013116  12:05:42.69            ISF032I CONSOLE ZOSUPD2 ACTIVATE FAIL
SC74      2013116  12:05:42.69            ISF042I CONSOLE ZOSUPD2 IS IN USE
SC74      2013116  12:06:01.12            ISF041I CONSOLE NAME ZOSUPD2 MODIFIED
SC74      2013116  12:06:01.12            ISF031I CONSOLE ZOSUPD2$ ACTIVATED
******************************** BOTTOM OF DATA ******************************









   -S1      *S2
```

Figure 8-5 Console name with one character $ appended in CONMOD ON

8.4.1 Invocation and use

Extending the console requires using these methods, which are described in the subsections that follow:

- **SET CONMOD** command
- REXX variables
- Java methods
- Customized properties (PROPLIST)

SET CONMOD command

The **SET CONMOD** command controls modification of the console name when the console name is in use.

SET CONMOD ON | OFF | ? — **ON** indicates that console name needs to be modified if activation fails because the console name is in use. **OFF** indicates that the console name is not to be modified, and **?** indicates that the current value is to be queried. The value of **SET CONMOD** is saved in the ISPF profile across SDSF sessions.

REXX variables

isfconmod — Corresponds to the **SET CONMOD** command. Valid values are **YES** and **NO**

When set to TRUE, **SET CONMOD ON** is not allowed. The first instance of the extended console is always used.

Default: FALSE

Customized properties (PROPLIST)

Console.EMCS.ConModChars — Specifies a list of up to 32 characters to choose from (in order) when appending a character to the extended console name. Valid characters are A-Z, 0-9, $, #, and @.

Default: $#@12345

8.5 Sort enhancement

In previous releases, SDSF supported only two sort criteria at a time. But sometimes, two criteria were not enough. In z/OS 2.1, the number of sort criteria increased to 10, and SET DISPLAY options are added to display sort criteria and filters. With this enhancement, there is greater flexibility sorting panel data.

8.5.1 Invocation and use

The sections that follow describe the three ways that you can use this enhancement:

- Through the SDSF command
- Through the REXX variable
- Through the Sort pop-up panel

Through the SDSF command

The **SORT** command continues to support only two criteria at a time on the command line. To sort on more than two criteria, the sort command must be issued multiple times. For example, to use 10 criteria, the sort command must be run five times, specifying two sort criteria each time. Example 8-2 shows the command syntax.

Example 8-2 SORT command syntax

```
SORT (column) (A | D) (column) (A | D)
     (+ | -)column (A | D)
     (OFF | ON)
     (?)
```

column — The title of the column to be sorted. Specify the title as it appears on the panel, or abbreviate it as a name that is unique on the panel. If the title contains blanks, either use an abbreviation that contains no blanks or enclose the title in quotation marks.

A or **D** — **A** specifies that the sort order is to be ascending; D specifies descending. **A** is the default, but you must specify either A or D when you enter two columns.

+ or **-** — Adds (+) or removes (-) sort criteria for a column.

ON or **OFF**
: Turns sorting on or off for the current panel. Even if off is specified, the sort criteria is retained and used when it is turned on again.

?
: Displays the sort criteria on the command line or in a pop-up panel. Under the Time Sharing Option (TSO), if the criteria do not fit on the command line, they are displayed on the message line.

  For example, if you want to set the job name and the job ID as ascending sort criteria, run this command:

  ```
  SORT JOBNAME A JOBID A
  ```

  And, if you want to set the job name and job ID as descending sort criteria also, run this command:

  ```
  SORT +TGNUM D TGPCT D
  ```

Through the REXX variable

The **isfsort** REXX variable now supports up to 10 criteria.

ISFSORT
: Specifies the sort criteria (up to 10 columns, with ascending or descending order). Use column names rather than column titles. Assigning the value to null (**isfsort=""**) sorts the panel by using the fixed field (the first column). See the **SORT** command in the online help for the syntax.

ISFSORT2
: Specifies the sort criteria (up to 10 columns, ascending or descending order) in the Secondary panel. Use column names rather than column titles. Assigning the value to null sorts the panel by using the fixed field (the first column). See the **SORT** command for other syntax.

Note: Secondary panels are accessed with an action character from another panel. For example, when you use the **?** action character from the Status panel to access the Job Data Set (JDS) panel, JDS is a secondary panel.

Through the Sort pop-up panel

The Sort pop-up panel can be used to specify the complete list. The list replaces “Major” and “Minor” column specification, which was used in previous releases. Figure 8-6 on page 106 shows an example from a previous z/OS release.

```
 Sort         Row 1 to 11 of 45
Command ===>

Type 1 or 2 column names and A or D for ascending or
descending order.  Columns are listed below.

Major Column                    Order    (A or D)
Minor Column                    Order    (A or D)


      JOBNAME
      JobID
      Owner
      Prty
      Queue
      C
      Pos
      SAff
      ASys
      Status
      PrtDest
```

Figure 8-6 “Major” and “Minor” column specification popup panel in a previous release

In the sort pop-up panel, enter Column Title and ascending (A) or descending (D), as shown in Figure 8-7. As this example shows, you can enter / for the Column Title to see a list.

```
Sort
Command ===> _____________________________________________

Type sort criteria.  Type / for Column Title to see a
list.  Press F11/23 to clear all sort criteria.

Sorting is ON

Column Title       A or D (Ascending or Descending)
JOBID              A
JOBNAME            A
OWNER              A
PRTY               D
QUEUE              D
C                  A
POS                D
SAFF               A
ASYS               A
STATUS             D
```

Figure 8-7 The Sort pop-up panel

8.5.2 Migration and coexistence considerations

In previous releases, all sort criteria set were saved in the ISPF profile as one ISPF variable (ISFSRTC for JES2, ISFSRTC3 for JES3). Starting with z/OS 2.1, sort criteria for each display are stored in separate variables, such as ISF2Snnn for JES2 and ISF3Snnn for JES3.

ISFSRTC and ISFSRTC3 are still maintained for compatibility, but they are populated only with the first two sort criteria. If both newer and compatibility variables exist in the same profile, the newer variable is used if the first two criteria match the compatibility variable. The compatibility variable is used if they are different. This ensures that the changes are honored if the sort criteria are changed on a previous release. If the sort criteria are not changed on a previous release, up to 10 sort criteria are remembered when returning to z/OS 2.1.

8.6 SET DISPLAY enhancement

In previous releases, the **SET DISPLAY ON** command displayed up to two sort criteria and count of filters. In z/OS 2.1, according to the sort enhancement described in 8.5, “Sort enhancement” on page 104, the number of additional criteria is listed as *+n* in addition to two sort criteria and count of filters.

A **SET DISPLAY LONG** command is provided also. Use this rather than the **SET DISPLAY ON** command if you want all of the sort criteria and count of filters to be displayed.

REXX special variable **ISFDISPLAYMODE** is also enhanced.

8.6.1 Invocation and use

When you run **SET DISPLAY ON** in an SDSF panel, the sort criteria and count of filters will be displayed. For example, three criteria display the first 2 and a +1 in Figure 8-8.

```
 Display  Filter  View  Print  Options  Search  Help

------------------------------------------------------------------------------
 SDSF HELD OUTPUT DISPLAY ALL CLASSES LINES 22,657       SET COMMAND COMPLETE
 COMMAND INPUT ===>                                        SCROLL ===> CSR
 PREFIX=**  DEST=(ALL)  OWNER=ZOSUPD*  SORT=JOBNAME/A Dest/A +1  SYSNAME=
 FILTERS=6
 NP   JOBNAME  JobID    Owner    Prty C ODisp Dest                  Tot-Rec
Tot-
      UCHIDAX  JOB05358 ZOSUPD2   144 X HOLD  LOCAL                     197
      ZOSUPD2  TSU05293 ZOSUPD2   144 S HOLD  LOCAL                     445
      ZOSUPD4A JOB04084 ZOSUPD4   144 T HOLD  LOCAL                   1,510
      ZOSUPD4A JOB04636 ZOSUPD4   144 T HOLD  LOCAL                      94
      ZOSUPD4A JOB04637 ZOSUPD4   144 T HOLD  LOCAL                      94
      ZOSUPD4A JOB04668 ZOSUPD4   144 T HOLD  LOCAL                      73
      ZOSUPD4A JOB04680 ZOSUPD4   144 T HOLD  LOCAL                   1,134
      ZOSUPD4A JOB04683 ZOSUPD4   128 T HOLD  LOCAL                   2,622
      ZOSUPD4A JOB04684 ZOSUPD4   128 T HOLD  LOCAL                   2,604
```

Figure 8-8 SET DISPLAY ON command with three sort criteria and six filters

When you run **SET DISPLAY LONG** in the SDSF panel, all sort criteria and all filters are displayed. The displayed values can exceed two lines. See Figure 8-9 on page 108.

```
 Display  Filter  View  Print  Options  Search  Help
-------------------------------------------------------------------------------
 SDSF HELD OUTPUT DISPLAY ALL CLASSES LINES 22,657       SET COMMAND COMPLETE
 COMMAND INPUT ===>                                        SCROLL ===> CSR
 PREFIX=**  DEST=(ALL)  OWNER=ZOSUPD*  SORT=JOBNAME/A Dest/A Owner/A  SYSNAME=
 FILTERS=JOBNAME,EQ,* ODisp,EQ,* Status,EQ,* Wtr,EQ,* Tot-Rec,LT,100000
 Output-Hold-Text,EQ,*
 NP   JOBNAME  JobID    Owner    Prty C ODisp Dest                  Tot-Rec
Tot-
      UCHIDAX  JOB05358 ZOSUPD2   144 X HOLD  LOCAL                     197
      ZOSUPD2  TSU05293 ZOSUPD2   144 S HOLD  LOCAL                     445
      ZOSUPD4A JOB04084 ZOSUPD4   144 T HOLD  LOCAL                   1,510
      ZOSUPD4A JOB04636 ZOSUPD4   144 T HOLD  LOCAL                      94
      ZOSUPD4A JOB04637 ZOSUPD4   144 T HOLD  LOCAL                      94
      ZOSUPD4A JOB04668 ZOSUPD4   144 T HOLD  LOCAL                      73
      ZOSUPD4A JOB04680 ZOSUPD4   144 T HOLD  LOCAL                   1,134
      ZOSUPD4A JOB04683 ZOSUPD4   128 T HOLD  LOCAL                   2,622
      ZOSUPD4A JOB04684 ZOSUPD4   128 T HOLD  LOCAL                   2,604
```

Figure 8-9 SET DISPLAY LONG command with three sort criteria and six filters

8.7 JESPlex scoping

In previous releases, some SDSF panels, such DA, CK, PS, and ENC, have Parallel Sysplex scope. But there are environments that have multiple JESPlexes in a single Parallel Sysplex, with separate operations for each JESPlex. In this case, it is better to have JESPlex scope, so you can manage those operations.

Starting with z/OS 2.1, you have the option to give panels with Parallel Sysplex scope JESPlex scope. JESPlexes within a Parallel Sysplex can now be managed independently, which makes operations easier and more flexible.

8.7.1 Invocation and use

The customized properties (PROPLIST) includes the following options:

Panel.DA.JESPlexScope	Sets the value for DA panels that default to Parallel Sysplex scope.
Panel.CK.JESPlexScope	Sets the value for CKpanels that default to Parallel Sysplex scope.
Panel.PS.JESPlexScope	Sets the value for PS panels that default to Parallel Sysplex scope.
Panel.ENC.JESPlexScope	Sets the value for ENC panels that default to Parallel Sysplex scope.
Panel.All.JESPlexScope	Sets the value for all panels that default to Parallel Sysplex scope.

When these properties set to `TRUE`, the corresponding panel has JESPlex scope. The scope encompasses all systems on which an active member of the JESPlex exists, regardless of whether the member is primary or secondary. Jobs still display if they are running on one of

these systems, even if the sysplex is running poly-JES and the job is not associated with the JESPlex.

8.8 System symbols and filtering support

System symbols offer the greatest advantage when two or more systems require different configurations, data sets, jobs, procedures, or entire PARMLIB members. But in previous releases, SDSF filtering did not support system symbols. Sometimes it was a problem when system ID was the filtering value in shared SDSF environment. If system symbols can be used in this case, filtering with the system ID for each system is easier.

In Version 2.1, system symbols can be specified within filter values. Filters can have different values on different systems in a shared SDSF environment.

Note: “Shared SDSF environment” refers to an environment that is sharing an ISPF profile and ISFPARMS between two or more systems.

8.8.1 Invocation and use

Specify system symbols in the following ways:

- **`FILTER`** command

  For example:

`FILTER SysID EQ &SYSNAME.`	SDSF filters all rows where **`SysID`** is equal to the value of the local system’s system name.
`FILTER Dest EQ &SYSNAME(1:2).`	SDSF filters all rows where **`Dest`** begins with the first two characters of the local system name.

- **`Filter`** pull-down menu

  The SDSF **`Filter`** pull-down option is also available for system symbols. See Figure 8-10 on page 110 for an example. In addition, symbols that are available in each environment are shown, and they can be selected from the filter menu by selecting **Prompt (PF4)** with a / in the value column.

```
 Filter            Row 1 to 9 of 25
Command ===> ____________________________________________

Type filter criteria. Press F4/16 in the Column or Oper
field for values, or in the Value field for system symbols.
Press F11/23 to clear all filter criteria.

Filtering is  ON

AND/OR between columns   AND  (AND/OR)
AND/OR within a column   OR   (AND/OR)

Column             Oper  Value (may include * and %)
DEST               EQ    &SYSNAME.
__________________ __    ____________________________
__________________ __    ____________________________
__________________ __    ____________________________
__________________ __    ____________________________
__________________ __    ____________________________
__________________ __    ____________________________
__________________ __    ____________________________
__________________ __    ____________________________
```

Figure 8-10 Example of Filter pull-down menu

Current symbolic values are also shown for reference in Figure 8-11.

```
System Symbols      Row 1 to 12 of 39
Command ===> ____________________________________________

The values may change. Use them only as examples.

Selection: _____
       Symbol      Value
   1.  &CATPK.     Z21CAT
   2.  &CIC42SET.  A
   3.  &CIC42VOL.  BH5CI1
   4.  &DAY.       30
   5.  &DLIB1.     Z21DZ1
   6.  &DLIB2.     Z21DZ2
   7.  &DLIB3.     Z21DZ3
   8.  &HHMMSS.    215320
   9.  &HR.        21
  10.  &IFAPRDXX.  00
  11.  &JDAY.      120
  12.  &JNKPK.     Z21JNK
```

Figure 8-11 Example of symbols available and current values used for each symbol

8.9 REXX and Java enhancements (BROWSE and LOG)

In previous releases, to browse SDSF data sets from REXX and Java, variables were needed for each data set that you wanted to browse. That required you to manage allocations of individual data sets. For example, with REXX, action character **SA** is required to allocate all data sets associated with the item, and **SJA** is required to allocate the JCL data set. The number of concurrent allocations for jobs with many data sets can result in other problems, such as below-the-line storage shortages.

In z/OS 2.1, you can use the **ISFBROWSE** REXX command to browse data. With this command, it is easier to look at spool the data sets, syslog, and operlog from REXX and Java. All allocation and unallocation processing and all OPEN and CLOSE calls are managed by SDSF. Also, Java methods that provide the function equivalent to **ISFBROWSE** are provided.

Also, the **ISFLOG** REXX command and Java methods that provide the function equivalent to **ISFLOG** are enhanced.

8.9.1 Invocation and use

This section describes the use of the **BROWSE** and **LOG** features.

ISFBROWSE command use

REXX command syntax:

```
ISFBROWSE panel_name TOKEN(token) (options)
```

Where **options** offers a list of options. The closing parenthesis is optional. These are the available options:

JCL	Browse just the JCL (jobs only).
NOCLOSE	Leave the data set open for subsequent requests to avoid the overhead of closing, deallocating, reallocating, and reopening the data set. To undo the allocations, use **ISFBROWSE** without **NOCLOSE** and set the *ISFSTARTLINETOKEN* special variable.
VERBOSE	Add diagnostic messages to the **isfmsg2** stem variable. The messages describe each variable that is created by SDSF. This can be useful for troubleshooting as you develop REXX execs.

See Figure 8-12 on page 112 for an example.

```
/* REXX */
rc=isfcalls("on")
/* Access status panel */
Address SDSF "ISFEXEC ST"
/* For all jobs */
do ix=1 to JNAME.0
  /* Examine job name */
  if JNAME.ix = "ZOSUPD2" then
    do
    /* Return 100 lines at a time */
    isflinelim = 100
      /* Loop until EOF reached */
      do until isfnextlinetoken=''
        /* Read lines */
        Address SDSF "ISFBROWSE ST TOKEN('"token.ix"') (JCL)"
          do jx=1 to isfline.0
            /* Display lines */
            say isfline.jx
          end
          /* Set next read origin */
          isfstartlinetoken = isfnextlinetoken
      end
    end
end
rc = isfcalls("off")
```

Figure 8-12 Sample REXX for using ISFBROWSE

Java browsing methods use

The following methods are added and documented in the SDSF Javadoc:

browse()

browseJCL()

addISFStartLineToken()

addISFScroll()

addISFScrollType()

addISFFind()

addISFFindLim()

getLineResults()

getLineRecordList()

getFirstLineToken()

getNextLineToken()

ISFLOG command use

More options for ISFLOG are provided. This is the command syntax:

`ISFLOG READ TYPE(SYSLOG|OPERLOG) (`*`options`*

You can specify either of these for ***options***:

WTOR	Returns all WTOR messages in the *isfwtor* stem variable
CCASIS	Returns all records without carriage control

Java logging methods use

The following methods are added and documented in the SDSF Javadoc:

addISFStartLineToken()

addISFScroll()

addISFScrollType()

addISFFind()

addISFFindLim()

getLineResults()

getLineRecordList()

getFirstLineToken()

getNextLineToken()

8.10 Security Assist SECTRACE facility

In previous releases, it was difficult to determine why a user could not access functions in SDSF, because the security environment was controlled by SAF, ISFUSER exit, and ISFPARMS (the last is usually used for backup definition in case the SAF profile does not exist). In addition, SDSF often shows only a panel message for a security error and gives no information about it.

In z/OS 2.1, a SECTRACE facility is added. This SECTRACE provides information about the results of security decisions (SAF or non-SAF) made. This provides diagnostic information about why access to panels, commands, or actions are denied by SDSF. Users can receive this information with ULOG messages or write-to-programmer messages. By using this facility, it is easier for a a user to diagnose security problems.

Examples of messages that can be received by activating the SECTRACE facility are shown in Figure 8-13 on page 114.

```
ISFO50I USER=D96CLW1 GROUP=ISFSPROG PROC=SDSF31EJ TERMINAL=LOCALC11

ISFO51I SAF Access allowed SAFRC=0 ACCESS=READ CLASS=SDSF
RESOURCE=ISFCMD.DSP.STATUS.JES2 ISFO53I COMMAND=LOG Access denied

ISFO53I COMMAND=I Access denied

ISFO51I SAF Access denied SAFRC=8 ACCESS=UPDATE CLASS=SDSF RESOURCE=ISFATTR.JOB.CLASS

ISFO51I SAF No decision SAFRC=4 ACCESS=UPDATE CLASS=SDSF RESOURCE=ISFATTR.JOB.PRTY

ISFO58I COLUMN=Prty Access allowed USERLEVEL=7 REQLEVEL=3

ISFO53I COMMAND=PREFIX Access allowed

ISFO55I ACTION=D Access denied  USERLEVEL=0 REQLEVEL=1                    RSN=09 command
authority insufficient

ISFO55I ACTION=H Access denied USERLEVEL=7 REQLEVEL=3 BPXAS SO000027 RSN=08 job name in
exclude list

ISFO57I GROUP=ISFSPROG Access allowed USERAUTH=OPER,ACCT,JCL REQAUTH=OPER,ACCT,JCL

ISFO51I SAF Access allowed SAFRC=0 ACCESS=READ CLASS=JESSPOOL
RESOURCE=SYSA1N.D96CLW1.D96CLW1.T0000036.D0000002.JESMSGLG
```

Figure 8-13 Sample SECTRACE messages

Note: Row-by-row decisions (such as Multi-Level Security) are not traced by SECTRACE.

8.10.1 Invocation and use

There are five ways to control the SECTRACE facility;

- **SET SECTRACE** command:

  Issue the **SET SECTRACE** command in any SDSF panels. Choose one of these options:

ON or ULOG	Indicates that messages are placed in ULOG
WTP	Indicates that messages are to be issued as write-to-programmer
OFF	Indicates that no messages are to be issued
?	Indicates the current value is to be queried

- SDSF command option:

  With the SECTRACE option on the SDSF command, you can turn on security tracing as soon as you access SDSF. For example:

  SDSF,SECTRACE(WTP)

  You will receive messages from SECTRACE before the SDSF Primary Option menu appears. See Figure 8-14 on page 115 for an example.

```
 READY
SDSF,SECTRACE(WTP)
 ISFO50I USER=ZOSUPD2 GROUP= PROC=IKJACCNT TERMINAL=SC38TC14
 ISFO51I SAF No decision    SAFRC=4 ACCESS=READ CLASS=SDSF RESOURCE=GROUP.UCHIDA
G.SDSFYU
 ISFO57I GROUP=UCHIDAG Access allowed USERAUTH=OPER,ACCT,JCL REQAUTH=OPER
 ISFO50I USER=ZOSUPD2 GROUP=UCHIDAG PROC=IKJACCNT TERMINAL=SC38TC14
 ISFO51I SAF Access allowed SAFRC=0 ACCESS=READ CLASS=SDSF RESOURCE=ISFCMD.FILTE
R.PREFIX
 ISFO51I SAF Access allowed SAFRC=0 ACCESS=READ CLASS=SDSF RESOURCE=ISFCMD.FILTE
R.OWNER
 ISFO51I SAF Access allowed SAFRC=0 ACCESS=READ CLASS=SDSF RESOURCE=ISFOPER.DEST
.JES2
 ISFO51I SAF Access allowed SAFRC=0 ACCESS=READ CLASS=SDSF RESOURCE=ISFOPER.ANYD
EST.JES2
 ISFO51I SAF Access allowed SAFRC=0 ACCESS=READ CLASS=SDSF RESOURCE=ISFCMD.DSP.A
CTIVE.JES2
 ISFO51I SAF Access allowed SAFRC=0 ACCESS=READ CLASS=SDSF RESOURCE=ISFCMD.DSP.I
NPUT.JES2
 ISFO51I SAF Access allowed SAFRC=0 ACCESS=READ CLASS=SDSF RESOURCE=ISFCMD.DSP.O
UTPUT.JES2
 ISFO51I SAF Access allowed SAFRC=0 ACCESS=READ CLASS=SDSF RESOURCE=ISFCMD.DSP.H
ELD.JES2
 ***
```

Figure 8-14 Activating SECTRACE facility from the SDSF command option

- REXX and Java:

  The **ISFSECTRACE** special variable is provided. If you specified `ISFSECTRACE ON`, **ISFMSG2** contains security trace messages, as shown in Figure 8-15. Also, **ISFULOG** contains security trace messages if you specified **ISFSECTRACE WTP**. See Figure 8-15 for an example.

```
isfmsg2.1 is: ISFO57I GROUP=ISFSPROG Access allowed USERAUTH=OPER,ACCT,JCL REQA
UTH=OPER,ACCT,JCL
isfmsg2.2 is: ISF767I Request completed.
```

Figure 8-15 Message that is contained in isfmsg2 when SECTRACE facility is activated by REXX

- **Options** pull-down menu:

  SECTRACE facility can also be activated from the SDSF panel **Options** pull-down menu. Select **22. Set security trace**, and you will see the **Select an option to control security trace** pop-up menu that is shown in Figure 8-16 on page 116.

```
             Set Security Trace

Select an option to control security trace.

    1.  Send messages to the user log
    2.  Issue write-to-programmer messages
    3.  Turn security trace off


F1=Help  F12=Cancel
```

Figure 8-16 Set Security Trace pop-up panel

8.11 Migration and coexistence considerations

If you are sharing z/OS 2.1 SDSF server parameters with earlier releases of SDSF, you must install the toleration PTFs that are associated with APARs PM37714 and PM78102:

- For z/OS V1R10 SDSF, you must install PTFs UK90037 and UK90708.
- For z/OS V1R11 SDSF, you must install PTFs UK90036 and UK90709.
- For z/OS V1R12 SDSF, you must install PTFs UK90034 and UK90710.
- For z/OS V1R13 SDSF, you must install PTFs UK90035 and UK90711.

8.12 Documentation and reference

z/OS SDSF Operation and Customization, SA23-2274, in the z/OS V2R1 Information Center:

https://ibm.biz/BdFxY8

9

Security server

This chapter describes the enhancements to the security server and the IBM Resource Access Control Facility (RACF) that were introduced in IBM z/OS Version 2 Release 1 (2.1). It covers the following changes:

- 9.1, “Removal of the BPX.DEFAULT.USER profile” on page 118
- 9.2, “RACF database unload utility record type” on page 121
- 9.3, “RACF Remote Sharing Facility TCP/IP support for TLS V1.2” on page 123
- 9.4, “RRSF TCP/IP support for IPv6” on page 124
- 9.5, “Comments and blank lines now allowed in the RACF parameter library” on page 125
- 9.6, “Enterprise PKCS#11 support” on page 127
- 9.7, “RACDCERT displays additional information for digital certificate chains” on page 130
- 9.8, “Prevent deletion of a certificate with a pending request” on page 136
- 9.9, “Documentation and reference” on page 138

© Copyright IBM Corp. 2014. All rights reserved.

9.1 Removal of the BPX.DEFAULT.USER profile

IBM z/OS V1R13 (1.13) was the last release to support the **FACILITY** class profile for **BPX.DEFAULT.USER**. From z/OS V2.1onward, this profile is ignored if it is still defined.

Because many users of z/OS UNIX can share a UID and GID when **BPX.DEFAULT.USER** is used, use of this profile caused several problems:

- Shared UIDs produce mismatched audit logs.

  Because the UID must correspond to the RACF user ID, sharing a UID between more than two user IDs causes a problem when auditing z/OS UNIX activity. For example, when a user with UID 1 is audited for an unauthorized access, you might not know the actual RACF user ID from the log if UID 1 is shared.

- If a UNIX service creates a resource while running with a shared UID, that resource is available to all users who are running with that shared UID.

  Resources that have owner attributes, such as directories and files, can be shared by users who are sharing the same UID. For example, even if access to a file is controlled with permission bits to restrict access for all UIDs except the owner of the file, it still can be accessed by all users who share a UID.

- Some UNIX services are disallowed.

  The use of **kill()**, **sigqueue()**, **pidaffinity()**, and **ptrace** is disallowed, because these services require a unique UID.

9.1.1 Invocation and use

With the removal of BPX.DEFAULT.USER, you must plan for one of two ways to assign users a UID or GID:

- Assign a unique UID and GID manually to each user who uses z/OS UNIX.
- Assign a unique UID and GID automatically by using the **FACILITY** class profile **BPX.UNIQUE.USER** support, which is available from z/OS V1R11 (1.11).

Assign a unique UID and GID manually

Use the **ALTUSER** command to assign an OMVS segment to each user. There are two ways to do this:

- Define UIDs explicitly with the **ALTUER** command. RACF still does not prevent sharing of UIDs and GIDs unless the profile SHARED.IDS in the UNIXPRIV class is defined:

  ```
  ALTUER userid OMVS(UID(user-identifier))
  ```

- Use the **AUTOUID** option in the **ALTUSER** command to enforce the uniqueness:

  ```
  ALTUSER userid OMVS(AUTOUID)
  ```

  To use the **AUTOUID** option, there are several requirements. For example, the RACF database must be at Application Identity Mapping (AIM) stage 2 or later, and the **UNIXPRIV** class must be set up properly. See “Enabling automatic assignment of unique UNIX identities” in *Security Server RACF Security Administrator's Guide,* SA23-2289 for more information (see 9.9, “Documentation and reference” on page 138).

Assign a unique UID and GID automatically

Unique UIDs and GIDs can be automatically assigned the first time that a user uses z/OS UNIX. In this way, no administrative intervention is required for each assignment. To use

automatic UID and GID assignments, there are several requirements, in addition to requirements for using the **AUTOUID** option, as already described. See Table 9-1 for a comparison of requirements. Also, see “Automatically assigning unique IDs through UNIX services” in the *Security Server RACF Security Administrator's Guide*, SA23-2289, for more information (see 9.9, “Documentation and reference” on page 138).

Table 9-1 Steps that are necessary for using each option

Steps	When defining UIDs explicitly	When defining UIDs explicitly by using the AUTOUID option	When defining UIDs automatically
Upgrade RACF database AIM Stage 3	Not needed	Not needed AIM stage 2 or later	Needed
Define SHARED.IDS profile for the UNIXPRIV class	Not needed	Needed	Needed
RACLIST the UNIXPRIV class	Not needed	Needed	Needed
Define the BPX.NEXT.USER profile for the FACILITY class	Not needed	Needed	Needed
Define the BPX.UNIQUE.USER profile for the FACILITY class	Not needed	Not needed	Needed

Note: Starting with z/OS 2.1, BPX.UNIQUE.USER is enhanced to allow specification of RACF variable **&racuid** in the home directory field of the model user’s OMVS segment:

```
ALTUSER BPXMODEL OMVS(HOME(/u/&racuid))
```

This definition substitutes a user ID for **&racuid** when a new OMVS segment is created for a user who is using **BPX.UNIQUE.USER**. If you use uppercase to specify &RACUID, a user ID in uppercase is substituted. If you specify it in lowercase, &racuid, a lowercase ID is substituted. This substitution is not made when the path name exceeds 1023 characters. If the RACF database is shared with down-level systems, there is no substitution on the down-level system.

Check which definitions are already defined to your system

The REXX **bpxcheck** program is available for analyzing your system and reporting on the relevant functions that are activated. Figure 9-1 on page 120 shows sample **bpxcheck** output. You can download the program from the RACF Downloads web page:

http://www.ibm.com/systems/z/os/zos/features/racf/downloads

```
Checking your authority to use this program:

You are authorized to IRR.RADMIN.LISTUSER.
You have the system SPECIAL attribute as well.
***
You are authorized to IRR.RADMIN.LISTGRP.
You are authorized to IRR.RADMIN.RLIST.
You are authorized to IRR.RADMIN.SETROPTS.LIST.

You have all the required authority! Continuing...

------------------------------------------------------------

The RACF database is in AIM stage 3.

------------------------------------------------------------

FACILITY class profile BPX.DEFAULT.USER is not defined.

------------------------------------------------------------

UNIXPRIV class profile SHARED.IDS is defined.
The UNIXPRIV class is active.
The UNIXPRIV class is RACLISTed.

------------------------------------------------------------

FACILITY class profile BPX.NEXT.USER is defined.

 UIDS are being assigned starting with 129 with an upper boundary of the maximu
m value of 2,147,483,647.
 GIDS are being assigned starting with 16 with an upper boundary of the maximum
 value of 2,147,483,647.
------------------------------------------------------------

FACILITY class profile BPX.UNIQUE.USER is not defined.

------------------------------------------------------------
```

Figure 9-1 Sample bpxcheck output

Migration and coexistence considerations

z/OS 2.1 systems do not use the BPX.DEFAULT.USER profile. If automatic generation of UIDs and GIDs is enabled, RACF automatically assigns a unique UID and GID. If a user without a UID and GID requests a service that requires a UID or GID (and automatic assignment of UIDs and GIDs are not enabled), an error notice indicates that the user's profile has no OMVS segment.

The following two RACF health checks and a migration health check are available:

RACF_UNIX_ID Checks whether RACF automatically assigns unique UIDs and GIDs when users without OMVS segments use certain UNIX services. If you are not using RACF to assign UIDs and GIDs, the check informs you that you must assign UIDs and GIDs before they are needed.

	If you are currently using **BPX.DEFAULT.USER** support, the check issues an error message. If you are already using **BPX.UNIQUE.USER** support, the check verifies requirements and indicates whether any exceptions are found. This check is also available for z/OS Versions 1.12 and 1.13 with the PTFs for APAR OA37164.
RACF_AIM_STAGE	Checks whether the RACF database is updated to AIM Stage 3. As previously described, Stage 3 is required for automatic assignment of UIDs and GIDs. AIM Stage 3 is valuable not only for UNIX System Service identity, but also for other applications, such as a DCE (Distributed Computing Environment). This check is also available for z/OS Versions 1.12 and 1.13 with the PTFs for APAR OA37164.
ZOSMIGV2R1_DEFAULT_UNIX_ID	Similar to the **RACF_UNIX_ID** check, **ZOSMIGV2R1_DEFAULT_UNIX_ID** checks whether you are relying on **BPX.DEFAULT.USER** or not. This check is available for users who are planning to migrate systems from z/OS Version 1.12 or 1.13 to Version 2.1. You need program temporary fixes (PTFs) for APAR OA37164.

9.2 RACF database unload utility record type

The RACF database unload utility (IRRDBU00) enables installations to create a sequential data set from a RACF database. The output sequential file from this utility is a relational representation of a RACF database.

In previous releases, although some information for digital certificate was provided with the General Resource Certificate Data Record (type 560), other information from the digital certificate was not unloaded by IRRDBU00, such as the signature algorithm. As a result, there was no way to search the IRRDBU00-generated records to query information for a certificate, such as the signature algorithm.

In z/OS 2.1, IRRDBU00 provides a record called the General Resource Certificate Information Record (type 1560). This record contains additional information that is associated with the digital certificate. It contains the subject's distinguished name, the issuer's distinguished name, and the signature algorithm.

9.2.1 Invocation and use

When generating a digital certificate, information such as the subject's distinguished name, issuer's distinguished name, and the signature algorithm is stored in the certificate that is being generated. This information is unloaded by IRRDBU00 in record type 1560.

IRRDBU00 assigns a four-digit number record type for every record that it creates, which is in PPSF format:

PP	Profile type
S	Segment number
F	Repeat group within the segment

With z/OS 2.1, a record type is added. A record type prefix (PP) of 1*x* (where x is a number) is now used to identify extended field processing records (record type 1560 is an example).

The record type from previous releases called the General Resource Certificate Data Record (record type 0560) is now called General Resource Certificate *Information* Record. The added record type is 1560.

You can see these record types in the first four columns of the IRRDBU00 output that is shown in Figure 9-2.

For more information about PPSF, see IRRDBU00 record types in the *Security Server RACF Macros and Interfaces*, SA23-2288 (see 9.9, “Documentation and reference” on page 138).

```
File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
ssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss
EDIT       ZOSUPD2.IRRDBU.V2R1                              CHARS '1560' found
Command ===>                                                  Scroll ===> CSR
=COLS> ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
002309 0500 0D8B4FEEAAD2185BF4756A9D29E17FFB.OU=Class¢1¢Public¢Primary¢Certific
002310 1560 0D8B4FEEAAD2185BF4756A9D29E17FFB.OU=Class¢1¢Public¢Primary¢Certific
002311 0560 0D8B4FEEAAD2185BF4756A9D29E17FFB.OU=Class¢1¢Public¢Primary¢Certific
002312 0500 00.personal-basic@thawte.com.CN=Thawte¢Personal¢Basic¢CA.OU=Certifi
002313 1560 00.personal-basic@thawte.com.CN=Thawte¢Personal¢Basic¢CA.OU=Certifi
002314 0561 00.personal-basic@thawte.com.CN=Thawte¢Personal¢Basic¢CA.OU=Certifi
```

Figure 9-2 Sample output of IRRDBU00 (column 1 - 72)

Table 9-2 shows the information that record type 1560 contains.

Table 9-2 General resource certificate information record

Field name	Type	Start position	End position	Comments
CERTN_RECORD_TYPE	Int	1	4	Record type of the General Resource Certificate Information Record (1560)
CERTN_NAME	Char	6	251	General resource name as taken from the profile name
CERTN_CLASS_NAME	Char	253	260	Name of the class to which the general resource profile belongs
CERTN_ISSUER_DN	Char	262	1285	Issuer’s distinguished name
CERTN_ISSUER_DN	Char	1287	2310	Subject’s distinguished name
CERTN_ISSUER_DN	Char	2312	2327	Certificate signature algorithm

Using the database Unload Utility output with IBM DB2

The samples for using the database unload utility output with IBM DB2® (RACDBULD in sys1.samplib) are updated to support the added 1560 record type. The table name of the record is GENR_CERTN_DATA.

See “Using the Database Unload Utility Output Effectively” in the *Security Server RACF Security Administrator’s Guide*, SA23-2289.

9.3 RACF Remote Sharing Facility TCP/IP support for TLS V1.2

The RACF Remote Sharing Facility (RRSF) allows RACF to communicate with other z/OS systems that use RACF so that you can maintain remote RACF databases. RRSF extends the RACF operating environment beyond the single system and shared DASD environments to an environment made of RRSF nodes that can communicate with each other. This support enables administration of multiple RACF databases from anywhere in the RRSF network, user ID associations, and automatic synchronization of RACF databases. The RRSF network can be established by using Advanced Program-to-Program Communication (APPC) or by using TCP/IP.

When you use TCP/IP for RRSF, Application Transparent Transport Layer Security (AT-TLS) is used to encrypt data between each RRSF connection. Starting with z/OS 2.1, RRSF supports the use of more cryptography suites that are available with AT-TLS, including the use of digital certificates based on Elliptic Curve Cryptography (ECC) for establishing AT-TLS sessions.

All cryptography suites in Transport Level Security (TLS) protocol Version 1.2 are supported in this release. ECC provides more secure encryption than others of the same key length. For example, the RSA 2048-bit key strength is equivalent to a 224-bit ECC key.

When a connection is established between two RRSF systems, the algorithm that is used to secure the connection is displayed in message IRRI027I. Figure 9-3 shows the informational message that is issued when RACF establishes the connection between two RRSF systems. In this example, `TLS_ECDH_ECDSA_WITH_AES_256_CBC_ SHA384` is the name of the one of the supported cipher suites.

```
IRRI027I (>) RACF COMMUNICATION WITH TCP NODE NODE1 HAS
BEEN SUCCESSFULLY ESTABLISHED USING CIPHER ALGORITHM
C026 TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384.
```

Figure 9-3 Message that confirms the RRSF connection and states the algorithm used

9.3.1 Invocation and use

To use ECC or any supported cipher suite for AT-TLS, RACF RRSF does not require configuration changes. However, the AT-TLS policy configuration for the RRSF server and RRSF client rules (including possible generation of new certificates) must be made if you require use of a newly supported cipher suite.

9.3.2 Interactions and dependencies

General AT-TLS policy configuration is necessary, as previously mentioned.

9.4 RRSF TCP/IP support for IPv6

In z/OS 2.1, the use of the IPv6 protocol for RRSF connections is available for communications between systems that are enabled for TCP/IP IPv6. You can choose between IPv4 and IPv6 addressing when setting up RRSF connections over TCP/IP.

9.4.1 Invocation and use

The network must be configured to use IPv6 as a prerequisite. To use IPv6 with RRSF, specify an IPv6 address in the ADDRESS operand in the TARGET command that defines an RRSF node. Example 9-1 shows an example of the TARGET command syntax for a remote node.

Example 9-1 TARGET command syntax for a remote node

```
TARGET NODE(nodename) PROTOCOL(TCP(ADDRESS(host-name)))-
PREFIX(qualifier) WORKSPACE(VOLUME(volser))OPERATIVE
```

For the ADDRESS operand, you can now specify either an IPv4 or an IPv6 address (or a host name that resolves to an IPv4 or IPv6 address). Example 9-2 shows a TARGET command with an IPV6 address.

Example 9-2 IPV6 address

```
TARGET NODE(NODE2)-
   PROTOCOL(TCP(ADDRESS(2001:0db8:85a3:0000:0000:8a2e:0370:7334))) -
   PREFIX(RRSJ.BRUCE) WORKSPACE(VOLUME(TEMP01)) OPERATIVE
```

You receive a message, IRRI027I, if the connection is established successfully. In addition, the **TARGET LIST** command displays resolved IPv6 addresses, where possible. See Figure 9-4 on page 125 for an example of the **TARGET LIST** output.

Note: If IPv6 is enabled, all addresses, including IPv4 addresses, are displayed in IPv6 format.

```
NODE1 <TARGET LIST NODE(NODE2)
NODE1 IRRM010I (<) RSWJ SUBSYSTEM PROPERTIES OF REMOTE RRSF NODE NODE2:
STATE - OPERATIVE ACTIVE
DESCRIPTION - <NOT SPECIFIED>
PROTOCOL - TCP
HOST ADDRESS - ALPS4012.POK.IBM.COM
IP ADDRESS - 2001:0db8:85a3:0000:0000:8a2e:0370:7334
LISTENER PORT - 18136
AT-TLS POLICY:
RULE_NAME - RRSF-CLIENT
CIPHER ALG - 35 TLS_RSA_WITH_AES_256_CBC_SHA
CLIENT AUTH - REQUIRED
TIME OF LAST TRANSMISSION TO - <NONE>
TIME OF LAST TRANSMISSION FROM - <NONE>
WORKSPACE FILE SPECIFICATION
PREFIX - "RSFJ.BRUCE"
WDSQUAL - <NOT SPECIFIED>
FILESIZE - 500
VOLUME - TEMP01
FILE USAGE
"RSFJ.BRUCE.NODE1.NODE2.INMSG"
- CONTAINS 0 RECORD(S)
- OCCUPIES 1 EXTENT(S)
"RSFJ.BRUCE.NODE1.NODE2.OUTMSG"
- CONTAINS 0 RECORD(S)
- OCCUPIES 1 EXTENT(S)
```

Figure 9-4 TARGET LIST command sample output

9.4.2 Interactions and dependencies

General network configuration to use IPv6 is required, as mentioned previously.

9.4.3 Migration and coexistence considerations

If you are migrating your system from a previous z/OS release that had IPv6 enabled, then immediately upon installation of z/OS 2.1, the **TARGET LIST** output may display addresses in the IPv6 address format. Even when IPv4 addresses are specified in the **TARGET** commands, the **TARGET LIST** detailed output for an RRSF node displays addresses in the IPv6 address format.

9.5 Comments and blank lines now allowed in the RACF parameter library

In previous releases, a blank line or a whole-line comment in the RACF parameter library caused an error message when the commands were issued.

A blank line would cause an error with this message:

```
IRRC003I (@) COMMAND ???????? IS NOT VALID.
```

Whole-line comment would cause an error with this message:

```
IRRC003I (@) COMMAND // THIS IS MY COMMENT IS NOT VALID.
```

In z/OS 2.1, blank lines and whole-line-comments are allowed. This allows for RACF parameter library members that contain **SET** and **TARGET** statements that define RRSF systems to be fully documented with comments.

9.5.1 Invocation and use

This enhancement allows the RACF parameter library to be easily documented with comment statements and allows the addition of blank lines for clearer formatting. The following rules apply:

- A whole-line comment is any line that starts with //, in any column.
- Specifying a continuation character in a whole-line comment has no effect. It is treated as part of the comment.
- A whole-line comment or blank line cannot be specified within a continued command.
- If the RACF parameter library is shared among down-level systems, error messages are issued on the down-level systems for blank lines and whole-line comments.

In Figure 9-5, all three cases are treated as comments.

```
//This is a comment line
// This is a comment line //
                // This is a comment line
```

Figure 9-5 Sample comments

The examples that are shown in Figure 9-5 are incorrect whole-line comments

```
//Example 1;
// This is a comment, trailing dash ignored -
This line is treated as a new command, not a continuation of above comment.

//Example 2;
TARGET LIST // This is not a valid comment and will fail.

Example 3;
TARGET -
// This will be treated as part of the TARGET command and will fail
LISTPROTOCOL
```

Figure 9-6 Sample incorrect comments

Figure 9-7 on page 127 shows a RACF parameter library member with blank lines and comments.

```
//*************************************************************/
// THIS MEMBER DEFINES 1 MULTI-SYSTEM NODE AND 1 SINGLE SYSTEM NODE
//
// NODE1/NODE1    LOCAL NODE1 MAIN
// NODE1/NODE2SY1 LOCAL NODE1 NON-MAIN
// NODE2          REMOTE SSN
//*************************************************************/

// THE LOCAL MSN */
TARGET NODE(NODE1) SYSNAME(NODE1) MAIN                              -
       PREFIX(RRSF.FILE) PROTOCOL(APPC(LUNAME(MF1APP01)))           -
       WORKSPACE(VOLUME(TEMP01) FILESIZE(500)) LOCAL OPERATIVE

TARGET NODE(NODE1) SYSNAME(NODE2SY1)                                -
       PREFIX(RRSF.FILE) PROTOCOL(APPC(LUNAME(MF2APP01)))           -
       WORKSPACE(VOLUME(TEMP01) FILESIZE(500)) LOCAL OPERATIVE

// THE REMOTE SSN */
TARGET NODE(NODE2)                                                  -
       PREFIX(RRSFRACF) PROTOCOL(APPC(LUNAME(MF3APP01)))            -
       WORKSPACE(VOLUME(TEMP02) FILESIZE(500))       OPERATIVE
```

Figure 9-7 Comments and blank lines in RACF parameter library member

9.5.2 Migration and coexistence considerations

If the RACF parameter library is shared among down-level systems, error messages are issued on the down-level systems for blank lines and whole-line comments

9.6 Enterprise PKCS#11 support

In previous z/OS Integrated Cryptographic Service Facility (ICSF) releases, ICSF PKCS#11 support was only for clear keys, including those in the Token Key Data Set (TKDS).

Note: “Clear key” represents a key that is not encrypted, and “secure key” represents an encrypted key with a native value that is never exposed outside of a tamper-proof device, such as the Crypto Express 4S card.

ICSF HCR77A0 introduced a function to support secure keys in the TKDS. The **RACDCERT GENCERT** command is enhanced to create a secure key for a specified PKCS#11 token in the TKDS during digital certificate creation. An Enterprise PKCS#11 coprocessor, with an active master key, must be available to generate and use secure PKCS#11 keys. In addition, a TKDS is required to use secure key PKCS#11.

On the zEnterprise EC12, a mode is available when defining the Crypto Express4S feature as a coprocessor. This mode is called the Secure IBM Enterprise Public-key Cryptography Standards (PKCS) #11 (EP11), and it provides open industry-standard cryptographic services. The EP11 coprocessor implementation is based on PKCS #11 specification 2.20, which is one of the specifications that are produced by RSA Laboratories.

With PKCS#11, you can generate and use digital certificates with hardware-protected keys in the TKDS. It provides better security on the key generation capability when using PKI Services.

9.6.1 Invocation and use

To use PKCS#11 with the **RACDCERT GENCERT** and **RACDCERT REKEY** commands, a sub-keyword **TOKEN** is added to indicate the generation of a secure TKDS key.

When using the **RACDCERT GENCERT** command to create a certificate with a secure key in the TKDS, the token value that is specified with the TOKEN keyword must exist. The token can be added by using the **RACDCERT ADDTOKEN** command.

These are two examples of using **RACDCERT GENCERT**:

- Generate a digital certificate with an RSA key stored in a token called MY.PKCS11.TOKEN1 in the TKDS:

```
RACDCERT GENCERT SUB(CN('Company A')) WITHLABEL('New RSA cert')-
RSA(TOKEN(MY.PKCS11.TOKEN1))
```

- Replicate (rekey) a digital certificate with an NISTECC key stored in a token called MY.PKCS11.TOKEN2 in the TKDS:

```
RACDCERT REKEY LABEL('New ECC cert') NISTECC(TOKEN(MY.PKCS11.TOKEN2))
```

The RACF panels for GENCERT and REKEY are updated to handle the support. Figure 9-8 on page 129 shows the modified GENCERT panel.

```
RACF - Generate a Digital Certificate
COMMAND ===>

                                                                More:   - +
 Select the key type to be generated:

    _  RSA(default)
    _  RSA in PKDS with an optional PKDS label or *:
       ____________________________________________________________________
    _  RSA in specified TKDS token:
       ________________________________
    _  RSA Modulus-Exponent in PKDS with an optional PKDS label or *:
       ____________________________________________________________________
    _  DSA
    _  NIST ECC
    _  NIST ECC in PKDS with an optional PKDS label or *:
       ____________________________________________________________________
    _  NIST ECC in specified TKDS token:
       ________________________________
    _  Brainpool ECC
    _  Brainpool ECC in PKDS with an optional PKDS label or *:
       ____________________________________________________________________
    _  Brainpool ECC in specified TKDS token:
       ________________________________



 Enter when the certificate is valid:

    NOTBEFORE Date: YYYY-MM-DD  (Default is current local date)
    NOTBEFORE Time: HH:MM:SS    (Default is 00:00:00)
```

Figure 9-8 RACF “Generate a Digital Certificate” panel

R_datalib service

PKCS#11 support is also available for the **R_datalib** service. The **R_datalib** service provides the function that is required to implement the OCSF (Open Cryptographic Services Facility) Data library functions. It uses RACF key rings and z/OS PKCS #11 tokens.

Private key types are added to the service. These key types are usable with the **DataGetFirst** function and the **DataGetNext** functions. **DataGetFirst** locates and returns the first trusted certificate in the ring, and **DataGetNext** locates and returns the next trusted certificate in the ring. These are the private key types:

X'0000000B'	RSA key token label in the TKDS
X'0000000D'	ECC key token label in the TKDS
X'0000000E'	DSA key token label in the TKDS

For the existing private key type, **X'00000002'** (ICSF key token label), the first character is an equals (=) sign if it is a key token from the TKDS. Otherwise, it is from the PKDS. In previous releases, **X'00000002'** was only for a key stored in the PKDS.

See *Security Server RACF Callable Services*, SA23-2293 for more information.

9.6.2 Interactions and dependencies

ICSF HCR77A0 is necessary, as previously mentioned. You can download it from the z/OS downloads web page:

http://www.ibm.com/systems/z/os/zos/tools/downloads/

Also, a Crypto Express4S in EP11 mode on an IBM zEnterprise server is required. Therefore, a Trusted Key Entry 7.2 workstation is also required.

9.6.3 Migration and coexistence considerations

In addition to the existing **SO.<token-label>** and **USER.<token-label>** resources in the SAF CRYPTOZ class, two additional resources for controlling access to tokens are introduced:

- The **CLEARKEY.<token-name>** resource is queried to determine the policy for creating a clear, in contrast to a secure key when CKA_IBM_SECURE=TRUE has not been specified for key generation.
- The **FIPSEXEMPT.<token-name>** resource is used for identifying applications that are subject to FIPS 140 restrictions when ICSF is running in FIPS compatibility mode.

Avoid setting a discrete or generic profile that restricts clear key creation in the omnipresent token. This token is used by other z/OS components to create session keys only. It is typical for session keys to be clear keys. The clear key resource that is checked for the omnipresent token is **CLEARKEY.SYSTOK-SESSIONONLY**.

See *Writing PKCS #11 Applications*, SA-2231-05 for details about these CRYPTOZ class resources.

9.7 RACDCERT displays additional information for digital certificate chains

When importing a digital certificate chain (such as PKCS#12 binary certificate package or PKCS#7 binary certificate package) by using the **RACDCERT ADD** command, only the end entity certificate can be named by using the specified label. RACF generates labels for rest of the digital certificates in the chain, but the generated labels for imported certificates other than the end entity are not displayed. In addition, there was previously no support in the **RACDCERT LIST** or the **RACDCERT CHECKCERT** command to display the certificate chain information. These commands list only certificate information of a single certificate.

In z/OS 2.1, the labels that are used for all the certificates in the chain are reported when the chain is added by using **RACDCERT ADD**. Information about the certificates in the chain is also provided for the **RACDCERT CHECKCERT** command. In addition, a command is added, **RACDCERT LISTCHAIN**, to display a certificate and any related issuer certificates. The **RACDCERT LIST**, **RACDCERT CHECKCERT**, and **RACDCERT LISTCHAIN** commands now show the signing algorithms for each listed certificate.

9.7.1 Invocation and use

This section covers the enhancements to the **RACDCERT** command.

RACDCERT ADD

When the **RACDCERT ADD** command successfully adds a chain of certificates, a new message is issued for each added certificate to indicate the label that is used.

For example, issuing the following command:

```
RACDCERT ID(ZOSUPD2) ADD(<dataset.contains.the.chain>) WITHLABEL('Test Cert')
```

This command issues messages similar to those shown in Figure 9-9. The message ID is IRRD199I.

```
Certificate with label 'Test Cert' is added under ID ZOSUPD2
Certificate with label 'LABEL00000002' is added under CERTAUTH
Certificate with label 'LABEL00000003' is added under CERTAUTH
```

Figure 9-9 Sample IRRD199I messages

RACDCERT LISTCHAIN

To list the certificate chain, the **RACDCERT LISTCHAIN** command can be used.

The **RACDCERT LISTCHAIN** command was introduced with z/OS 2.1. The command displays digital certificate information about a certificate that is owned by an ID, SITE, or CERTAUTH and its issuer's certificates that are owned by CERTAUTH in a chain of certificates.

The **RACDCERT LISTCHAIN** command generates **RACDCERT LIST** output of individual certificates, with the end entity certificate listed first, followed by zero or more issuer certificates. Additional information is displayed in a Chain Information section at the end of the command display output. The list output can contain the following information:

- The number of certificates in the chain
- Whether the RACF database contains the complete chain
- Indication that the chain contains a certificate that has the NOTRUST status or is expired
- Whether the certificates in the chain have a key ring in common

RACDCERT LISTCHAIN is used to list certificates in RACF database. For example:

```
RACDCERT ID(ZOSUPD2) LISTCHAIN(LABEL('ZOSUPD2 TEST CERTICATE'))
```

This generates output that is similar to Example 9-3.

Example 9-3 Sample output for RACDCERT LISTCHAIN command

```
READY
  RACDCERT ID(ZOSUPD2) LISTCHAIN(LABEL('ZOSUPD2 TEST CERTICATE'))
 Certificate 1:
 Digital certificate information for user ZOSUPD2:

   Label: ZOSUPD2 TEST CERTICATE
   Certificate ID: 2Qfp1uLk18Ty6dbi5NfE8kDjxeLjQMPF2ePJw8HjxUBA
   Status: TRUST
   Start Date: 2013/05/03 00:00:00
   End Date:   2013/12/31 23:59:59
   Serial Number:
        >02<
   Issuer's Name:
        >CN=ZOSUPD2 TEST CERTAUTH2.T=V2R1 TEST CERTAUTH1.OU=IBM CORPORATION.O=<

        >IBM.C=JP<
```

```
  Subject's Name:
       >CN=ZOSUPD2 TEST CERTICATE.T=V2R1 TEST CERTIFICATE.OU=IBM CORPORATION.O=<

       >IBM.C=JP<
  Signing Algorithm: sha1DSA
  Key Type: DSA
  Key Size: 1024
  Private Key: YES
  Ring Associations:
  *** No rings associated ***

Certificate 2:
Digital certificate information for CERTAUTH:

  Label: ZOSUPD2 TEST CERTAUTH2
  Certificate ID: 2Qfp1uLk18Ty6dbi5NfE8kDjxeLjQMPF2ePJw8HjxUBA
  Status: TRUST
  Start Date: 2013/05/03 00:00:00
  End Date:   2013/12/31 23:59:59
  Serial Number:
       >01<
  Issuer's Name:
       >CN=ZOSUPD2 TEST CERTAUTH1.T=V2R1 TEST CERTAUTH1.OU=IBM CORPORATION.O=<

       >IBM.C=JP<
  Subject's Name:
       >CN=ZOSUPD2 TEST CERTAUTH2.T=V2R1 TEST CERTAUTH2.OU=IBM CORPORATION.O=<

       >IBM.C=JP<
  Signing Algorithm: sha1DSA
  Key Usage: CERTSIGN
  Key Type: DSA
  Key Size: 1024
  Private Key: YES
  Ring Associations:
  *** No rings associated ***

Certificate 3:
Digital certificate information for CERTAUTH:

  Label: ZOSUPD2 TEST CERTAUTH1
  Certificate ID: 2QiJmZmDhZmjgenW4uTXxPJA48Xi4ODDxdnjweTjyPJA
  Status: TRUST
  Start Date: 2013/05/03 00:00:00
  End Date:   2013/12/31 23:59:59
  Serial Number:
       >00<
  Issuer's Name:
       >CN=ZOSUPD2 TEST CERTAUTH1.T=V2R1 TEST CERTAUTH1.OU=IBM CORPORATION.O=<

       >IBM.C=JP<
  Subject's Name:
       >CN=ZOSUPD2 TEST CERTAUTH1.T=V2R1 TEST CERTAUTH1.OU=IBM CORPORATION.O=<

       >IBM.C=JP<
```

```
  Signing Algorithm: sha1DSA
  Key Usage: CERTSIGN
  Key Type: DSA
  Key Size: 1024
  Private Key: YES
  Ring Associations:
  *** No rings associated ***

Chain information:
  Chain contains 3 certificate(s), chain is complete
  Chain contains no ring in common
READY
END
```

Support for the **RACDCERT LISTCHAIN** command is added to the RACF ISPF panels. The RACF - Digital Certificate Services Main Panel now has another choice (option number 5) to list a certificate with its chain of issuers' certificates, as shown in Figure 9-10.

```
 RACF - Digital Certificate Services Main Panel          Top of data
 OPTION ===>


                      Personal              Certificate
                     (user ID)      Site     Authority
 For Certificate Type  ________  or  _    or  _

   Select one of the following options:

   1. Add a digital certificate to the RACF database.

   2. List information for certificates and be given the
      opportunity to change or delete them.

   3. List a certificate using a filter and be given the opportunity
      to change the trust status, label, or delete it.

   4. Check whether the digital certificates in the specified
      dataset have been added to the RACF database and associated
      with a user ID, by entering a data set name:
       ____________________________________________

      Password for PKCS12 format data set(in quotes):
      =>                                                          <=
      =>                                                          <=
      =>                                                          <=
      =>                                                          <=

   5. List a certificate with its chain of issuers' certificates by
      entering the label name (in quotes):
       ___________________________________
```

Figure 9-10 RACF panel update

RACDCERT CHECKCERT

The **RACDCERT CHECKCERT** command is used to list certificates in a data set. It is enhanced in the following ways:

- If the certificate is already in the RACF database and the user is authorized to list it, the output looks like the **RACDCERT LISTCHAIN** output, except for the private key and ring information.
- If the certificate is not in the RACF database or the user is not authorized, the output does not show the RACF-related information.
- If there is no error encountered, the certificates are displayed with the end-entity certificate listed first, followed by the subsequent issuers', and this information about the chain:
  - The number of certificates in the chain.
  - Indication of whether the data set contains the complete chain.
  - Indication whether any of the certificates in the chain are expired.

For example, the following command generates the output shown in Example 9-4:

```
RACDCERT CHECKCERT('ZOSUPD2.CERTEXPT')
```

Example 9-4 RACDCERT CHECKCERT command output

```
READY
 RACDCERT CHECKCERT('ZOSUPD2.CERTEXPT')
 Certificate 1:
 Digital certificate information for user ZOSUPD2:

   Label: ZOSUPD2 TEST CERTICATE
   Certificate ID: 2Qfp1uLk18Ty6dbi5NfE8kDjxeLjQMPF2ePJw8HjxUBA
   Status: TRUST
   Start Date: 2013/05/03 00:00:00
   End Date:   2013/12/31 23:59:59
   Serial Number:
        >02<
   Issuer's Name:
        >CN=ZOSUPD2 TEST CERTAUTH2.T=V2R1 TEST CERTAUTH1.OU=IBM CORPORATION.O=<

        >IBM.C=JP<
   Subject's Name:
        >CN=ZOSUPD2 TEST CERTICATE.T=V2R1 TEST CERTIFICATE.OU=IBM CORPORATION.O=<

        >IBM.C=JP<
   Signing Algorithm: sha1DSA
   Key Type: DSA
   Key Size: 1024
   Private Key: YES

 Certificate 2:
 Digital certificate information for CERTAUTH:

   Label: ZOSUPD2 TEST CERTAUTH2
   Certificate ID: 2QiJmZmDhZmjgenW4uTXxPJA48Xi4ODDxdnjweTjyPJA
   Status: TRUST
   Start Date: 2013/05/03 00:00:00
   End Date:   2013/12/31 23:59:59
   Serial Number:
```

```
       >01<
  Issuer's Name:
       >CN=ZOSUPD2 TEST CERTAUTH1.T=V2R1 TEST CERTAUTH1.OU=IBM CORPORATION.O=<

       >IBM.C=JP<
  Subject's Name:
       >CN=ZOSUPD2 TEST CERTAUTH2.T=V2R1 TEST CERTAUTH2.OU=IBM CORPORATION.O=<

       >IBM.C=JP<
  Signing Algorithm: sha1DSA
  Key Usage: CERTSIGN
  Key Type: DSA
  Key Size: 1024
  Private Key: YES

Certificate 3:
Digital certificate information for CERTAUTH:

  Label: ZOSUPD2 TEST CERTAUTH1
  Certificate ID: 2QiJmZmDhZmjgenW4uTXxPJA48Xi4ODDxdnjweTjyPFA
  Status: TRUST
  Start Date: 2013/05/03 00:00:00
  End Date:   2013/12/31 23:59:59
  Serial Number:
       >00<
  Issuer's Name:
       >CN=ZOSUPD2 TEST CERTAUTH1.T=V2R1 TEST CERTAUTH1.OU=IBM CORPORATION.O=<

       >IBM.C=JP<
  Subject's Name:
       >CN=ZOSUPD2 TEST CERTAUTH1.T=V2R1 TEST CERTAUTH1.OU=IBM CORPORATION.O=<

       >IBM.C=JP<
  Signing Algorithm: sha1DSA
  Key Usage: CERTSIGN
  Key Type: DSA
  Key Size: 1024
  Private Key: YES

Chain information:
  Chain contains 3 certificate(s), chain is complete
READY
END
```

Addition of Signing Algorithm field in the output display

The output of **RACDCERT LIST**, **RACDCERT LISTCHAIN**, and **RACDCERT CHECKCERT** is also changed in z/OS 2.1. A Signing Algorithm output field has been added. Figure 9-11 on page 136 shows sample output of a **RACDCERT LIST** command, showing the added output field.

```
 RACDCERT CERTAUTH LIST(LABEL('V2R1 RACF TEST CERTIFICATE'))

Digital certificate information for CERTAUTH:

  Label: V2R1 RACF TEST CERTIFICATE
  Certificate ID: 2QiJmZmDhZmjgeXy2fFA2cHDxkDjxeLjQMPF2ePJxsnDwePF
  Status: TRUST
  Start Date: 2013/04/17 00:00:00
  End Date:   2013/12/31 23:59:59
  Serial Number:
       >00<
  Issuer's Name:
       >CN=V2R1.RACFTEST.IBM.COM.T=V2R1 TEST CERTIFICATE.OU=IBM CORPORATION.O<

       >=IBM.C=JP<
  Subject's Name:
       >CN=V2R1.RACFTEST.IBM.COM.T=V2R1 TEST CERTIFICATE.OU=IBM CORPORATION.O<

       >=IBM.C=JP<
  Signing Algorithm: sha1DSA
  Key Usage: CERTSIGN
  Key Type: DSA
  Key Size: 1024
  Private Key: YES
  Ring Associations:
  *** No rings associated ***
```

Figure 9-11 Sample RACDCERT LIST command

9.8 Prevent deletion of a certificate with a pending request

The **RACDCERT GENREQ** command is used to create a certificate request (certificate signing request), based on an existing certificate that has a private key associated with it in the RACF database. The certificate that is being sent to the CA for signing does not contain its private key. Upon completion of the signing request, the certificate authority (CA) signed certificate is imported back into the RACF database, replacing the existing certificate but retaining the existing certificates private key, which is then associated with the CA signed certificate.

If the original certificate used to generate the certificate signing request is deleted before the signed certificate is imported back into the RACF database, the returned CA signed certificate is useless. This is because the certificate's private key is associated with the original certificate is not included in the certificate signing request. Therefore, the deletion of the original certificate also deletes the certificates private key.

For example, when generating a certificate that is to be signed by a CA, complete these steps as a part of the procedure:

1. Using the **RACDCERT GENCERT** command, create a self-signed certificate, and name it (this example uses `CERTA` as the name).
2. Using the **RACDCERT GENREQ** command, create a certificate signing request (CSR) data set (named `CERTA.CSR`, for example), specifying `CERTA`. At this point, CERTA is not deleted and remains in the RACF database (including the private key).
3. Send CERTA.CSR (no private key) to the certificate authority for signing.

4. Using the **RACDCERT ADD** command, import the CERTA.CSR that is signed and returned by the CA to the RACF database, and replace the existing CERTA certificate (retaining original certificates private key).

The CERTA certificate in the RACF database contains a private key, and CERTA.CSR does not. In step 4, CERTA is replaced by the CA-signed CERTA.CSR, but the private key is not deleted. Instead, it is associated with the imported CA-signed CERTA.CSR. With this procedure, you can generate a CA-signed certificate without exporting its secret key from the RACF database.

In the past, after the CSR request was generated, there were some cases where the original self-signed certificate was accidentally deleted. In this example, CERTA was deleted between step 2 and 4. Because the private key of the certificate is associated with the original self-signed certificate, deleting the certificate also deletes the private key. The returned CA-signed certificate can still be imported, but it is useless, because it contains only the public key.

z/OS Version 2.1 prevents the deletion of a self-signed certificate that has been used for creating a CSR. However, you have the option to force a deletion.

9.8.1 Invocation and use

An attempt to delete a self-signed certificate that is used for generating a CSR data set fails if it is performed by using the **RACDCERT DELETE** command before it is replaced by the certificate that is returned by the CA. Instead, you receive the message that is shown in Figure 9-12.

```
 RACDCERT ID(ZOSUPD2)   DELETE(LABEL('V2R1 RACF TEST'))
The certificate has been used for generating a request. It was not deleted.
```

Figure 9-12 Sample output when a self-signed certificate was not deleted

If you want to delete a certificate that was used to generate a CSR, use the **FORCE** option. Example 9-5 shows a sample command.

Example 9-5 Sample command for a RACDCERT DELETE FORCE option

```
RACDCERT ID(ZOSUPD2)   DELETE(LABEL('V2R1 RACF TEST')) FORCE
```

This additional protection also applies to the **RACDCERT ROLLOVER** command. If a user attempts to use the **RACDCERT ROLLOVER** command against a certificate with a pending CSR, the command fails and the message in Example 9-6 is issued.

Example 9-6 Response to using RADCERT ROLLOVER for a certificate with a pending CSR

```
IRRD198I The certificate has been used for generating a request. It was not
superseded.
```

However, a **FORCE** option can be added to the **RACDCERT ROLLOVER** command to force the rollover and bypass the error.

9.9 Documentation and reference

For more information, see the following publications and web pages:

- *z/OS V2R1 Communications Server: IPv6 Network and Application Design Guide*, SC27-3663

  https://ibm.biz/BdFx8S
- *z/OS Security Server RACF Callable Services,* SA23-2293

  https://ibm.biz/BdFx8v
- *z/OS Security Server RACF Command Language Reference*, SA23-2292

  https://ibm.biz/BdFx8m
- *z/OS Security Server RACF Macros and Interfaces*, SA23-2288

  https://ibm.biz/BdFx8n
- *z/OS Security Server RACF Security Administrator's Guide*, SA23-2289

  https://ibm.biz/BdFx8b
- Using the z/OS Integrated Cryptographic Service Facility (ICSF)

  https://ibm.biz/BdFxqE
- *Writing PKCS #11 Applications*, SC14-7510

  https://ibm.biz/BdFx8A
- *z/OS Hot Topics* article: *Nobody's deFault but mine*, ISSUE26 AUGUST 2012

  https://ibm.biz/BdFx8C

10

PKI Services

This chapter describes the PKI Services enhancements, functions, and features that were introduced with IBM z/OS Version V2R1 (2.1):

- 10.1, “Enterprise PKCS #11 support” on page 140
- 10.2, “Extended Validation Certificate support” on page 141
- 10.3, “Granular certificate administration control” on page 142
- 10.4, “Enhancement for Basic Constraints extension” on page 144
- 10.5, “IBM HTTP Server V7.0 support” on page 145
- 10.6, “More about Certificate Revocation List processing” on page 146
- 10.7, “Documentation and reference” on page 147

PKI Services enables you to use z/OS to establish the Public Key Infrastructure (PKI) and serve as a certificate authority for your internal and external users, issuing and administering digital certificates in accordance with your own organization’s policies. PKI applications can be used to request and obtain digital certificate through their own web browsers. Your authorized PKI administrators approve, modify, or reject these requests through their own browsers. For basic information about PKI Services, see *Cryptographic Services PKI Services Guide and Reference*, SA23-2286 (see 10.7, “Documentation and reference” on page 147 for a link).

PKI provides applications with a framework for performing the following types of security-related activities:

- Authenticate all parties that engage in electronic transactions.
- Authorize access to sensitive systems and repositories.
- Verify the author of each message through its digital signature.
- Encrypt the content of all communications.

© Copyright IBM Corp. 2014. All rights reserved.

10.1 Enterprise PKCS #11 support

PKI Services now supports IBM Enterprise Public-key Cryptography Standards (PKCS) #11 (EP11). This enhancement uses the updates to support secure keys in the TKDS made to RACF, System SSL, and ICSF (HCR77A0).

Starting with z/OS 2.1, PKI Services can create a secure key in the Token Key Data Set (TKDS) during certificate creation. The requester receives a PKCS#12 package that contains the secure key through (or updated) System SSL APIs, which envelop the secure key to be exported. This provides increased security for the key-generation capability in PKI Services.

Note: RACDCERT EXPORT cannot export a secure key, either from PKDS or TKDS.

10.1.1 Invocation and use

You can use this enhancement for these actions:

- Enable PKI Services certificate authority (CA) to have its private key stored as a secure key in the TKDS.
- Enable PKI Services to create secure keys in the TKDS.
- Generate secure key pairs through the Certificate Management Protocol (CMP).

Enable PKI Services CA to have a secure key in the TKDS

A REXX script, **IKYSETUP**, is provided in SAMPLIB. The **IKEYSETUP** script is used for the RACF administration tasks for setting up PKI Services. Update the script to choose the key type to generate the CA certificate. The following key types are added to generate a secure key for the CA's private key:

key_type=8	Generate the Rivest-Shamir-Adleman (RSA) algorithm key by using the Enterprise PKCS#11 Cryptographic Coprocessor and store the key in the ICSF Token Key Data Set (TKDS).
key_type=9	Generate the NIST Elliptic Curve Cryptography (ECC) key by using the Enterprise PKCS#11 Cryptographic Coprocessor, and store the key in the ICSF Token Key Data Set (TKDS).
key_type=10	Generate the Brainpool Elliptic Curve Cryptography (ECC) key by using the Enterprise PKCS#11 Cryptographic Coprocessor and store the key in the ICSF TKDS.

If no domain is specified, the key is stored in the TKDS with the token name of `pki_services_daemon_name.CATOKEN`. If a value for the *`ca_domain`* variable is specified during **IKYSETUP** processing, the key is stored as `pki_services_daemon_name.CATOKEN.ca_domain`. This token is defined during **IKEYSETUP** processing before key pair generation for key types 8, 9, and 10.

The **IKYSETUP** script is updated to verify a key size that is based on the selected key type.

Enable PKI Services to create secure keys in the TKDS

Specify `T` for the **SecureKey** entry in the PKI Services configuration file (`pkiserv.conf`) to enable secure key generation. A sample `pkiserv.conf` file is provided in `/usr/lpp/pkiserv/samples/`. Specify both under the SAF section:

SecureKey=T — T (True) indicates that secure keys are generated in the TKDS. Specify F (False) or remove this keyword, if you want to generate clear keys in the TKDS. If key generation capability is turned on (and keyword TokenName is specified), this switch is processed, otherwise it is ignored.

TokenName=token-name — Specify a token name that you want to use. SecureKey is ignored if TokenName is not specified.

Generate secure keys through the Certificate Management Protocol API

The CMP program is updated to handle creation of certificates with secure keys. The following environment variables control this function:

- **_PKISERV_CMP_SECUREKEY_domain**

  Specifies whether to generate secure keys in the TKDS. The value 1 specifies that secure keys are to be generated. The value 0 specifies that clear keys are to be generated.

- **_PKISERV_CMP_SECUREKEY_KEYENCALG_dom**

  Specifies the algorithm to envelop the private key. Valid values are AES256, AES128, and TDES. If not set, it defaults to AES256. This variable is ignored if you do not set _PKISERV_CMP_SECUREKEY_domain.

10.1.2 Interactions and dependencies

In addition to Security Server (Resource Access Control Facility, RACF), the z/OS Integrated Cryptographic Service Facility (ICSF) web deliverable #12 (FMID is HCR77A0) is needed. You can download it from the z/OS Downloads web page. Also, a Crypto Express4S in EP11 mode on a IBM zEnterprise server is necessary. That server requires a Trusted Key Entry 7.2 workstation.

10.1.3 Migration and coexistence considerations

The existing TKDS might contain clear keys. After secure key generation is configured, there can be a mixture of clear and secure keys there.

You can restrict the generation of clear keys by defining a profile that protects the CLEARKEY.token_name in the CRYPTOZ class.

If the SecureKey keyword is set to F or is not specified, PKI Services generates clear keys if profiles in the CRYPTOZ class that protect the CLEARKEY function allow clear key generation. If CLEARKEY profiles do not allow clear key generation, PKI Services generates a secure key. For example, the following RACF command prevents clear key generation on the token named PKISRVD.PKITOKEN:

```
RDEF CRYPTOZ CLEARKEY.PKISRVD.PKITOKEN UACC(NONE)
```

10.2 Extended Validation Certificate support

An Extended Validation Certificate (EV) is a certificate that is issued according to a specific set of identity verification criteria. These criteria require extensive verification of the requesting entity's identity by the certificate authority (CA).

In previous releases, the certificates created by PKI Services did not support some relative distinguished name (IBM RDN®), which is required for an EV certificate.

Starting with z/OS 2.1, RDNs are needed for an EV certificate are supported, as shown in Table 10-1. This enhancement enables PKI Services administrators to potentially issue EV certificates, which are in growing demand.

Table 10-1 RDNs added to PKI Services

Name of RDN	Required	Object Identifier (OID)[a]
businessCategory	Yes	2.5.4.15
jurisdictionOfIncorporationCountryName	Yes	1.3.6.1.4.1.311.60.2.1.3
jurisdictionOfIncorporationStateOrProvinceName	No	1.3.6.1.4.1.311.60.2.1.2
jurisdictionOfIncorporationLocalityName	No	1.3.6.1.4.1.311.60.2.1.1

a. An identifier that is used to name an object.

10.2.1 Invocation and use

Update the PKI Services configuration file (`pkiserv.conf`). Specify the EV certificate RDNs in the OID section:

```
BUSINESSCATEGORY=2.5.4.15
JURISDICTIONCOUNTRY=1.3.6.1.4.1.311.60.2.1.3
JURISDICTIONSTATEPROV=1.3.6.1.4.1.311.60.2.1.2
JURISTDICTIONLOCALITY=1.3.6.1.4.1.311.60.2.1.1
```

Use the template, “2-Year EV SSL Server Certificate,” in PKI Services template files (`/usr/lpp/pkiserv/samples/pkiserv.tmpl` or `/usr/lpp/pkiserv/samples/pkitmpl.xml`). When an EV certificate is required, choose the 2-Year EV SSL Server Certificate as the certificate template to use when you are requesting a certificate from the Certificate Administration web page.

10.2.2 Interaction and dependencies

Support for EV certificate RDNs is limited to the PKI Services daemon and the R_PKIServ callable service (with PKI Services as the provider). **`RACDCERT`** does not generate certificates with EV certificate RDNs, but the `ADD`, `LIST`, and `CHECKCERT` options of **`RACDCERT`** work with certificates that contain EV RDNs. The **`GENCERT`** and **`EXPORT R_PKIServ`** services that use the z/OS System Authorization Facility (SAF) as the provider do not support EV certificate RDNs.

10.3 Granular certificate administration control

In z/OS 2.1, segregation of administration inside the same PKI instance is available. For example, you can control which administrator can approve a server digital certificate and which administrator cannot. This enhancement enables multiple PKI Services administrators to perform different actions on different types of certificates on different domains.

10.3.1 Invocation and use

Activate granular administration authorization control on requests and certificates that are based on the domain, action, and the template. A RACF class, `PKISERV`, is created for resources that are used by different types of administration functions. If granular checking is

activated, these resources are checked in addition to the existing authority check on the administrative functions.

Complete the following steps:

1. Set up authorization for PKI.
2. Configure PKI Service.
3. Create profiles in the `PKISERV` class.

Set up authorization for PKI

A REXX script, **`IKYSETUP`**, is provided in SAMPLIB. Update the script to enable granular certificate administration control, and specify these conditions:

`AdminGranularControl=`	`1` to set up granular control, otherwise, `0`
`template.`*`n`* **`=`**	Template nicknames that you want to act on and assign to the groups (“n” must be numeric)
`pkigroup`*`n`*`=`	PKI Services Administrator group name (“n” must be numeric)
`pki_gid`*`n`*`=`	PKI Services Administrator group ID (“n” must be numeric)
`pkigroup`*`n`*`_mem.0=0`	Number of pkigroup members to connect (“n” must be numeric)

Configure PKI Service

Specify `T` for the AdminGranularControl entry in the PKI Services configuration file (`pkiserv.conf`) to enable granular control. A sample `pkiserv.conf` file is provided in `/usr/lpp/pkiserv/samples/`. Specify the following information under the SAF section:

`AdminGranularControl=T`	`T` (True) to indicate that granular authority control is enabled F (False) or remove this keyword to disable granular control

Create profiles in the PKISERV class

Create profiles in the PKISERV class that consist of the domain name, action, and template to restrict access to the administrative function in this form:
`ca_domain-name.action.template_nickname`.

For example:

```
MYDOMAIN.QUERYREQS.1YBSSL
MYDOMAIN.QUERYCERTS.1YBSSL
```

In this example, the administrator who has READ access to these two profiles can perform the QUERYREQS and QUERYCERTS functions with R_PKIServ-callable service (IRRSPX00). These authorizations apply to certificate requests that are created with the “1-Year PKI SSL Browser Certificate” template named “1YBSSL” in the domain named MYDOMAIN.

Note: **`QUERYREQS`** queries PKI Services for a certificate. **`QUERYCERTS`** queries PKI Services about issued certificates.

10.3.2 Migration and coexistence

By default, granular controls are disabled. To use this feature, the administrator must enable them.

When granular authorization is enabled, the `RACF PKISERV` class and its resources affect only the behavior of Version 2.1 and later versions of PKI Services. Any earlier versions of PKI

Services in the IBM Parallel Sysplex that might be sharing an IBM Resource Access Control Facility (RACF) database are not affected by these controls.

In a configuration with multiple PKI Services versions that share the RACF database, administrators have the following privileges and restrictions:

- An administrator with access to the RACF `IRR.RPKISERV.PKIADMIN` class on Version 2.1 of PKI Services requires READ or UPDATE authority on the appropriate resource in the RACF `PKISERV` class to perform the requested administration action.
- An administrator with access to the RACF `IRR.RPKISERV.PKIADMIN` class on prior versions of PKI Services can perform any administrative action on any PKI Services request, even if a resource in the `PKISERV` class would prevent that action in PKI Services V2.1.

10.4 Enhancement for Basic Constraints extension

z/OS 2.1 provides additional information in the Basic Constraints extension in an x.509 format digital certificate. The Basic Constraints extension consists of this information:

- CA indication field (whether the subject of the digital certificate is a CA required field)
- Path length constraint value (an optional field for the maximum depth of valid certification paths that include this digital certificate)

See RFC 5280 for more information about the Basic Constraints extension.

In previous releases, PKI Services created only the CA indication field, not the path length constraint value. Although it is an optional value, many IBM clients this value set to control the number of CAs that can follow.

A path length constraint identifies the maximum number of not self-signed or intermediate CA certificates that can follow this CA in a valid certification path (*certificate chain*).

- A path length of zero indicates that no non-self signed or intermediate CA certificates can follow this certificate in a valid certification path.
- Omission of the path length field from the basic constraints extension indicates that no limit is placed in the number of intermediate CA certificates that can follow in a valid certification path.

In z/OS 2.1, there is an option for PKI Services to create the path length constraint value in the Basic Constraints extension. Path length constraint value can restrict a subordinate CA from signing another subordinate CA.

10.4.1 Invocation and use

Modify the PKI Services configuration file (`pkiserv.conf`) to enable the verification of the maximum path length. A sample `pkiserv.conf` file is provided in `/usr/lpp/pkiserv/samples/`. Specify both under the Cert policy section:

`EnablePathLenConstraint=T`	`T` (True) if maximum path length verification is enabled F (False), or remove this keyword, if you want to disable it
`PathLength=1`	The maximum path lengths to be included in the Basic Constraints extension of intermediate CA certificates that are created by this CA

If the EnablePathLenConstraint keyword is specified, the following conditions must be met by the CA certificate in use by the PKI Services daemon at initialization time:

- KeyUsage extension with the keyCertSign bit set
- Basic constraint extension present with the following settings:
  - Marked critical
  - Value of CA is true
  - The value of pathLenConstratint must be within the 0 - 16 range

If these conditions are not met, the IKYS019I message shown in Example 10-1 is issued during PKI Services initialization, and PKI Services stops.

Example 10-1 IKYS019I message about CA certificate path length constraint capability

```
IKYS019I The CA certificate does not have the path length constraint capability.
```

The PathLength value that is specified must be positive and smaller than the path length constraint value of the CA certificate. Otherwise, the IKYS020I message shown in Example 10-2 is issued during PKI Services initialization, and PKI Services stops.

Example 10-2 IKYS020I message about CA path length constraint value conflict

```
IKYS020I The specified PathLength value {none | value1} conflicts with the CA path
length constraint value value2.
```

The PathLength keyword value must be in the range of 0 - 16 and must be less than the pathLenConstraint value in the Basic Constraint extension CA certificate, if it is present. Otherwise, the message IKYS021I in Example 10-3 is issued during PKI Services initialization, and PKI Services stops.

Example 10-3 Message IKYS021I, PathLength value not valid

```
IKYS021I The value that is specified for the PathLength keyword is not allowed.
```

If the PKI Services CA certificate in use has a path length constraint value of zero, which prohibits creation of subordinate or intermediate CA certificates, the IKYS022I message in Example 10-4 is issued when PKI Services is initialized.

Example 10-4 Message IKYS022I, CA cannot create intermediate certificates

```
IKYS022I This CA is restricted from creating intermediate CA certificates.
```

10.5 IBM HTTP Server V7.0 support

PKI Services uses IBM HTTP Server Version 5.3 for z/OS, which is based on the CERN web server. The PKI Services update supports the IBM HTTP Server, Version 7.0 (based on the Apache Software Foundation's Apache HTTP Server), which is available through the IBM Ported Tools for z/OS (see 10.7, "Documentation and reference" on page 147). All of the PKI functions that are provided by the previous HTTP server remain the same in the one.

10.5.1 Invocation and use

Several sample files are provided to configure the IBM HTTP Server V7.0 to work with PKI Services:

```
src/samples/ihs7/httpd.conf
src/samples/ihs7/vhost80.conf
src/samples/ihs7/vhost443.conf
src/samples/ihs7/vhost1443.conf
```

These files have the following purposes:

httpd.conf	The main configuration file. The server is configured by placing directives in plain text. More configurations files are added by using the `Include` directive.
vhost80.conf	Virtual host configuration file for non-SSL processing.
vhost443.conf	Virtual host configuration file for server authentication in SSL processing.
vhost1443.conf	Virtual host configuration file for client authentication in SSL processing.

10.6 More about Certificate Revocation List processing

The Certificate Revocation List (CRL) is a list of certificates that have been revoked for reasons other than expiration, such as improper issuance. In previous releases, no notification was issued after the CRL was processed.

In z/OS 2.1, PKI Services can optionally issue either of these console messages when CRL processing ends:

```
IKYP044I CRL number crl-serial-number processing for CA domain ca-domain completed
successfully

IKYP045I CRL number crl-serial-number processing for CA domain ca-domain failed
```

With this enhancement, a console message for CRL completion can act as a trigger for certain automated processing. For example, CRLs can be saved for either legal reasons or as a matter of policy. It can be configured to prevent unwanted console messages from being issued.

10.6.1 Invocation and use

To enable CRL notification, the `CRLWTONotification` keyword is added to the CertPolicy section of the `pkiserv.conf` file. The following are accepted values:

`None`	Indicates that no CRL processing console message will be issued. This is the default, and it behaves as in previous releases.
`File`	Indicates that CRL processing console messages will be issued after the CRLs are available in the system.

This keyword is ignored unless large CRL posting support is enabled (`EnableLargeCRLPosting=T`) or at least one HTTP protocol CRL distribution point URI is defined.

10.7 Documentation and reference

For more information, see the following publications and web pages:

- IBM Ported Tools for z/OS web page

  http://www.ibm.com/systems/z/os/zos/features/unix/port_tools.html

- RFC 5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile

  http://www.ietf.org/rfc/rfc5280.txt

- *z/OS V2R1.0 Cryptographic Services PKI Services Guide and Reference*, SA23-2286, which you can download from the z/OS V2R1 Elements and Features Library (PDF files):

  http://www.ibm.com/systems/z/os/zos/library/bkserv/v2r1pdf/

11

System SSL

This chapter describes the following IBM System SSL enhancements, functions, and features that were introduced with IBM z/OS Version 2 Release 1 (2.1):

- 11.1, “Enhancements for TLS V1.2” on page 150
- 11.2, “Enhancement for Suite B cryptography for TLS” on page 154
- 11.3, “RFC 5280 certificate validation support” on page 156
- 11.4, “SAF key ring certificate chain validation support” on page 157
- 11.5, “DSA 2048-bit key support” on page 159
- 11.6, “System SSL applications in FIPS mode using ICSF functions” on page 161
- 11.7, “Enterprise PKCS #11 support” on page 162
- 11.8, “Documentation and reference” on page 163

© Copyright IBM Corp. 2014. All rights reserved.

11.1 Enhancements for TLS V1.2

Before z/OS 2.1, System SSL supported the Secure Sockets Layer (SSL) V2 and V3, Transport Security Layer (TLS) V1.0 and V1.1 industry standard protocols. In addition to these existing protocols, System SSL is updated so that applications can specify and negotiate by using TLS Version 1.2 protocol, as defined in RFC 5246.

With the TLS 1.2 protocol, the following enhancements are available:

- AES-GCM (Galois Counter Mode) cipher suites for TLS.
- TLS Elliptical Curve cipher suites with SHA-256 or SHA-384 and AES GCM.
- SHA-256 and SHA-384 can be used for message authentication.
- MD5 and SHA-1 pseudo random function (PRF) is replaced by cipher suite-specified PRFs. The default PRF used for TLS 1.2 is an SHA-256 based PRF.

TLS 1.2 is a prerequisite for implementing Suite-B security profile.

This section covers the two main enhancement areas:

- cipher suites
- TLS handshake extension support

11.1.1 Cipher suites

z/OS 2.1 adds 37 cipher suites for use in SSL sessions. Table 11-1 shows the list of cipher suites that were added to be used with the TLS 1.2 protocol.

For more information about TLS 1.2, see RFC 5246. For a list of C cipher suites that are supported by FIPS 140-2, Base Security Level FMID HCPT410, and Security Level 3 FMID JCPT411 see *Cryptographic Services System Secure Sockets Layer Programming*, SC41-7495. (11.8, “Documentation and reference” on page 163 includes links to both.)

Note: A cipher suite is a combination of algorithms that are used when negotiating how a connection is secured.

Table 11-1 Cipher suites added to be used with the TLS 1.2 protocol

Type	Name	Cipher specification value
NULL	TLS_RSA_WITH_NULL_SHA256	003B
AES-CBC	TLS_RSA_WITH_AES_128_CBC_SHA256 TLS_RSA_WITH_AES_256_CBC_SHA256 TLS_DH_DSS_WITH_AES_128_CBC_SHA256 TLS_DH_RSA_WITH_AES_128_CBC_SHA256 TLS_DHE_DSS_WITH_AES_128_CBC_SHA256 TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 TLS_DH_DSS_WITH_AES_256_CBC_SHA256 TLS_DH_RSA_WITH_AES_256_CBC_SHA256 TLS_DHE_DSS_WITH_AES_256_CBC_SHA256 TLS_DHE_RSA_WITH_AES_256_CBC_SHA256	003C 003D 003E 003F 0040 0067 0068 0069 006A 006B

Type	Name	Cipher specification value
AES_GCM	TLS_RSA_WITH_AES_128_GCM_SHA256 TLS_RSA_WITH_AES_256_GCM_SHA384 TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 TLS_DH_RSA_WITH_AES_128_GCM_SHA256 TLS_DH_RSA_WITH_AES_256_GCM_SHA384 TLS_DHE_DSS_WITH_AES_128_GCM_SHA256 TLS_DHE_DSS_WITH_AES_256_GCM_SHA384 TLS_DH_DSS_WITH_AES_128_GCM_SHA256 TLS_DH_DSS_WITH_AES_256_GCM_SHA384	009C 009D 009E 009F 00A0 00A1 00A2 00A3 00A4 00A5
AES-GCM symmetric algorithm	TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256 TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384 TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256 TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384	C023 C024 C025 C026 C027 C028 C029 C02A
AES-GCM symmetric algorithm	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256 TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384 TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256 TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384	C02B C02C C02D C02E C02F C030 C031 C032

11.1.2 TLS handshake extension support

In addition, TLS 1.2 defines a TLS handshake extension called Signature Algorithms. This extension is used to indicate to the server which signature/hash algorithm pair can be used in digital signatures. The extension contains an enumerated list in descending order of preference of hash/signature algorithm combinations that the client supports.

Before TLS 1.2, the hash and signature algorithms that were used by the handshake processing for digital signature were limited to the values defined by the cipher that was being used. TLS 1.2 adds the flexibility to allow the application to widen or limit the hash and signature algorithms. For example, if you do not want an application to use SHA-1 when establishing a secure connection, the SHA-1 based signature algorithms are not specified for use during the TLS handshake.

With TLS handshake extension, enumerated hash algorithms and enumerated signature algorithms are represented as numeric values, as shown in Table 11-2 on page 152 and Table 11-3 on page 152.

Table 11-2 Enumerated hash algorithms and numeric values

Enumerated Hash Algorithms	Numeric Value
None[a]	0
MD5	1
SHA-1	2
SHA-224	3
SHA-256	4
SHA-384	5

a. None is not supported by System SSL. Provided only by TLS 1.2

Table 11-3 Enumerated signature algorithms and numeric values

Enumerated Signature Algorithms	Numeric Value
Anonymous[a]	0
RSA	1
DSA	2
ECDSA	3

a. Anonymous is not supported by System SSL. Provided only by TLS 1.2

To make a hash/signature algorithm pair, the hash algorithm and signature algorithm enumerations are combined together, as shown in Table 11-4.

Table 11-4 Enumerated hash/signature algorithm pair

Algorithm	Enumerated Value (hash and signature algorithm pair)
MD-5 with RSA	0101
SHA-1 with RSA	0201
SHA-224 with RSA	0301
SHA-256 with RSA	0401
SHA-384 with RSA	0501
SHA-512 with RSA	0601
SHA-1 with DSA	0202
SHA-224 with DSA	0302 (Not supported in V1.13)
SHA-256 with DSA	0402 (Not supported in V1.13)
SHA-1 with ECDSA	0203
SHA-224 with ECDSA	0303
SHA-256 with ECDSA	0403

Algorithm	Enumerated Value (hash and signature algorithm pair)
SHA-384 with ECDSA	0503
SHA-512 with ECDSA	0603

11.1.3 Invocation and use

The default is that the TLS V1.2 protocol is not enabled. You must explicitly enable the TLS 1.2 protocol for use. This can be done either of two ways:

1. Set the **GSK_PROTOCOL_TLSV1_2** environment variable:

   Set **GSK_PROTOCOL_TLSV1_2=** to `1`, `ON`, or `ENABLED`, as shown in Example 11-1. If the variable is not specified or set to `OFF`, TLS 1.2 is not negotiated.

   Example 11-1 GSK_PROTOCOL_TLSV1_2 environment variable setting

   ```
   export GSK_PROTOCOL_TLSV1_2=1
   ```

2. Use the `gsk_attribute_set_enum()` API at either the TLS environment level or at the connection level, as Example 11-2 shows.

   Example 11-2 gsk_attribute_set_enum() API

   ```
   rc = gsk_attribute_set_enum(env, GSK_PROTOCOL_TLSV1_2,GSK_PROTOCOL
   _TLSV1_2_ON);
   ```

Override: Using the `gsk_attribute_set_enum()` API overrides the environment variable setting. Also, deprecated APIs do not support the TLS 1.2 protocol.

In addition to being able to specify TLS 1.2 during the handshake, you can query the TLS 1.2 setting by using the `gsk_attribute_get_enum()` API. To query the state of the TLS 1.2 setting, use this command:

```
rc = gsk_attribute_get_enum(conHandle,GSK_PROTOCOL_TLSV1_2, &tlsProtocol);
```

To query whether the TLS 1.2 protocol is being used for the newly established connection, use this command:

```
rc = gsk_attribute_get_enum(conHandle, GSK_PROTOCOL_USED,&protocolUsed);
```

TLS handshake extension support

To use this support, specify the supported signature algorithm. This can be done by either of these methods:

1. Set the **GSK_TLS_SIG_ALG_PAIRS** environment variable:

   Specify enumerated value of hash and signature algorithm pair for the **GSK_TLS_SIG_ALG_PAIRS** environment variable. For example:

   ```
   export GSK_TLS_SIG_ALG_PAIRS=0201
   ```

2. Use the `gsk_attribute_set_buffer()` API:

   Use API at either TLS environment level or at the connection level. For example:

   ```
   rc = gsk_attribute_set_buffer(env, GSK_TLS_SIG_ALG_PAIRS, "0201");
   ```

The default signature algorithm list is 0601 0603 0501 0503 0401 0403 0402 0301 0303 0302 0201 0203 0202 0101.

Override: Using gsk_attribute_set_buffer() API overrides the environment variable setting.

11.1.4 Interactions and dependencies

There are several software dependencies:

- ICSF HCR7780 (z/OS 1.13) or HCR7790 (z/OS 2.1) or later must be installed to allow use of AES-GCM cipher suites.
- System SSL calls ICSF PKCS#11 callable services for AES-GCM encrypt and decrypt functions.
- ICSF controls access to the cryptographic services through the RACF CSFSERV class. Access is needed to the CSF1SKE, CSF1SKD, CSF1TRC, and CSF1TRD resources.
- ICSF determines whether to use CP Assist for Cryptographic Functions (CPACF) hardware to perform AES-GCM encrypt and decrypt functions.

11.1.5 Migration and coexistence considerations

TLS 1.2 support is also available now in z/OS 1.13 with APAR OA39422. ICSF HCR7780 or higher is required for Version 1.13.

Also, toleration APAR OA37102 is available for Versions 1.11 and 1.12 to handle session resumption of SSL sessions from the Parallel Sysplex session ID cache that use an unsupported TLS protocol version. This APAR alters System SSL so that, in a Parallel Sysplex where Parallel Sysplex session ID caching is enabled, an attempt to resume an SSL session by using an unsupported TLS protocol forces a full handshake again.

11.2 Enhancement for Suite B cryptography for TLS

NSA Suite B Cryptography is a policy that defines a set of cryptographic protocols and algorithms for use by the United States government.

Suite B cryptography for TLS 1.2 defines a set of profiles that specify what cryptographic algorithms can be used for key establishment, authentication, and encryption.

In z/OS 2.1, System SSL provides Suite B support in a fully compliant mode.

In System SSL, Suite B operates at the SSL environment level, not at the lower-level connection layer. Therefore, if Suite B is configured in the environment, it applies to all connections tied to that System SSL environment.

When Suite B for TLS is specified, the following restrictions are enforced:

- Use of the TLS 1.2 protocol is required.
- The symmetric algorithm that is used by the handshake and application payload is based on AES-CBC 256 bits or AES-GCM.
- A secure connection is negotiated with ECDH key exchange.
- Certificates that are used during the handshake are signed with ECDSA.

- Elliptical curves that are allowed are P-256 and P-384 curves.
- Restrictions are placed on the client's and server's ability to specify which hash and signature algorithms they accept.

There are two different Suite B profiles supported, one for 128 bits and another for 192 bits. See Table 11-5 for comparison.

Table 11-5 Characteristics of two Suite B profiles

	Suite B 128-bit profile	**Suite B 192-bit profile**
Corresponds to	Elliptical Curve size of 256 bits (secp256r1) Encryption using AES 128 SHA-256 hash algorithm	Elliptical Curve size of 384 bits (secp384r1) Encryption using AES 256 SHA-384 hash algorithm
Cipher suites available	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (C02B) TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 (C023)	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (C02C) TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 (C024)
Signature algorithm used	SHA-256 with ECDSA (0403)	SHA-384 with ECDSA (0503)

11.2.1 Invocation and use

Applications must explicitly enable the Suite B profile. This can be done by either of those methods:

- Set the **GSK_SUITE_B_PROFILE** environment variable:

  Specify the profile that you want to be used for the **GSK_SUITE_B_PROFILE** environment variable. The following values are available:

  `OFF` Default value. Suite B will not be applied.

  `128` The 128-bit profile is used.

  `192` The 192-bit profile is used.

  `ALL` Both 128-bit profile and 192-bit profile are used.

  For example:

  ```
  export GSK_SUITE_B_PROFILE=128
  ```

- Using the `gsk_attribute_set_enum()` API.

  Use the `gsk_attribute_set_enum()` API at the TLS environment level. This cannot be specified at the connection level.

  Use the **GSK_ENUM** ID for the `GSK_SUITE_B_PROFILE` and specify a value for `GSK_ENUM_VALUE`:

  `GSK_SUITE_B_PROFILE_OFF` Default value. Suite B will not be applied.
  `GSK_SUITE_B_PROFILE_128` The 128-bit profile is used.
  `GSK_SUITE_B_PROFILE_192` The 192-bit profile is used.
  `GSK_SUITE_B_PROFILE_ALL` Both 128-bit profile and 192-bit profile are used.

  For example:

  ```
  rc = gsk_attribute_set_enum(env_handle,
  GSK_SUITE_B_PROFILE,GSK_SUITE_B_PROFILE_ 128);
  ```

If not specified or set to OFF, Suite B will not be enforced.

Override: Using gsk_attribute_set_enum() API overrides the environment variable setting. Also, Deprecated APIs do not support Suite B.

11.2.2 Interactions and dependencies

- ICSF HCR7790 or higher must be installed.
- System SSL calls ICSF PKCS#11 CSFPSKE and CSFPSKD callable services to perform AES-GCM encrypt and decrypt functions.
- ICSF controls access to the cryptographic services through the RACF CSFSERV class. Access is needed to the CSF1SKE, CSF1GKP, CSF1GAV, CSF1TRD, CSF1TRC, CSF1PKS, CSF1PKV, and CSF1DVK resources.
- ICSF determines whether to use CPACF hardware to perform AES-GCM encrypt and decrypt functions.

11.3 RFC 5280 certificate validation support

Before z/OS 1.11, z/OS System SSL was made to validate X.509 certificates according to RFC 2459 (Internet X.509 Public Key Infrastructure Certificate and CRL Profile), which was published in 1999. z/OS System SSL validated whether the digital certificate created is consistent with RFC 2459 or not.

RFC 2459 was made obsolete in 2002 by RFC 3280, the Certificate and Certificate Revocation List (CRL) Profile, which contains several differences from RFC 2459. With z/OS 1.11, System SSL enforced certificate validation at the RFC 3280 level with compatibility with RFC 2459. In 2008, RFC 3280 was made obsolete by RFC 5280. With z/OS 2.1, support for certificate validation as defined by RFC 5280 is added to System SSL.

z/OS support for RFC 5280 enables you to use industry-standard specifications and to specify your preferred mode, which enables certificate validation under the more recent and secure specification. With RFC 5280 certificate validation, you can use certificates that follow the standard that is defined in RFC 5280. The validation level can be controlled also.

Certificates that are defined by RFCs 2459 and 3280 are still supported. You can choose which of these validation types to use:

- RFC 2459 only
- RFC 3280 only
- RFC 5280 only (in z/OS 2.1)
- RFC 2459, RFC 3280, and RFC 5280 (updated in z/OS 2.1)

11.3.1 Invocation and use

The sections that follow describe use of SSL/TLS and Certificate Management Services (CMS) APIs.

SSL/TLS users

You can set the certificate validation mode for SSL/TLS users by one of the following ways:

- Set the **GSK_CERT_VALIDATION_MODE** environment variable.

  Specify how the certificate validation is done for the **GSK_CERT_VALIDATION_MODE** environment variable. The following values are available:

2459	Validate according to RFC 2459.
3280	Validate according to RFC 3280.
5280	value in z/OS 2.1. Validate according to RFC 5280.
ANY	Validation can use any supported X.509 certificate validation method. First, validation according to RFC 2459 occurs. If it fails, RFC 3280 validation occurs. If it also fails, RFC 5280 validation occurs in z/OS 2.1.

  For example:

  ```
  export GSK_CERT_VALIDATION_MODE=5280
  ```

- Use the `gsk_attribute_set_enum()` API (SSL environment only).

  These are the possible values:

GSK_CERT_VALIDATION_MODE_5280	value in z/OS 2.1. Validate according to RFC 5280.
GSK_CERT_VALIDATION_MODE_3280	Validate according to RFC 3280.
GSK_CERT_VALIDATION_MODE_2459	Validate according to RFC 2459.
GSK_CERT_VALIDATION_MODE_ANY	Validation can use any supported X.509 certificate validation method.

  For example:

  ```
  rc = gsk_attribute_set_enum (env_handle,GSK_CERT_VALIDATION_MODE,GSK_CERT_
  VALIDATION_MODE_5280);
  ```

- Use the `validation_mode` parameter on the `gsk_validate_certificate_mode()` API.

  Specify the validation_mode parameter. These are the possible values:

GSKCMS_CERT_VALIDATION_MODE_5280	value in z/OS 2.1.Validate according to RFC 5280.
GSKCMS_CERT_VALIDATION_MODE_3280	Validate according to RFC 3280.
GSKCMS_CERT_VALIDATION_MODE_2459	Validate according to RFC 2459.
GSKCMS_CERT_VALIDATION_MODE_ANY	Validation can use any supported X.509 certificate validation method.

Certificate Management Services (CMS) API users

CMS APIs can be used to create or manage your own key database files in a function similar to the SSL **gskkyman** utility.

For CMS API users, setting the validation mode for can be done by specifying the `validation_mode` parameter with the **gsk_validate_certificate_mode()** call, as shown previously for SSL/TLS users.

11.4 SAF key ring certificate chain validation support

With z/OS 2.1, digital certificates in SAF key rings, such as Resource Access Control Facility (RACF) and other security products, can be validated just like certificates in a PKCS#11

token or key database file (validation is always on the root CA for these). With this enhancement, you can validate the root certificate for certificates that are stored in System Authorization Facility (SAF) key rings.

11.4.1 Invocation and use

By default, SAF key ring certificate chain validation is done only for the trust anchor certificate, which is the same behavior as in previous releases. That is, if validation to the root certificate is not enabled, a sole intermediate certificate is found in a SAF key ring, and the next issuer is not found in the same SAF key ring, then the intermediate certificate is allowed to act as a trust anchor and the certificate chain is considered complete.

If validation to the root certificate is enabled, an intermediate certificate in a SAF key ring is not allowed to be established as a trust anchor and full certificate validation to the root CA must occur.

To enable validation to the root certificate for certificates in SAF key ring, you must explicitly set this behavior by using one of the following methods:

- Use the `gsk_attribute_set_enum()` API (SSL environment level only).

  These are the possible values:

`GSK_CERT_VALIDATE_KEYRING_ROOT_ON`	Validate SAF key ring certificates to the root CA.
`GSK_CERT_VALIDATE_KEYRING_ROOT_OFF`	Validate SAF key ring certificates to the intermediate trust anchor.

  For example:

  ```
  rc = gsk_attribute_set_enum (env,GSK_CERT_VALIDATE_KEYRING_ROOT,GSK_CERT_
  VALIDATE_KEYRING_ROOT_ON);
  ```

- Use the `validate_root` parameter on the `gsk_validate_certificate_mode()` API (parameter).

  These are the possible values:

`GSKCMS_CERT_VALIDATE_KEYRING_ROOT_ON`	Validate SAF key ring certificates to the root CA.
`GSKCMS_CERT_VALIDATE_KEYRING_ROOT_OFF`	Validate SAF key ring certificates to the intermediate trust anchor.

  Also, the `arg_count` parameter must be set to `1` in this API.

- Use the GSK_CERT_VALIDATE_KEYRING_ROOT (environment variable)

  These are the possible values:

`ON`	Validate SAF key ring certificates to the root CA.
`OFF`	Default value. Validate SAF key ring certificates to the intermediate trust anchor.

Note: These APIs and environment variables cannot be applied to an individual SSL connection level. They are applied to the SSL environment level, which means that all SSL connections are affected. Also, these APIs and environment variable have no effect on certificate validation with certificates in key database files or PKCS #11 tokens. Certificate validation in key database files or PKCS #11 tokens is always done to the root.

11.5 DSA 2048-bit key support

In z/OS 2.1, both the **gskkyman** utility and Certificate Management Services (CMS) APIs are updated to support the creation of DSA 2048-bit keys for CA certificates, user or server certificates, and to enable the use of DSA 2048-bit keys.

11.5.1 Invocation and use

In z/OS 2.1, **gskkyman** supports creation and management of CA and end-entity digital signature algorithm (DSA) 2048-bit certificates. Previously, all certificate types (CA and user or server) were presented on one panel, as shown in Figure 11-1 on page 160.

Some of the existing **gskkyman** menus (when running in interactive mode, not the command-line mode) have been refined to make the tasks simpler and more intuitive. The previous certificate panel is now multiple panels. The first menu now gives you a choice of whether you want a CA, User, or Server certificate. That menu is then followed by a choice of the key algorithm type. Based on the key algorithm type, a menu indicates the key sizes that are available. After you make your selection, you can select the signature digest type. See Figure 11-1 on page 160.

Note: The **gskkyman** utility provides full function for a PKCS #11 token certificate with a clear private key. But if a PKCS#11 certificate has a secure private key, the following functions are not allowed:

- Copying the certificate and key to another token
- Exporting the certificate and key to a file
- Creating a signed certificate and key
- Creating a certificate renewal request

When displaying Token Key information for a PKCS #11 certificate's private key, the private key type is indicated in the Private Key Type section in Token Key information panel, which shows that the private key is either Clear or Secure.

.

```
       Certificate Usage

   1 - CA certificate
   2 - User or server certificate

Select certificate usage (press ENTER to return to menu): 1

       Certificate Key Algorithm

   1 - Certificate with an RSA key
   2 - Certificate with a DSA key
   3 - Certificate with an ECC key

Select certificate key algorithm (press ENTER to return to menu): 2

       DSA Key Size

   1 - 1024-bit key
   2 - 2048-bit key

Select DSA key size (press ENTER to return to menu): 2

       Signature Digest Type

   1 - SHA-224
   2 - SHA-256

Select Digest Type (press ENTER to return to menu): 2
Enter label (press ENTER to return to menu): Test CA certificate for Redbook
Enter subject name for certificate
  Common name (required): Test CA
  Organizational unit (optional):
  Organization (required): ITSO
  City/Locality (optional): Pok
  State/Province (optional): NY
  Country/Region (2 characters - required): US
Enter number of days certificate will be valid (default 365):

Enter 1 to specify subject alternate names or 0 to continue: 0

Please wait .....

Certificate created.
```

Figure 11-1 gskkyman menus

CMS updates

The following CMS APIs are updated:

- gsk_make_signed_data_content
- gsk_make_signed_data_content_extended
- gsk_make_signed_data_msg
- gsk_make_signed_data_msg_extended
- gsk_read_signed_data_content

- gsk_read_signed_data_content_extended
- gsk_read_signed_data_msg
- gsk_read_signed_data_msg_extended
- gsk_sign_certificate
- gsk_sign_crl
- gsk_sign_data
- gsk_verify_certificate_signature
- gsk_verify_crl_signature
- gsk_verify_data_signature
- gsk_construct_certificate
- gsk_construct_renewal_request
- gsk_construct_self_signed_certificate
- gsk_construct_signed_certificate
- gsk_create_certification_request
- gsk_create_database_renewal_request
- gsk_create_database_signed_certificate
- gsk_create_self_signed_certificate
- gsk_create_signed_certificate
- gsk_create_signed_certificate_record
- gsk_create_signed_certificate_set
- gsk_create_signed_crl_record
- gsk_generate_key_pair
- gsk_generate_key_parameters

11.5.2 Migration and coexistence considerations

Installations and products that have created automated scripts to interact with the **gskkyman** menus need to be modified to work with the menus. Documentation that describes the **gskkyman** menus is updated to describe the menus. See the Certificate/Key management section in the *Cryptographic Services System Secure Sockets Layer Programming*, SC41-7495 manual for more details about **gskkyman** use.

11.6 System SSL applications in FIPS mode using ICSF functions

In z/OS 2.1, ICSF must be started before running System SSL applications in FIPS mode. The random number generator and Diffie-Hellman functions of ICSF are used in this environment.

When a System SSL application is running in non-FIPS mode (default mode), System SSL attempts to use ICSF's random number generation. The Random number generation of ICSF allows the application to take advantage of the ICSF software or the Crypto Express cards for generating random data. System SSL's software is used for random number generation if ICSF is not active.

11.6.1 Invocation and use

You must ensure that the user ID that starts the SSL application, as well as any applications that use SSL, have adequate permissions. For example, let's say that the user ID calls the **gskkyman** utility to manage FIPS key database files, or that the GSKSRVR started task in FIPS mode is called. If they cannot access the CSFRNG resource of the CSFSERV class, System SSL will not be able to use ICSF. Informational message ICH408I (which

indicates insufficient authorization) might also be issued to the console. Although System SSL processing would continue, your application would be using the System SSL's random number generation support. Therefore, adequate permissions for an application or user to the CSFRNG resource are necessary.

To use Diffie-Hellman functions of ICSF, access to the following resources in RACF CSFSERV class is required: CSF1TRC, CSF1DVK, CSF1GKP, CSF1GSK, CSF1GAV, CSF1TRD. See RACF CSFSERV resource requirements in *Cryptographic Services System Secure Sockets Layer Programming*, SC41-7495 for details.

11.6.2 Migration and coexistence considerations

APAR OA40916 on z/OS V1R12 and z/OS V1R13 assists with finding SSL applications that are running in FIPS mode without ICSF available.

11.7 Enterprise PKCS #11 support

System SSL APIs are updated to support PKCS #11 secure keys by label. This uses the enhancements in other components to support PKCS #11 secure keys, such as ICSF (HCR77A0) and RACF. These enhancements are used by other components, such as PKI Services (see Chapter 10, "PKI Services" on page 139).

11.7.1 Invocation and use

System SSL APIs are updated to use ICSF PKCS #11 callable services to perform operations by using secure private keys that are located in the Token Key Data Set (TKDS). See *Cryptographic Services System Secure Sockets Layer Programming*, SC41-7495 for a full description of the updated APIs.

Two APIs were introduced with z/OS 2.1:

- `gsk_modify_pkcs11_key_label()`
- gsk_make_enveloped_private_key_message()

The following APIs are updated to support PKCS #11 secure keys:

- `gsk_get_cms_vector()`
- `gsk_sign_data()`
- `gsk_make_enveloped_data_content()`
- `gsk_make_enveloped_data_content_extended()`
- `gsk_make_enveloped_data_msg()`
- `gsk_make_enveloped_data_msg_extended()`
- `gsk_read_enveloped_data_content()`
- `gsk_read_enveloped_data_content_extended()`
- `gsk_read_enveloped_data_msg()`
- `gsk_read_enveloped_data_msg_exteneded()`
- `gsk_encode_export_key()`
- `gsk_export_key()`

11.8 Documentation and reference

For more information on the topics in this chapter, see the following sites and publications:

- z/OS Cryptographic Services System SSL Programming, SC24-5901-11

  https://ibm.biz/BdFx6t

- *Cryptographic Services System Secure Sockets Layer Programming*, SC41-7495 (index)

  https://ibm.biz/BdFx6W

- *RFC 5246, The Transport Layer Security (TLS) Protocol Version 1.2*

  http://datatracker.ietf.org/doc/rfc5246/

- *RFC 5280, Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*

  http://datatracker.ietf.org/doc/rfc5280/

- *RFC 5288, AES Galois Counter Mode (GCM) cipher suites for TLS*

  http://datatracker.ietf.org/doc/rfc5288

- *RFC 5289, TLS Elliptic Curve cipher suites with SHA-256/384 and AES Galois Counter Mode (GCM)*

  http://datatracker.ietf.org/doc/rfc5289/

- *RFC 5430, Suite B Profile for Transport Layer Security (TLS)*

  http://datatracker.ietf.org/doc/rfc5430/

- Suite B Cryptography / Cryptographic Interoperability, formalized in CNSSP-15, National Information Assurance Policy on the Use of Public Standards for the Secure Sharing of Information Among National Security Systems, March 2010

  http://nsabackups.com/ia/programs/suiteb_cryptography/index.shtml

12

z/OS Batch Runtime

This chapter describes the Batch Runtime enhancements and functions that were introduced with IBM z/OS Version V2R1 (2.1):

- 12.1, “Overview of Batch Runtime changes” on page 166
- 12.2, “Support for launching PL/I applications” on page 166
- 12.3, “IBM batch programming model support” on page 169
- 12.4, “Documentation and reference” on page 171

© Copyright IBM Corp. 2014. All rights reserved.

12.1 Overview of Batch Runtime changes

z/OS Batch Runtime allows interoperability between PL/I, COBOL, and Java applications that run on z/OS. It also allows for applications to be re-engineered to use the IBM Java Batch Common Programming Model Support. It provides a managed environment that enables shared access to an IBM DB2 connection by PL/I, COBOL, and Java programs. Updates to DB2 are committed in a single transaction. (Updates to multiple databases are not supported.)

Figure 12-1 shows a high-level overview of the z/OS Batch Runtime environment. The batch container performs the initialization that sets up the environment for PL/I, COBOL, Java, and DB2 interoperability. This includes the following tasks:

- Set up the proper Language Environment for the PL/I or COBOL programs to run.
- Set up the job step under the umbrella of a Resource Recovery Service (RRS)-managed global transaction.
- Initiate the DB2 JDBC driver in this special Batch Container mode.
- Start the DB2 JDBC driver to create a DB2 connection and attachment thread.
- Start the primary PL/I, COBOL, or Java application after the environment is initialized.

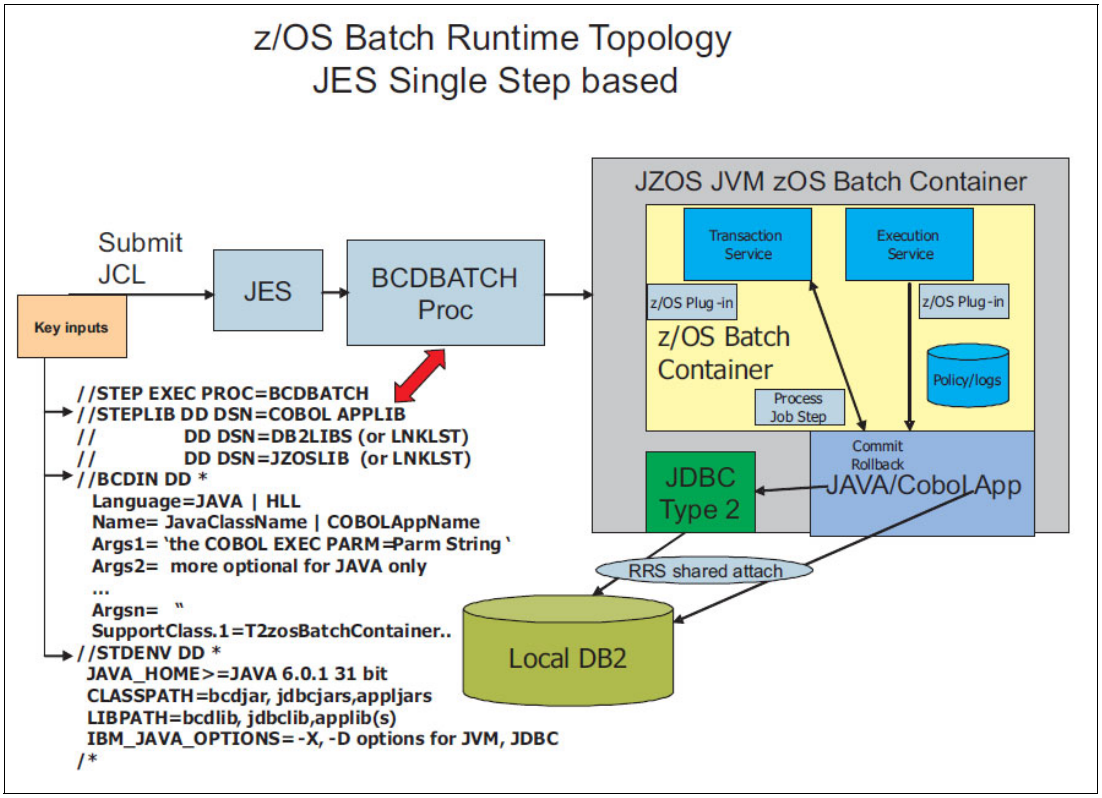


Figure 12-1 z/OS Batch Runtime overview

12.2 Support for launching PL/I applications

In z/OS 2.1, the Batch Runtime feature is enhanced to include PL/I as a supported language. When mixing COBOL, PL/I, and Java programs in a batch environment, the z/OS Batch Runtime provides a managed environment to govern the different programming models that

are being used. Just as with the IBM COBOL support, the environment is bound and centered around traditional JES submitted job steps.

In particular, this managed environment provides a framework and APIs to enable shared access to a local DB2 for z/OS database connection by COBOL, PL/I, and Java programs. Updates to the database across language boundaries are committed within a single Resource Recovery Services (RRS) managed transaction scope. in z/OS 2.1 is support for Transactional Virtual Storage Access Method (VSAM) across these same language environments.

Also, Batch Runtime can now launch an enterprise PL/I main routine, similar to the launch of COBOL that was introduced in z/OS Version 1.13.

12.2.1 Invocation and use

Identical to the IBM COBOL support, z/OS Batch Runtime control statements are read from the BCDIN DD defined in the batch JCL.

The `bcd.applicationLanguage=` statement is changed to accept the Language Environment value to indicate that the program being launched is written in either COBOL or PL/I. The COBOL language value is still accepted, but the Language Environment value is the preferred syntax. Batch Runtime processes both COBOL and PL/I applications in the same way. Figure 12-2 shows an example.

```
//BCDIN DD *
bcd.applicationLanguage=LE
bcd.applicationName=PLITEST
```

Figure 12-2 Start a PL/I program by using Batch Runtime

The z/OS Batch Runtime starts an application by calling and fetching it. As a result, the PL/I external procedure that is being started must specify the *Fetchable* and *Assembler* options. The Main option cannot be used. Therefore, at a minimum, any PL/I Main routine that is started must be at least slightly modified, as shown in Figure 12-3. The Assembler option is also needed so that any PL/I Main application started can set a return code by using the PLIRETC function upon return to z/OS Batch Runtime.

```
PLITEST: Procedure Options( Fetchable
                            Assembler );
```

Figure 12-3 Specifying the Fetchable and Assembler options

When binding the PL/I application, the **ENTRY CEESTART** must not be used. Instead, the entry point must specify the name of the external procedure. For example, the procedure that is shown in Figure 12-3 would be bound as shown in Figure 12-4.

```
//BIND.SYSIN DD *
ENTRY PLITEST
NAME PLITEST(R)
```

Figure 12-4 Binder options to specify a name for the entry point

Commit and Rollback helper functions

To simplify the process of calling the Batch Runtime commit and rollback routines, the Batch Runtime provides the convenience of `bcdcommit()` and `bcdrollback()` methods, which can be called directly from a PL/I application. Use of the helper functions replaces the JNI calls to call back to the batch container. Figure 12-5 shows an example of using the helper functions from a PL/I program.

The methods are in a DLL `libbcduser.so` that is included with z/OS 2.1. For PL/I callers, an include file is provided in SYS1.SAMPLIB(BCDPLIH) that defines the entry points. PL/I applications must include the `bcdlibuser.x` sidedeck when binding their applications to make the methods accessible to the program. Figure 12-6 on page 169 shows how to bind a PL/I program with the commit and rollback helper functions.

```
*Process Limits( Extname( 100 ) );
*Process Margins( 1, 100 );
*Process Display(STD) Rent;
*Process Default( ASCII ) Or('|');
*Process Options InSource Source Nest Macro Storage;
*Process Aggregate Offset;
*Process List Flag(I) MarginI('|');
*Process Opt(2) Attributes(Full) Xref(Short);

/* PL/I Module calling batch container helpers */
PLIHELP: Procedure Options( Fetchable
                           Assembler );

%INCLUDE BCDPLIH;

Dcl rc fixed bin(31);
Display("Calling bcdcommit helper ...");
rc = bcdcommit();
Display("bcdcommit rc=" || rc);

Display("Calling bcdrollback helper ...");
rc = bcdrollback();
Display("bcdrollback rc=" || rc);

End PLIHELP;
```

Figure 12-5 Calling the commit and rollback helper functions from PL/I

```
//jobname JOB
//*
//* Complies and links PLIHELP
//*
//JCLLIB JCLLIB ORDER=PLI.PLI410.SIBMZPRC
//*
//STEP1 EXEC IBMZCB,
// REGION=0M,
// LNGPRFX='PLI.PLI410',
// PARM.BIND=('OPTIONS=OPTS')
//*
//PLI.SYSLIB DD DSN=SYS1.SAMPLIB,DISP=SHR
//*
//PLI.SYSIN DD DSN=IBMUSER.BATCH.SOURCE(PLIHELP),DISP=SHR
//*
//BIND.SYSLMOD DD DSN=IBMUSER.BATCH.LOAD,DISP=SHR
//BIND.OPTS DD *
MAP
RENT
DYNAM=DLL
CASE=MIXED
LIST=ALL
XREF
//*
//BIND.SYSIN DD *
INCLUDE '/usr/lpp/bcp/lib/libbcduser.x'
ENTRY PLIHELP
NAME PLIHELP(R)
//
```

Figure 12-6 Binder options for PL/I program using the commit and rollback helper functions

12.2.2 Installation

z/OS Batch Runtime is provided as part of z/OS Version 2.1. The batch container is implemented in Java and installs into the existing path: `/usr/lpp/bcp`. The BCDPROC procedure is installed into `SYS1.PROCLIB`. The sample JCL BCDBATCH is installed in SYS1.SAMPLIB.

12.3 IBM batch programming model support

Further enhancements are made to support the IBM batch programming model. This support, fully documented in Appendix A of *z/OS Batch Runtime: Planning and User's Guide*, SA23-1376, is for JES-submitted Java applications that are described in an XML-like policy (called *xJCL*) and follows the rules intrinsic to the IBM batch programming model. This is the same descriptive language that is used in IBM WebSphere® batch support, although with a traditional JES submission and limited to a single, non-persistent Java virtual machine (JVM) for each job step.

With this support, applications can be deployed on WebSphere Application Server (for z/OS or distributed platforms). Also, if they are compliant with the limitations of z/OS Batch Runtime, they can be deployed as a z/OS batch job.

12.3.1 Invocation and use

In the IBM batch programming model, a job is described by the xJCL language. Use the IBM Rational® Application Developer xJCL editor to create xJCL files. xJCL is an XML-based file that describes the batch job and its steps. Each batch step contains business logic to run a portion of the job and typically reads a record, implements the business logic, and then gets the next record.

To support the IBM batch programming model, the z/OS Batch Runtime procedure, BCDPROC, now supports two data definition (DD) cards. The BCDXJCL DD names a file that contains the xJCL. This file is usually stored in a file system. The BCDPROP DD names a file that contains substitution properties to be applied to xJCL. This file is usually provided inline, within the job. The BCDBATCH sample JCL is provided in SYS1.SAMPLIB and shows how to use the DD cards.

To run a Java program by using the IBM batch programming model, **`bcd.applicationLanguage=XJCL`** must be specified under the BCDIN DD. The **`bcd.xJCLEncoding=encoding-name`** option can be used to set the xJCL file encoding. The default is ISO8859-1.

Job restart is supported through the checkpoint process. The xJCL defines a checkpoint policy that determines when a checkpoint is to be taken. When a checkpoint occurs, the Batch Runtime initiates a commit sequence.

If a step fails, the job is terminated. When restart is requested, the Batch Runtime resumes the job at the failed step and restarts processing from the last commit. The input and output to the step is repositioned so that processing based on the last committed records processed can restart.

By default, restart is not enabled for a job. Use the **`bcd.xJCLRestartEnabled`** control statement to enable the job for restart.

Note: Enabling this option causes the Batch Runtime feature to write persistence data to the file system that the checkpoint state.

The restart job ID is not the same as the JES job ID. When the job is submitted, an internal job ID is created and used by the Batch Runtime to manage the job. The job ID that is assigned is contained in message BCD0310I when the job starts. For a restart, use this job ID in the **`bcd.xJCLRestartJobId`** statement.

Figure 12-7 on page 171 shows a fragment of the batch JCL that is used to start an xJCL job. Notice the format of the BCDXPROP DD input. Each line specifies a substitution property name and its value. These correspond to the property names that are defined in the xJCL. When the xJCL is processed, the values that are specified here override defaults in the xJCL.

```
//BATCH EXEC BCDPROC,REGION=0M,LOGLVL='+T',VERSION=61
//*
//BCDXJCL DD PATH='yourpathxJCL.xml',
// PATHOPTS=ORDONLY
//*
//BCDXPROP DD *
inputDataStream=inputFile.txt
outputDataStream=outputFile.txt
fileEncoding=IBM-1047
debugEnabled=true
//BCDIN DD *
bcd.applicationLanguage=XJCL
bcd.xJCLEncoding=ibm-1047
bcd.xJCLRestartEnabled=true
#
bcd.supportClass.1=com.ibm.db2.jcc.t2zos.T2zosBatchContainerSupport
```

Figure 12-7 JCL to invoke a xJCL job

12.3.2 Installation

z/OS Batch Runtime is provided as part of z/OS 2.1. The batch container is implemented in Java and installs into the existing path: `/usr/lpp/bcp`. The BCDPROC procedure is installed into SYS1.PROCLIB. The sample JCL BCDBATCH is installed in SYS1.SAMPLIB.

12.4 Documentation and reference

For documentation and references for the described enhancements, see the following publications and the citations in the “Related publications” on page 537.

- *New Ways of Running IBM z/OS Batch Applications*, SG24-8116

  http://ibm.co/1m9eMRs
- *z/OS Batch Runtime Planning and User's Guide*, SA23-2270-01

  http://ibm.co/1xH77gx
- *z/OS Batch Runtime: Planning and User's Guide,* SA23-1376

  https://ibm.biz/BdFDM4

13

Consoles and auto-reply

This chapter describes the following enhancements, functions, and features for consoles and auto-reply that were introduced in IBM z/OS Version 2 Release 1 (2.1):

- 13.1, “Dynamically add and delete consoles” on page 174
- 13.2, “z/OS support for the Integrated 3270 Console interface” on page 178
- 13.3, “Enhanced synchronous WTOR notification” on page 185
- 13.4, “FORCE TCB command” on page 187
- 13.5, “Auto-reply support for synchronous WTOR messages” on page 188
- 13.6, “Auto-reply rate limit” on page 191
- 13.7, “Documentation and reference” on page 192

© Copyright IBM Corp. 2014. All rights reserved.

13.1 Dynamically add and delete consoles

Before z/OS V2.1, an IPL was required to define a console and an IBM Parallel Sysplex-wide IPL was required to delete a console (except with the use of SAMPLIB utility **IEARELCN**). Starting with z/OS 2.1, it is possible to add and delete MCS-type consoles dynamically, without an outage. The following console types support can be added dynamically:

- Multiple console support (MCS)
- HMC integrated 3270 console (HMCS), which is described in 13.2, “z/OS support for the Integrated 3270 Console interface” on page 178
- Shared MCS (SMCS)
- Subsystem

13.1.1 Invocation and use

To dynamically add a console, update the CONSOLxx PARMLIB member with the console definition, and then use the **SET CON=xx** command, where xx is the CONSOLxx two letter suffix. The **SET CON=xx** command reads the specified CONSOLxx PARMLIB member and creates consoles. The command also processes the operational settings that are defined in CONSOLxx. After execution of the **SET CON=xx** command, any consoles are defined, and console operational settings are altered. The following operational settings can be changed:

- DEFAULT statement: HOLDMODE and SYNCHDEST keywords
- HARDCOPY statement: HCFORMAT keyword
- INIT statement: CMDDELIM and the CTRACE default keywords

Unlike CONSOLxx processing during IPL time, the absence of a keyword does not cause the default to be implemented. Only specified settings are processed by the **SET CON=xx** command.

Note: The **SET CON=xx** command is supported only in console distributed mode.

To delete a console, use the **SETCON DELETE,CN=conname** command, where conname is the name of the console to be deleted. The command can be used to delete MCS, HMCS, SMCS, subsystem, and Extended Multiple Console Support (EMCS) consoles.

Example of adding and deleting a console dynamically

Figure 13-1 shows our initial console configuration.

```
D C,L
CNZ4100I 09.42.12 CONSOLE DISPLAY 019
CONSOLES MATCHING COMMAND: D C,L
MSG:CURR=0    LIM=1500 RPLY:CURR=0    LIM=999   SYS=SC74     PFK=00
HARDCOPY  LOG=(SYSLOG,OPERLOG)  CMDLEVEL=CMDS
      ROUT=(ALL)
NAME     TYPE    STATUS         DEFINED             MATCHED
SMCSCON1 SMCS    INACT          *ALL                *ALL
SMCSCON2 SMCS    INACT          *ALL                *ALL
SMCSCON3 SMCS    INACT          *ALL                *ALL
SMCSCON4 SMCS    INACT          *ALL                *ALL
```

Figure 13-1 Initial console configuration

We update the CONSOL00 PARMLIB member with two console definitions, as shown in Figure 13-2. Our system name is SC74, so the value of the symbol &SYSNAME is equal to SC74. We define the symbols &SYSCON1 and &SYSCON2 with the values F400 and F480.

```
CONSOLE DEVNUM(&SYSCON1.)
        UNIT(3270-X)
        NAME(&SYSNAME.&SYSCON1.)
        AUTH(MASTER)
        USE(FC)
        AREA(NONE)
        MSCOPE(*)
        CMDSYS(*)
        DEL(RD)
        LEVEL(ALL)
        CON(N)
        RNUM(20)
        SEG(20)
        RTME(1/4)
        MFORM(T,S)
        PFKTAB(PFKTAB1)
        INTIDS(Y)
        UNKNIDS(Y)
CONSOLE DEVNUM(&SYSCON2.)
        UNIT(3270-X)
        NAME(&SYSNAME.&SYSCON2.)
        AUTH(MASTER)
        USE(FC)
        AREA(NONE)
        MSCOPE(*)
        CMDSYS(*)
        DEL(RD)
        LEVEL(ALL)
        CON(N)
        RNUM(20)
        SEG(20)
        RTME(1/4)
        MFORM(T,S)
        PFKTAB(PFKTAB1)
        INTIDS(Y)
        UNKNIDS(Y)
```

Figure 13-2 Console definitions in CONSOL00

We verify that we are running in console distributed mode by using the **D O,MODE** command, and then use the **SET CON=00** command to add the consoles, as shown in Figure 13-3 on page 176.

```
D O,MODE
CNZ9006I 13.32.14 DISPLAY O,MODE 096
CURRENT: DISTRIBUTED
SYSTEM   MIGRATION STATUS
SC75     MIGRATION SUPPORTED
SC74     MIGRATION SUPPORTED
SYSPLEX ABLE TO MIGRATE: YES
SET CON=OO
CNZ6003I COMMAND ACCEPTED FOR EXECUTION: SET CON=OO
IEE252I MEMBER CONSOLOO FOUND IN SYS1.PARMLIB
CNZ6004I SYSCONS CANNOT BE ADDED DYNAMICALLY
IEE252I MEMBER  CNGRPOO FOUND IN SYS1.PARMLIB
IEE712I SET CNGRP PROCESSING COMPLETE
T PFK=OO   ISSUED FOR SET CON=OO
VARY ,HARDCPY,CMDS,ROUT=ALL   ISSUED FOR SET CON=OO
T MPF=(OO)   ISSUED FOR SET CON=OO
MN DSNAME   ISSUED FOR SET CON=OO
D C,HC,L=Z
CNZ4100I 13.33.57 CONSOLE DISPLAY 108
CONSOLES MATCHING COMMAND: D C,HC
MSG:CURR=0    LIM=1500 RPLY:CURR=0    LIM=999   SYS=SC74     PFK=00
HARDCOPY  LOG=(SYSLOG,OPERLOG)  CMDLEVEL=CMDS
      ROUT=(ALL)
LOG BUFFERS IN USE: 0       LOG BUFFER LIMIT: 6000
IEE252I MEMBER PFKTABOO FOUND IN SYS1.PARMLIB
IEE252I MEMBER MPFLSTOO FOUND IN SYS1.PARMLIB
IEE536I PFK      VALUE OO NOW IN EFFECT
IEE712I SET MPF  PROCESSING COMPLETE
```

Figure 13-3 Using SET CON= command to dynamically add consoles

Figure 13-4 shows the updated console configuration.

```
D C,L
CNZ4100I 13.34.50 CONSOLE DISPLAY 116
CONSOLES MATCHING COMMAND: D C,L
MSG:CURR=0    LIM=1500 RPLY:CURR=0    LIM=999   SYS=SC74     PFK=00
HARDCOPY  LOG=(SYSLOG,OPERLOG)  CMDLEVEL=CMDS
      ROUT=(ALL)
NAME     TYPE    STATUS         DEFINED             MATCHED
SC74F400 MCS     INACT          SC74                SC74
SC74F480 MCS     INACT          SC74                SC74
SMCSCON1 SMCS    INACT          *ALL                *ALL
SMCSCON2 SMCS    INACT          *ALL                *ALL
SMCSCON3 SMCS    INACT          *ALL                *ALL
SMCSCON4 SMCS    INACT          *ALL                *ALL
```

Figure 13-4 Updated console configuration

We vary the consoles online, as shown in Figure 13-5 on page 177.

```
V F400,CONSOLE
CNZ4100I 13.35.35 CONSOLE DISPLAY 122
CONSOLES MATCHING COMMAND: D C,U=(F400)
MSG:CURR=0    LIM=1500 RPLY:CURR=0    LIM=999   SYS=SC74     PFK=00
SC74F400  TYPE=MCS      STATUS=ACT-SC74
          DEFINED=(SC74)
          MATCHED=(SC74)
   ATTRIBUTES ON SC74
      AUTH=(MASTER)    CMDSYS=*             NBUF=0        SUPSBY=N
      DEV=F400         LOGON=OPTIONAL       USERID=N/A
      MFORM=(T,S)      AREA=(Z)             PFKTAB=PFKTAB1
      USE=FC  DEL=RD   RTME=1/4  RNUM=20    SEG=20    CON=N
      LEVEL=(ALL)
      MONITOR=(NONE)                        INTIDS=Y  UNKNIDS=Y
      ROUT=(NONE)
      MSCOPE=(*)
V F480,CONSOLE
CNZ4100I 13.36.41 CONSOLE DISPLAY 131
CONSOLES MATCHING COMMAND: D C,U=(F480)
MSG:CURR=0    LIM=1500 RPLY:CURR=0    LIM=999   SYS=SC74     PFK=00
SC74F480  TYPE=MCS      STATUS=ACT-SC74
          DEFINED=(SC74)
          MATCHED=(SC74)
   ATTRIBUTES ON SC74
      AUTH=(MASTER)    CMDSYS=*             NBUF=0        SUPSBY=N
      DEV=F480         LOGON=OPTIONAL       USERID=N/A
      MFORM=(T,S)      AREA=(Z)             PFKTAB=PFKTAB1
      USE=FC  DEL=RD   RTME=1/4  RNUM=20    SEG=20    CON=N
      LEVEL=(ALL)
      MONITOR=(NONE)                        INTIDS=Y  UNKNIDS=Y
      ROUT=(NONE)
      MSCOPE=(*)
```

Figure 13-5 Vary consoles online

Figure 13-6 on page 178 shows how to use the **SETCON DELETE** command to delete the consoles. The consoles must be brought offline before they can be deleted.

```
SETCON DELETE,CN=SC74F400
CNZ4301I CONSOLE SC74F400 WAS NOT REMOVED. CONSOLE IS ACTIVE
V CN(SC74F400),OFFLINE
IEF524I F400 PENDING OFFLINE
IEE303I SC74F400 OFFLINE
V CN(SC74F480),OFFLINE
IEF524I F480 PENDING OFFLINE
IEE303I SC74F480 OFFLINE
SETCON DELETE,CN=SC74F400
CNZ4300I MCS CONSOLE SC74F400 HAS BEEN REMOVED
SETCON DELETE,CN=SC74F480
CNZ4300I MCS CONSOLE SC74F480 HAS BEEN REMOVED
D C,L
CNZ4100I 15.27.48 CONSOLE DISPLAY 656
CONSOLES MATCHING COMMAND: D C,L
MSG:CURR=0    LIM=1500 RPLY:CURR=0    LIM=999   SYS=SC74     PFK=00
HARDCOPY  LOG=(SYSLOG,OPERLOG)  CMDLEVEL=CMDS
      ROUT=(ALL)
NAME     TYPE    STATUS         DEFINED             MATCHED
SMCSCON1 SMCS    INACT          *ALL                *ALL
SMCSCON2 SMCS    INACT          *ALL                *ALL
SMCSCON3 SMCS    INACT          *ALL                *ALL
SMCSCON4 SMCS    INACT          *ALL                *ALL
```

Figure 13-6 Dynamically delete consoles

13.1.2 Installation

If necessary, `CONSOLxx` PARMLIB members may be created to dynamically add consoles.

13.1.3 Migration and coexistence considerations

Any SMCS or subsystem console that is defined dynamically on a z/OS 2.1 system by using this function is available to be used on any system in the Parallel Sysplex. Systems with this function can delete inactive consoles, regardless of which system defined them. A z/OS 2.1 system can delete a console that was defined on a system with an earlier z/OS version.

13.2 z/OS support for the Integrated 3270 Console interface

Starting with Version 2.1, z/OS supports using the Integrated 3270 Console interface on the Hardware Management Console (HMC) as a standard console in z/OS. This console type is called *HMCS*. The HMCS provides access to a z/OS operator console when existing z/OS consoles are unavailable. After it is defined, the HMCS uses the same interface as nucleus initialization program (NIP), MCS, and SMCS consoles. This enables operators to use them easily. This is an improvement over the system console interface in the HMC, which is very different from a standard z/OS console.

Table 13-1 on page 179 lists HMCS functions in comparison to the system console.

Table 13-1 System console problem and HMCS solution

System console problem	HMCS solution
Operator interface is very different from MCS or SMCS consoles. Operator errors occur during critical situations because of unfamiliarity.	HMCS console interface is identical to the current NIP, MCS, or SMCS interface, so there is no learning curve.
Replying to priority WTORs requires a check box to be marked before the reply is accepted.	Priority WTORs are displayed in the same manner as on MCS consoles, and replying to these WTORs does not require special check boxes.
Using the system console in PD mode can cause high processor use and storage shortages in the console address space.	Message display on HMCS consoles is identical to MCS and SMCS consoles, and identical queuing structures are used. The long queue lengths associated with the system console do not exist with HMCS consoles.
Displayed messages contain only a time stamp and message text. Other useful information, such as system name and job name where the message was issued, is not displayed.	HMCS consoles can display additional text when the message is displayed. This is identical to the MCS and SMCS capabilities.
Messages that are retained on the screen can be deleted manually. Each message must be deleted individually, and it takes several mouse clicks to delete one message.	HMCS consoles have the same message retention and commands to remove the messages as do MCS and SMCS consoles. Groups of messages can be removed at one time.
No LOGON support for security product control over operator commands.	Security product controls commands as on MCS and SMCS consoles.

13.2.1 Invocation and use

Use the `DEVNUM(HMCS)` keyword to define the HMCS in `CONSOLxx`. The HMCS can be dynamically added without an IPL. Figure 13-7 on page 180 shows an example of defining an HMCS console in the `CONSOLxx` PARMLIB member. Only one HMCS can be defined per z/OS image. The HMCS screen size is always 43 rows by 80 columns. The HMCS supports seven colors and extended highlighting (reverse video, blinking, underscore), which are controlled by `MPFLSTxx` PARMLIB specifications. As with SMCS consoles, these conditions apply:

- SD and MS modes are not supported by the HMCS.
- Messages are displayed on the HMCS console if the operator has not yet logged on, and `LOGON` is defined as `REQUIRED` or `AUTO`. Commands are rejected until a LOGON is accepted, `LOGON(OPTIONAL)` can be specified for the HMCS console.

```
CONSOLE DEVNUM(HMCS)
        NAME(&SYSNAME.HMCS)
        AUTH(MASTER)
        USE(FC)
        AREA(NONE)
        MSCOPE(*)
        CMDSYS(*)
        DEL(RD)
        LEVEL(ALL)
        CON(N)
        RNUM(20)
        SEG(20)
        RTME(1/4)
        MFORM(T,S)
        PFKTAB(PFKTAB1)
        INTIDS(Y)
        UNKNIDS(Y)
```

Figure 13-7 HMCS console definition in CONSOLxx

After the HMCS console is defined, it remains in standby, as shown in Figure 13-8 on page 181. *Console standby* state is a state for consoles. It is described in “Standby console state” on page 183.

```
SET CON=00
CNZ6003I COMMAND ACCEPTED FOR EXECUTION: SET CON=00
IEE252I MEMBER CONSOL00 FOUND IN SYS1.PARMLIB
CNZ6004I SYSCONS CANNOT BE ADDED DYNAMICALLY
IEA196I CONSOL00 F400: DEVNUM ALREADY DEFINED. STATEMENT IGNORED.
IEA196I CONSOL00 F480: DEVNUM ALREADY DEFINED. STATEMENT IGNORED.
IEE252I MEMBER  CNGRP00 FOUND IN SYS1.PARMLIB
IEE712I SET CNGRP PROCESSING COMPLETE
T PFK=00   ISSUED FOR SET CON=00
VARY ,HARDCPY,CMDS,ROUT=ALL   ISSUED FOR SET CON=00
T MPF=(00)   ISSUED FOR SET CON=00
MN DSNAME   ISSUED FOR SET CON=00
D C,HC,L=Z
CNZ4100I 16.42.33 CONSOLE DISPLAY 000
CONSOLES MATCHING COMMAND: D C,HC
MSG:CURR=0    LIM=1500 RPLY:CURR=0    LIM=999   SYS=SC74     PFK=00
HARDCOPY  LOG=(SYSLOG,OPERLOG)  CMDLEVEL=CMDS
      ROUT=(ALL)
LOG BUFFERS IN USE: 0       LOG BUFFER LIMIT: 6000
IEE252I MEMBER PFKTAB00 FOUND IN SYS1.PARMLIB
IEE252I MEMBER MPFLST00 FOUND IN SYS1.PARMLIB
IEE536I PFK      VALUE 00 NOW IN EFFECT
IEE712I SET MPF  PROCESSING COMPLETE
D C,L
CNZ4100I 16.43.12 CONSOLE DISPLAY 006
CONSOLES MATCHING COMMAND: D C,L
MSG:CURR=0    LIM=1500 RPLY:CURR=0    LIM=999   SYS=SC74     PFK=00
HARDCOPY  LOG=(SYSLOG,OPERLOG)  CMDLEVEL=CMDS
      ROUT=(ALL)
NAME     TYPE    STATUS         DEFINED             MATCHED
SC74F400 MCS     INACT          SC74                SC74
SC74F480 MCS     INACT          SC74                SC74
SC74HMCS HMCS    STDBY-SC74     SC74                SC74
SMCSCON1 SMCS    INACT          *ALL                *ALL
SMCSCON2 SMCS    INACT          *ALL                *ALL
SMCSCON3 SMCS    INACT          *ALL                *ALL
SMCSCON4 SMCS    INACT          *ALL                *ALL
```

Figure 13-8 Dynamically adding an HMCS console

To activate the HMCS, log on to the HMC and select the z/OS LPAR image. Then, select the Integrated 3270 Console task listed under the Recovery task group, as shown in Figure 13-9 on page 182.

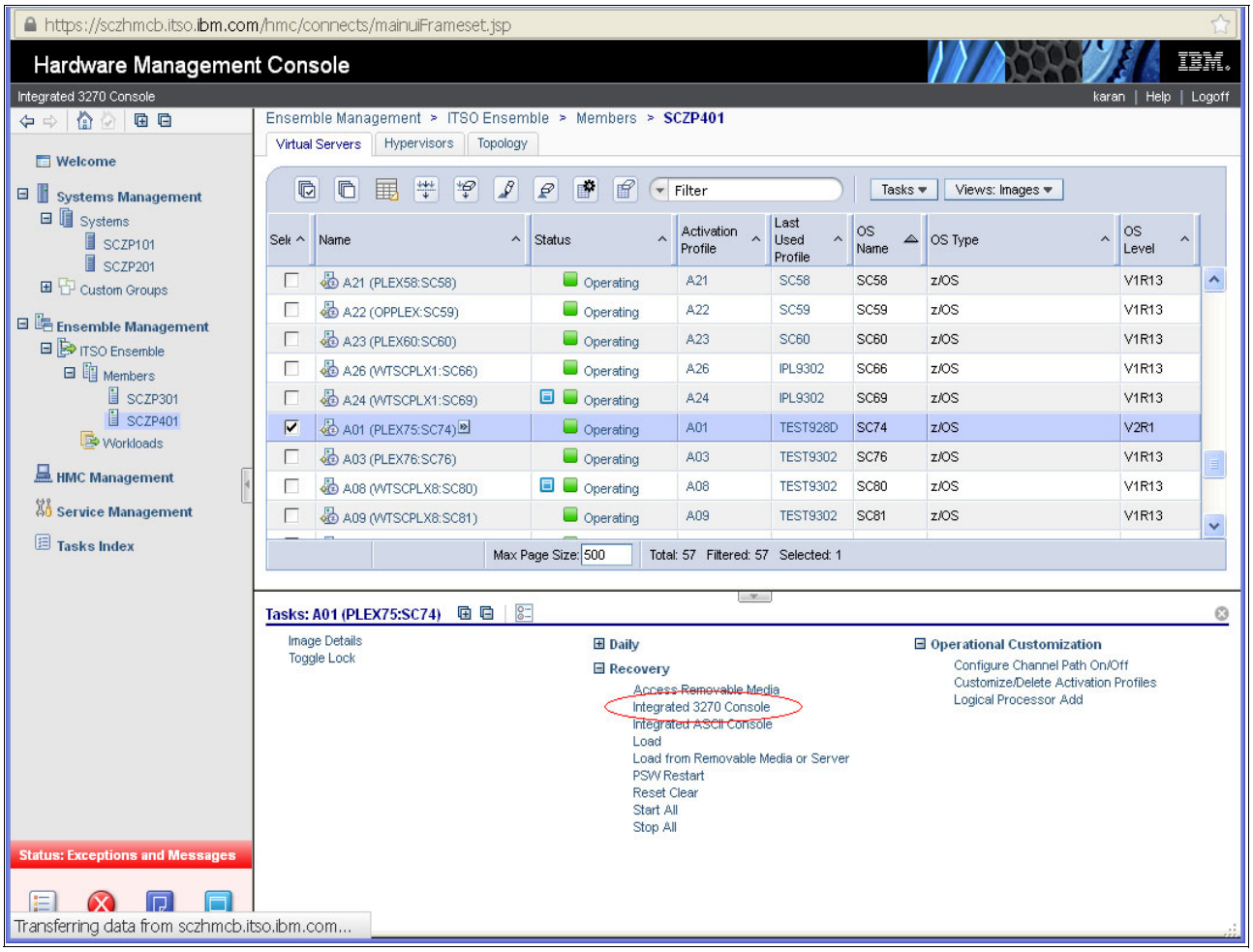


Figure 13-9 Integrated 3270 Console task on the HMC

The integrated 3270 console window opens and the console remains in the standby state (the console screen is blank). Pressing any attention-generating key on the keyboard activates it. Figure 13-10 on page 183 shows the HMCS console after pressing the ENTER key.

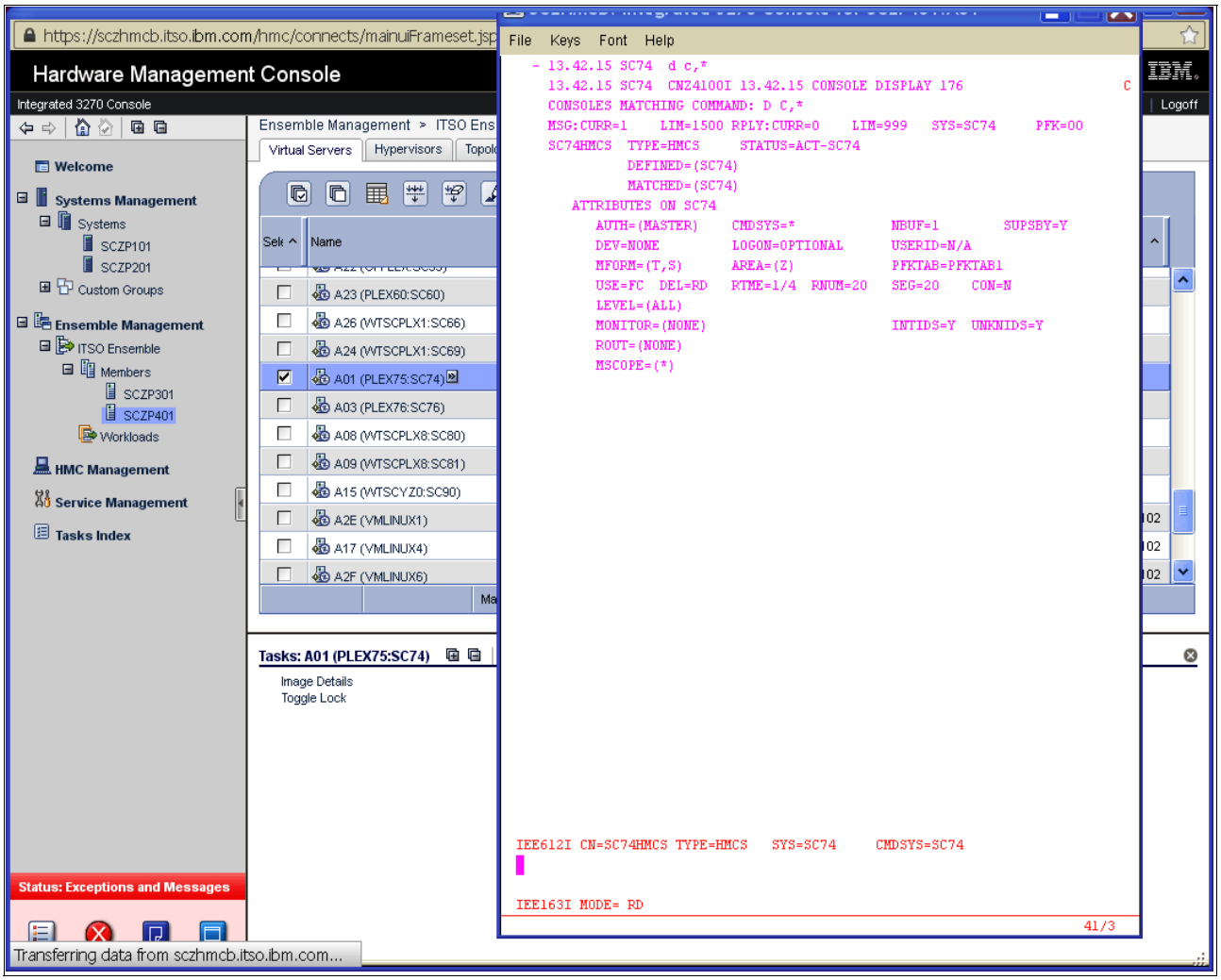


Figure 13-10 Active HMCS console on the HMC

Using HMCS during NIP

When the Integrated 3270 Console is attached to a z/OS LPAR image and z/OS is IPLed, z/OS uses the HMCS console for NIP messages. No IODF or `CONSOLxx` definitions are needed for the NIP use. This is the z/OS selection order for the console to use during NIP:

1. HMCS (Integrated 3270 Console interface in the HMC)
2. Devices that are defined in the IODF by NIPCON specifications
3. System console (operating system messages interface in the HMC)

When the NIP is finished, the HMCS console can no longer be used if there are no `CONSOLxx` definitions for it.

Standby console state

The *standby* console state is in z/OS 2.1. A console in standby state is neither active not inactive. When the console is in standby state, pressing any attention-generating key, such as ENTER, PFx, or ATTN, causes it to enter the active state. System resources are preallocated for consoles in standby state, so it enters the active state quicker than an inactive console.

The standby state is available only for MCS and HMCS consoles in full capability (FC) mode. SMCS, EMCS, subsystem consoles, the system console, printer consoles, or consoles in

status display (SD) or message stream (MS) mode are not supported. Consoles in standby state are part of the system limit of 99 active consoles when in distributed mode. Consoles in standby state are part of the 99 defined consoles per Parallel Sysplex limit when in shared mode.

The SUPSBY console attribute indicates whether the console supports standby state or not. The SUPSBY attribute is defined per console in CONSOLxx. Specify SUPSBY=Y to indicate that the console supports standby. The default is SUPSBY=N. The HMCS always runs with SUPSBY=Y. The **VARY CN(xx)** command can be used to dynamically change the SUPSBY attribute, as shown in Figure 13-11. The **VARY CN(xx)** command can also be used to change a console into standby state and back to active, as Figure 13-11 also shows. Changing a console into standby state requires update authority to the MVS.VARYSTANDBY.CN profile in the OPERCMDS class.

```
V CN(SC74F400),STANDBY
CNZ0005I VARY CN REJECTED. REASON=CONSOLE DOES NOT SUPPORT STANDBY
V CN(SC74F400),SUPSBY=Y
IEE712I VARY CN  PROCESSING COMPLETE
V CN(SC74F400),STANDBY
CNZ4303I CONSOLE SC74F400 STATUS CHANGED FROM ACTIVE TO STANDBY.
REASON: VARY COMMAND
```

Figure 13-11 Change console to standby state

To take a console out of standby state, either vary the console online or press any attention-generating key on the console. When pressing a key on the console, the system indicates that the console is changed back to active state via message CNZ4304I, which is shown in Figure 13-12.

```
CNZ4303I CONSOLE SC74F480 STATUS CHANGED FROM STANDBY TO ACTIVE.
REASON: SESSION ACTIVATED
```

Figure 13-12 Message CNZ4303I

The **D C** command is enhanced to display standby consoles, as shown in Figure 13-13.

```
D C,L
CNZ4100I 09.50.27 CONSOLE DISPLAY 805
CONSOLES MATCHING COMMAND: D C,L
MSG:CURR=0    LIM=1500 RPLY:CURR=0    LIM=999   SYS=SC74     PFK=00
HARDCOPY  LOG=(SYSLOG,OPERLOG)  CMDLEVEL=CMDS
      ROUT=(ALL)
NAME     TYPE    STATUS         DEFINED             MATCHED
SC74F400 MCS     STDBY-SC74     SC74                SC74
SC74F480 MCS     STDBY-SC74     SC74                SC74
SC74HMCS HMCS    STDBY-SC74     SC74                SC74
SMCSCON1 SMCS    INACT          *ALL                *ALL
SMCSCON2 SMCS    INACT          *ALL                *ALL
SMCSCON3 SMCS    INACT          *ALL                *ALL
SMCSCON4 SMCS    INACT          *ALL                *ALL
```

Figure 13-13 D C command

13.2.2 Installation

Define the HMCS in CONSOLxx.

13.2.3 Migration and coexistence considerations

Coexistence APAR OA37696 must be installed on systems before z/OS Version 2.1. An earlier level system without OA37696 installed cannot join a Parallel Sysplex that contains a z/OS 2.1 system.

13.3 Enhanced synchronous WTOR notification

Synchronous WTORs indicate critical system problems. When a synchronous WTOR is issued, the system is stopped until a reply is provided. Delays in replying might affect other systems in the Parallel Sysplex.

Until z/OS 2.1, when a synchronous WTOR is issued, it is displayed on one operator console at a time. If a reply is not given within two minutes, the synchronous WTOR is moved to another console. Only a reply from the console currently displaying the synchronous WTOR is acceptable. All other consoles in the system appear hung. The order by which the synchronous WTOR chooses a console to display on is determined by the **SYNCHDEST** parameter of the DEFAULT statement in CONSOLxx. When no **SYNCHDEST** is specified, all consoles in the system appear hung, and the synchronous WTOR is displayed only on the system console in the HMC.

Important: In such a case, it is difficult for the operator to determine what is going on in the system, because all consoles appear hung with no reason. The operator must notice that there is a reply waiting in the system console in the HMC and reply to it promptly. A delay in responding to the WTOR can lead to a system outage.

Starting with z/OS 2.1, synchronous WTOR processing is enhanced to better notify the operator of a pending synchronous WTOR, as the section that follows explains.

13.3.1 Invocation and use

When a synchronous WTOR is issued, processing is determined by the **SYNCHDEST** specification in CONSOLxx.

When no **SYNCHDEST** is specified in CONSOLxx, all consoles in the system are stopped and the synchronous WTOR is displayed on the system console in the HMC. Message CNZ4215I is displayed on all MCS and HMCS consoles, indicating that a synchronous WTOR is waiting for a reply. Message CNZ4251I also indicates which console in the system is displaying the WTOR. The WTOR console background changes to red to attract the operator's attention, as shown in Figure 13-14 on page 186.

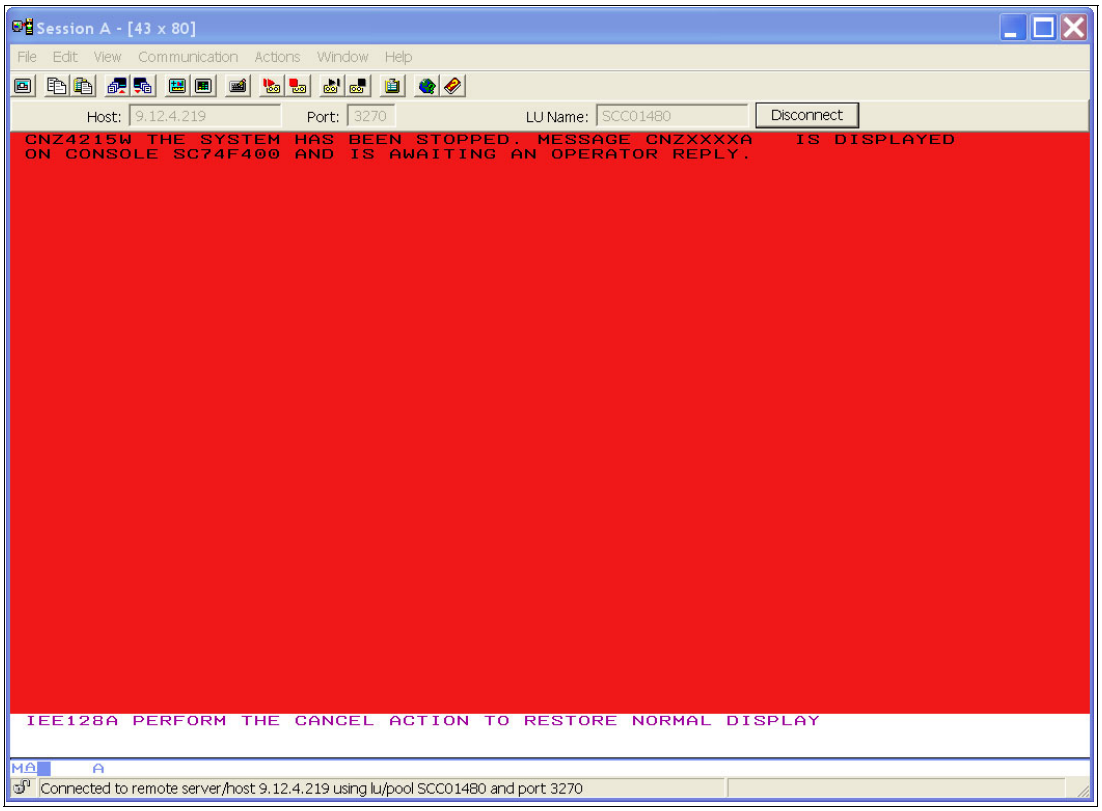


Figure 13-14 Message CNZ4251I displayed on a red background

If the HMCS is active, message CNZ4251I is displayed on it also. Notification messages are displayed only on active HMCS and MCS consoles on the same system where the WTOR is issued. To achieve the benefit of notification, each system must have at least one MCS or HMCS console active.

When **SYNCHDEST** is specified in CONSOLxx, the synchronous WTOR is displayed on the consoles that are specified in the **SYNCHDEST** group, following the order in which they are specified. All other consoles, including the HMCS, show notification message CNZ4251I.

13.3.2 Installation

Verify your **SYNCHDEST** specification in CONSOLxx and CNGRPxx. Figure 13-15 on page 187 and Figure 13-16 on page 187 show the CONSOLxx and CNGRPxx definitions that we used for testing.

```
INIT     MLIM(1500)
         RLIM(999)
         LOGLIM(6000)
         AMRF(Y)
         APPLID(SCSMCS&SYSCLONE.)
         UEXIT(N)
         MPF(00)
         PFK(00)
         CNGRP(00)
         MONITOR(DSNAME)
         CMDDELIM(<)
DEFAULT  ROUTCODE(1-16)
         RMAX(999)
         HOLDMODE(YES)
         SYNCHDEST(DCCF)
```

Figure 13-15 Specifying CNGRP and SYNCHDEST in CONSOLxx

```
GROUP NAME(DCCF)   MEMBERS(CASTOR,POLLUX,
                          SC75,
                          SC74F400,SC74F480,SC74HMCS)
```

Figure 13-16 CNGRPxx member definitions

13.4 FORCE TCB command

The **FORCE TCB** command gives you the ability to terminate an unresponsive task that holds critical system resources, without terminating the whole address space. A task that is terminated by using **FORCE TCB** is terminated with ABEND code 80D and reason code 1.

Note: This command is intended to be used at the guidance of IBM support only.

13.4.1 Invocation and use

Example 13-1 shows the syntax of the **FORCE TCB** command.

Example 13-1 FORCE TCB command syntax

```
FORCE workunit,TCB=tttttt[,RETRY=NO|YES][,A=asid]
```

The task to terminate is identified by a hexadecimal TCB address. `RETRY=YES` or `NO` indicates whether the task's recovery processing is allowed to retry. The default is `RETRY=NO`.

To issue the command, a CONTROL access level is required for the following profiles in the `OPERCMDS` class:

- MVS.FORCETCB.JOB.jobname
- MVS.FORCETCB.STC.mbrname.id
- MVS.FORCETCB.TSU.userid
- MVS.FORCETCB.DEV.device
- MVS.FORCETCB.STC.mbrname.jobname

An address space can protect itself from the **FORCE TCB** command by turning on the ASCBNOFT bit in the Address Space Control Block (ASCB). When ON, the **FORCE TCB** command is rejected when you attempt to terminate a task in that address space.

For example, the command in Example 13-2 shows how to terminate the task whose TCB address is X'5E6D90' in the JOBA1 address space.

Example 13-2 Using the FORCE TCB command

```
FORCE JOBA1,TCB=5E6D90
```

13.5 Auto-reply support for synchronous WTOR messages

Starting with z/OS 2.1, auto-reply supports automating a response to synchronous WTOR messages.

Synchronous WTOR messages must be replied to promptly. These messages (such as IEA367A) prevent the system from updating its status on the Parallel Sysplex couple data set. This can lead to Sysplex Failure Management (SFM) deciding that the system is not responding normally and removing it from the Parallel Sysplex. If you have a standard response to synchronous WTOR messages such as these, you can automate the reply to these messages to avoid delay and avoid SFM partitioning the system out of the sysplex. However, MPF and typical automation products are unable to automate a response to these messages.

13.5.1 Invocation and use

There are no external changes to auto-reply for the support. The following synchronous WTORs are added to the AUTOR00 member that is included with z/OS 2.1:

- BLW004A
- IEA015A
- IEA367A
- IEA394A
- IEA500A
- IEA502A

To enable support for synchronous WTORs, update your `AUTOR00` PARMLIB member with the one provided in z/OS 2.1. Ensure that member `AUTOR00` exists in your active PARMLIB concatenation and that `IEASYSxx` points to it by using keyword `AUTOR=xx`. During IPL, if the PARMLIB member `AUTOR00` exists, auto-reply processing is activated.

Use the **D AUTOR,POLICY** command to verify that the WTOR message IDs are defined in the auto-reply policy. Example 13-3 shows an example.

Example 13-3 Verification of the auto-reply policy definition

```
D AUTOR,POLICY
CNZ2603I 09.56.40 AUTOR POLICY 064
POLICY ACTIVATED AT 12.00.31 ON 04/20/2013  NOTIFYMSGS(HC)
FROM PARMLIB MEMBERS 00
--MSG ID-- DELAY MEM RATE ----REPLY TEXT----
ANTU2220D   60S   00 ---  C
ANTX8925A   60S   00 ---  N
ANTX8926A   60S   00 ---  N
```

```
ANTX8942A    6OS    OO ---  N
... messages deleted ...
BLWOO4A      6OS    OO ---  REPAIR
... messages deleted ...
IEAO15A       2M    OO ---  RETRY
... messages deleted ...
```

By using messages, the system indicates whether a synchronous WTOR is managed by the auto-reply policy or not. The console that the synchronous WTOR is issued to displays the message (also shown in Figure 13-17):

```
CNZ2610I THE FOLLOWING SYNCHRONOUS WTOR IS BEING MANAGED BY AUTO-REPLY
```

By comparison, this was the previous message:

```
IEE107I THE FOLLOWING MESSAGE IS ISSUED BY SYNCHRONOUS WTO/R SERVICE
```

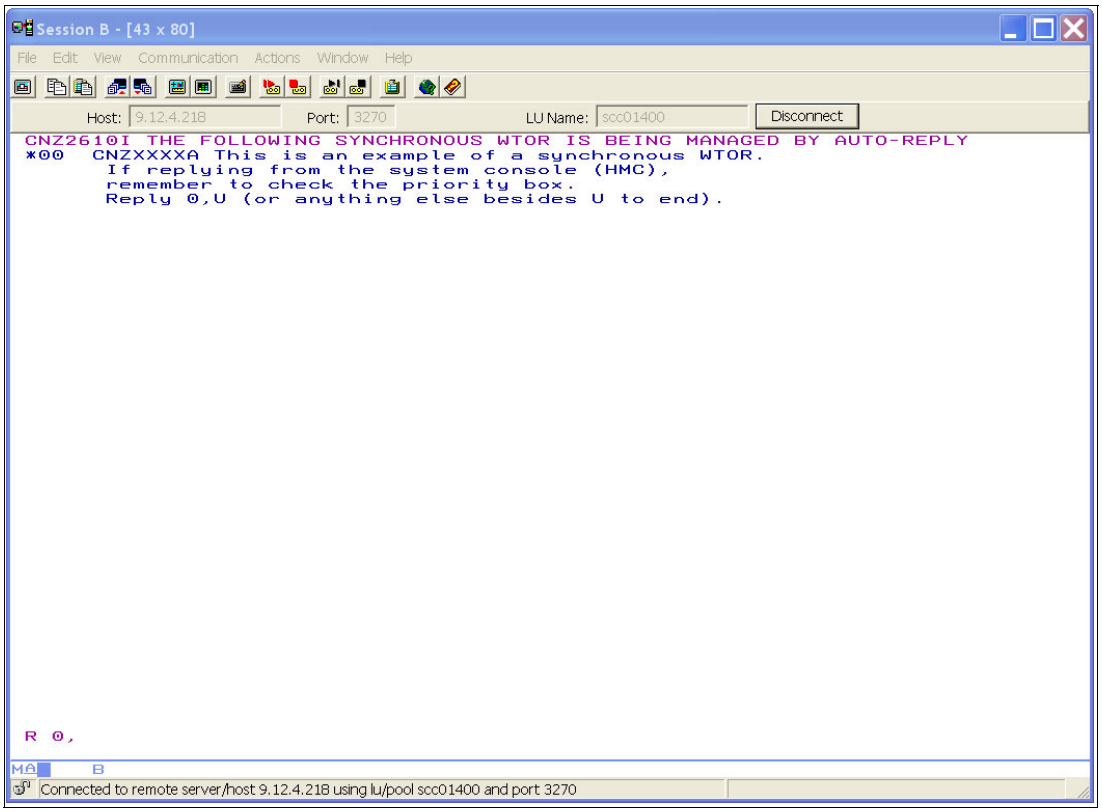


Figure 13-17 Message CNZ2610I

In addition, all other active consoles display the following message, as shown in Figure 13-18 on page 190:

```
CNZ4215W THE SYSTEM HAS BEEN STOPPED. MESSAGE msgid IS DISPLAYED
ON CONSOLE conname AND IS AWAITING AN OPERATOR REPLY.
THE MESSAGE IS BEING MANAGED BY AUTO-REPLY.
```

The previous message, for comparison:

```
CNZ4215W THE SYSTEM HAS BEEN STOPPED. MESSAGE msgid IS DISPLAYED
ON CONSOLE conname AND IS AWAITING AN OPERATOR REPLY.
```

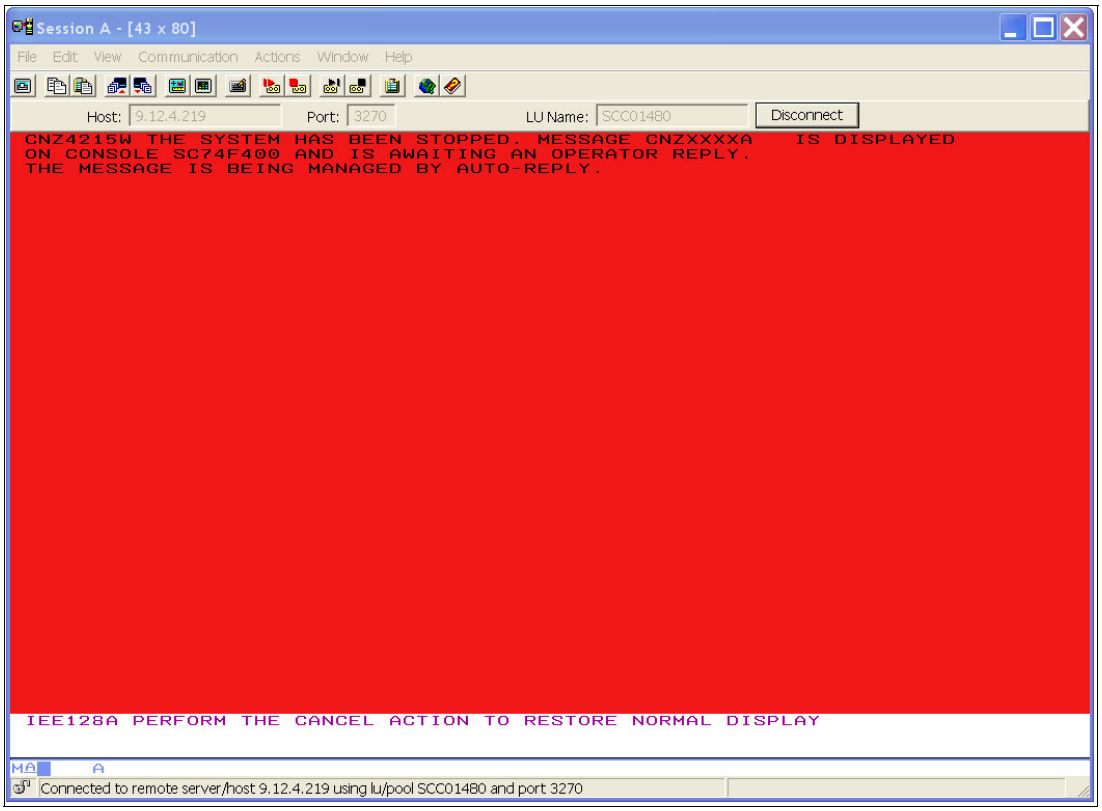


Figure 13-18 Message CNZ4215I

Figure 13-19 on page 191 shows a synchronous WTOR message that was replied to automatically by auto-reply, as indicated by message CNZ2602I.

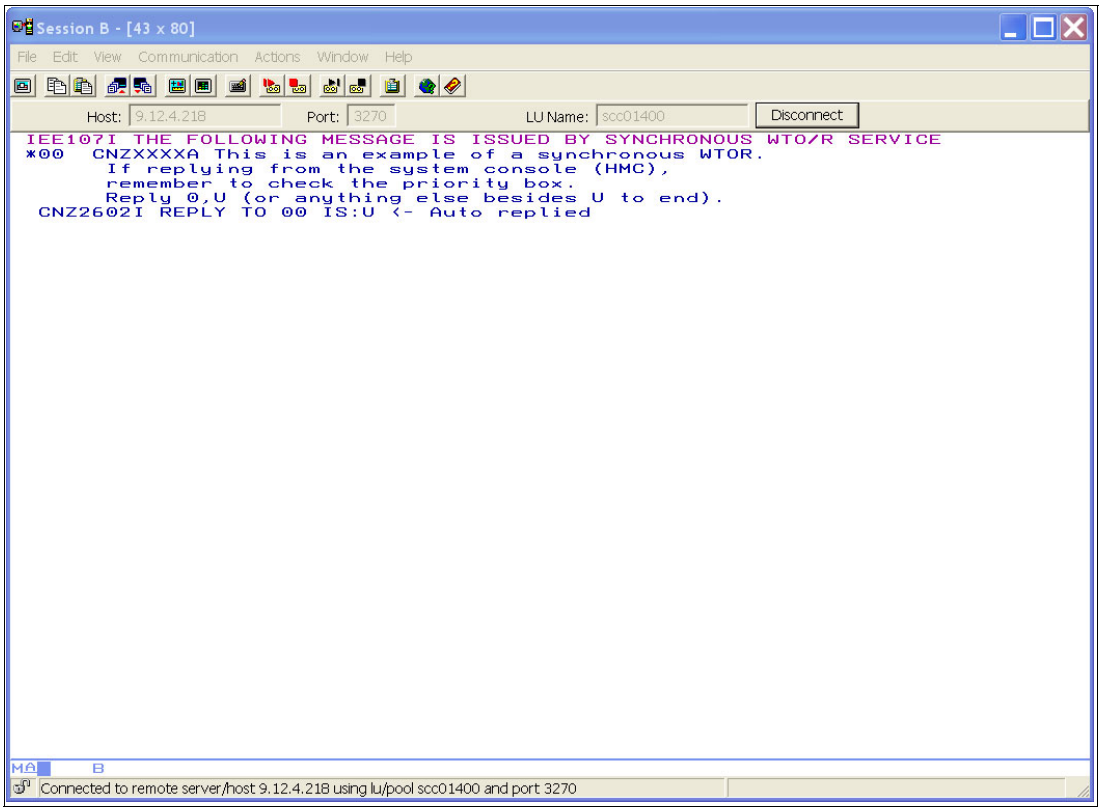


Figure 13-19 Auto-reply to a synchronous WTOR

13.5.2 Installation

Update your AUTOR00 PARMLIB member with the one provided in z/OS 2.1. If auto-reply is to be used, ensure that the AUTOR00 member exists in your active PARMLIB concatenation.

13.6 Auto-reply rate limit

Auto-reply processing monitors for the possibility of “runaway” scenarios. A runaway WTOR can occur when an incorrect reply is provided to the AUTORxx PARMLIB entry when keyword DELAY is set to 0. In such a case, for each invalid response, the WTOR message is reissued. When auto-reply processing detects the same WTOR message 20 times within a second, it does not automatically reply to the next one.

However, more than 20 WTORs of the same message ID might be valid in some environments. Auto-reply is enhanced in z/OS 2.1 to provide the ability to specify the particular rate limit per WTOR message in the AUTORxx PARMLIB member.

13.6.1 Invocation and use

A RATELIMIT(nnn) option is added on the MSGID statement in **AUTORxx**. RATELIMIT(nnn) is allowed only when DELAY(0S) or DELAY(0M) are also specified. For example, the auto-reply policy that is shown in Example 13-4 on page 192 is configured to reply to 30 WTORs per second for message ID ABD123D. If more than 30 ABD123D messages are issued in one second, the system responds to only the first 30 messages.

Example 13-4 Using the RATELIMIT keyword in AUTORxx

```
Msgid(ABD123D) Delay(0S) Reply(Y) RateLimit(030)
```

The **D AUTOR,POLICY** command is updated to display the rate limit for each message ID. When no rate limit is defined, the rate is displayed as --- (three dashes or hyphens). See Example 13-3 on page 188 for an example.

13.6.2 Migration and coexistence considerations

Coexistence APAR OA39287 has been created to ignore the RATELIMIT keyword in z/OS V1.13 and z/OS 2.1.

13.7 Documentation and reference

For complete documentation and reference of the described enhancements, see the following publications in the IBM Knowledge Center:

- *z/OS V2R1 MVS Initialization and Tuning Reference*, SA23-1380

  http://ibm.co/TOcA6f
- *z/OS V2R1 MVS Planning: Operations,* SA23-1390

  http://ibm.co/1n8w0i3
- *z/OS V2R1 MVS System Commands,* SA38-0666

  http://ibm.co/1sohwe8

14

Generic Tracker Facility

This chapter includes the following sections that describe the Generic Tracker, which replaces the One-byte console ID tracker in IBM z/OS Version 2.1:

- 14.1, “Generic Tracker” on page 194
- 14.2, “Documentation and reference” on page 200

© Copyright IBM Corp. 2014. All rights reserved.

14.1 Generic Tracker

The One-byte console ID tracker started as a single-purpose migration aid. Over time, other z/OS components used it to track events that aid in their migration efforts. The One-byte console ID tracker was not designed for general purpose use, so it soon reached its limits.

The Generic Tracker allows users to track more information and is more flexible when it comes to working with the tracker. It is a general tool to assess migration needs and to assess use of product functions.

By using the Generic Tracker during migration, you can find users of interfaces that are obsolete or soon to be. Use or adoption of any service, especially functions, can be assessed through tracked records also.

14.1.1 Invocation and use

To use this service, the program that is being tracked must be modified. The client program calls a service to track use by using the Generic Tracker. The tracking facility stores the track data for later retrieval.

To track an event, use the GTZTRACK system service. The caller of GTZTRACK provides the event address and other track information. The track information should provide enough data to understand the track record later. The tracking facility uses the event address for program name lookup and to calculate the offset in the program. The GTZZTRK mapping macro provides EQUATES for return and reason codes and other constants that are related to the GTZTRACK service. For a complete description of the GTZTRACK macro, see *z/OS V2R1.0 MVS Programming: Assembler Services Reference ABE-HSP* (see 14.2, “Documentation and reference” on page 200 for a link).

The Assembler program that is shown in Example 14-1 is a simple example of using the GTZTRACK macro.

Example 14-1 Using GTZTRACK

```
TESTGTZ  CSECT
TESTGTZ  AMODE    31
TESTGTZ  RMODE    ANY
         BAKR     R14,0
         BASR     R12,0
         USING    *,R12
*
         EPSW     R4,R5
         STM      R4,R5,MY_PSW
*
TRACK@   GTZTRACK OWNER=MY_OWNER,                                      C
               SOURCE=MY_SRC,                                          C
               EVENTDESC=MY_DESC,                                      C
               EVENTDESCLEN=MY_DESCL,                                  C
               EVENTDATA=ALLZERO,                                      C
               EVENTADDR=MY_ADDR,                                      C
               EVENTPSW8=MY_PSW,                                       C
               EVENTASID=HOME
*
         CIJE     R15,GTZTRACKRC_OK,EXIT
         WTO      'GENERIC TRACKER ERROR'
```

```
EXIT     DS       0H
         PR
*
MY_OWNER DC       CL16'ITSOSAMPLE'
MY_SRC   DC       CL8'ITSOSAMP'
MY_DESC  DC       C'SAMPLE GENERIC TRACKER EVENT'
MY_DESCL DC       AL2(*-MY_DESC)
MY_ADDR  DC       A(0),A(TRACK@)
MY_PSW   DS       XL8
*
         GTZZTRK  ,
         YREGS
         END      TESTGTZ
```

Controlling the Generic Tracker

Use the **SETGTZ** command to dynamically change various Generic Tracker parameters. To turn tracking on or off, use the **SETGTZ TRACKING** command. An example is shown in Example 14-2.

Example 14-2 Using SETGTZ TRACKING

```
SETGTZ TRACKING=ON
GTZ1105I SETGTZ TRACKING PROCESSING IS COMPLETE
```

The **SETGTZ CLEAR** command is used to remove and reset various tracking facility data, settings, and statistics. Example 14-3 shows an example of clearing the collected tracking data.

Example 14-3 Using SETGTZ CLEAR

```
SETGTZ CLEAR=TRACKDATA
GTZ1103I SETGTZ CLEAR (297) ACCEPTED FOR PROCESSING
GTZ1104I SETGTZ CLEAR (297) PROCESSING IS COMPLETE.
```

Use the **SETGTZ EXCLUDE** command to add an EXCLUDE statement to the facility. The EXCLUDE statement consists of a filter for tracked instances. Any future **GTZTRACK** request with data that matches the filter is not recorded, and any recorded tracked instances that match are cleared from the facility. Example 14-4 shows how to exclude all events with descriptions that start with the text “SMS-I:3 DEVTYPE” that was issued in any cross memory environment by any program if it is not a z/OS UNIX executable program.

Example 14-4 Using SETGTZ EXCLUDE

```
SETGTZ EXCLUDE(EVENTDESC='SMS-I:3 DEVTYPE*' HOMEJOB=* PROGRAMTYPE=NOPA
TH PROGRAM=*)
GTZ1103I SETGTZ EXCLUDE (388) ACCEPTED FOR PROCESSING
GTZ1104I SETGTZ EXCLUDE (388) PROCESSING IS COMPLETE.
```

Use the **SETGTZ DEBUG** command to add a DEBUG statement to the facility. DEBUG statements are intended as a diagnostic aid to further debug certain tracked instance occurrences. The DEBUG statement has parameters to describe the debug action and which tracked instances trigger this action (via a filter). Any future GTZTRACK request with data that matches the filter will run the requested debug action. An EXCLUDE statement does not prohibit a DEBUG statement from triggering, but just from being recorded. The DEBUG command is also useful in cases where the Generic Tracker cannot provide all data about a tracker instance, such as when the program name is unknown.

If there is more than one debug statement that matches a GTZTRACK request, the most recently entered debug statement is used. To be able to do that, the system tracks the order in which debug statements are specified. It is important to be as specific as possible in your DEBUG filter and to avoid similar statements, such as having different reason code or debug actions, to avoid unpredictable behavior.

The debug action can be either to ABEND the caller of GTZTRACK or to initiate a memory dump when GTZTRACK is called. In either case, the ABEND system completion code is SE77. An example of using the **SETGTZ DEBUG** command is shown in Example 14-5.

Example 14-5 Using SETGTZ DEBUG

```
SETGTZ DEBUG=(ACTION=DUMP,REASON=1,LIMIT=5,OWNER=ITSOSAMPLE)
GTZ1103I SETGTZ DEBUG (429) ACCEPTED FOR PROCESSING
GTZ1104I SETGTZ DEBUG (429) PROCESSING IS COMPLETE.
D GTZ,DEBUG
GTZ1004I  14.30.09 GTZ DEBUG 583
FOUND 1 MATCHING DEBUG STATEMENT(S)
------------------------------------------------------------------------
 STATEMENT:     1                   ORIGIN:        COMMAND
 ACTION:        DUMP                REASON:        x0001
 ACTION COUNT:  0                   LIMIT:         5
 OWNER:         ITSOSAMPLE
$HASP100 PELEGASM ON INTRDR                            FROM TSU05357
PELEG
IRR010I  USERID PELEG    IS ASSIGNED TO THIS JOB.
IEF677I WARNING MESSAGE(S) FOR JOB PELEGASM ISSUED
ICH70001I PELEG    LAST ACCESS AT 14:28:18 ON MONDAY, APRIL 29, 2013
$HASP373 PELEGASM STARTED - INIT 1    - CLASS A        - SYS SC74
IEF450I PELEGASM G ASMACL - ABEND=SE77 U0000 REASON=00000001
$HASP395 PELEGASM ENDED
$HASP309 INIT 1    INACTIVE ******** C=ABCDEFG12345
SE '14.30.14 JOB05394 $HASP165 PELEGASM ENDED AT WTSCPLX7 - ABENDED
SE77 U0000',LOGON,USER=(PELEG)
```

In addition, a **SETGTZ DIAGNOSE** command is available to aid in collecting diagnostic information.

Note: Use the **SETGTZ DIAGNOSE** command only as directed and requested by IBM service personnel.

For further description of the **SETGTZ** commands, see *z/OS V2R1.0 MVS System Commands* (see 14.2, “Documentation and reference” on page 200 for a link).

The GTZ address space

The Generic Tracker is automatically started during IPL by using the GTZ procedure in SYS1.PROCLIB. The procedure starts the GTZINIT load module. If the Generic Tracker fails, it can be restarted by using the **START GTZ,SUB=MSTR** command. Tracked data is recorded in the GTZ address space storage. You can use the MEMLIMIT= keyword in the procedure to control how much data the tracking facility is able to store.

The only supported, optional parameter for GTZINIT is **PARM='GTZPRM=*NONE'** to let the Generic Tracker start without reading in any GTZPRMxx PARMLIB members. By default, the tracker uses the PARMLIB members at startup, as identified by the GTZ system parameter.

GTZPRMxx PARMLIB member

GTZPRMxx PARMLIB members contain the following statements that control the behavior of the Generic Tracker:

EXCLUDE	Add an EXCLUDE statement.
TRACKING	Enable or disable tracking.
CLEAR	Reset or clear settings, statistics, and data.
DEBUG	Add a DEBUG statement.
DIAGNOSE	Use for debugging and servicing.

Use one of these statements to specify to specify which list of GTZPRMxx members the tracker is to consider:

- The system parameter **GTZ** on the reply to message IEA101A at IPL time, via **GTZ={xx|(xx,...,zz)}**
- The system parameter **GTZ** in IEASYSxx, via **GTZ={xx|(xx,...,zz)}**
- The operator command **SET GTZ={xx|(xx,...,zz)}**

Also see SYS1.SAMPLIB(GTZPRM00) for a working GTZPRMxx PARMLIB member that contains recommended statements (mainly exclusions in the form of EXCLUDE statements) as known to IBM at the time that release becomes available.

For the most current version of the GTZPRM00 PARMLIB member, check the z/OS Downloads web page:

http://www.ibm.com/systems/z/os/zos/downloads/

Displaying track data

The **DISPLAY GTZ,STATUS** command is used to display an overview of the Generic Tracker's status, recorded tracking data, and configuration settings. Example 14-6 shows a sample command.

Example 14-6 Using D GTZ,STATUS command

```
D GTZ,STATUS
GTZ1001I  14.31.59 GTZ STATUS 461
 TRACKING:  ENABLED          2013-04-29 11:05:53
 TRACKED:   UNIQUE=1         TOTAL=15
 EXCLUDE:   DEFINED=45       APPLIED=8
 DEBUG:     DEFINED=1        APPLIED=1
 GTZPRMXX:  00
 MEMORY:    99% AVAILABLE
 CLEARED:   ALL TRACKDATA EXCLUDE DEBUG
```

To display details for all or a subset of the recorded tracked data from previous **GTZTRACK** requests, use the **DISPLAY GTZ,TRACKDATA** command. The command supports filters to display only recorded instances by specific owner, source, description, program, job name, and more. An example is shown in the output in Figure 14-1 on page 198. We requested a display of only events with a job name that starts with the letter P.

```
D GTZ,TRACKDATA(HOMEJOBNAME=P*)
GTZ1002I  12.40.20 GTZ TRACKDATA 394
FOUND 3 MATCHING TRACKED INSTANCE(S)
------------------------------------------------------------------------
 INSTANCE:      1                   COUNT:         5
 EVENTDESC:     'SMS-I:3 DEVTYPE'
 OWNER:         IBMCNZ              SOURCE:        CNZTRKR
 EVENTDATA:     x0000000000000000   x0000000000000003
 PROGRAM:       ASMA90              PROGRAMOFFSET: x000000000005D4F4
 HOMEJOB:       PELEGASM            HOMEASID:      x002A
 EVENTJOB:      PELEGASM            EVENTASID:     x002A
 AUTHORIZED:    YES                 FIRST TIME:    2013-04-29 12:13:23
------------------------------------------------------------------------
 INSTANCE:      2                   COUNT:         5
 EVENTDESC:     'SMS-I:3 DEVTYPE'
 OWNER:         IBMCNZ              SOURCE:        CNZTRKR
 EVENTDATA:     x0000000000000000   x0000000000000003
 PROGRAM:       IEWBIND             PROGRAMOFFSET: x000000000018F1E6
 HOMEJOB:       PELEGASM            HOMEASID:      x002A
 EVENTJOB:      PELEGASM            EVENTASID:     x002A
 AUTHORIZED:    YES                 FIRST TIME:    2013-04-29 12:13:23
------------------------------------------------------------------------
 INSTANCE:      3                   COUNT:         4
 EVENTDESC:     'SAMPLE GENERIC TRACKER EVENT'
 OWNER:         ITSOSAMPLE          SOURCE:        ITSOSAMP
 EVENTDATA:     x0000000000000000   x0000000000000000
 PROGRAM:       TESTGTZ             PROGRAMOFFSET: x000000000000000E
 HOMEJOB:       PELEGASM            HOMEASID:      x002A
 EVENTJOB:      PELEGASM            EVENTASID:     x002A
 AUTHORIZED:    NO                  FIRST TIME:    2013-04-29 12:13:23
```

Figure 14-1 Using the D GTZ,TRACKDATA command

As Figure 14-2 shows, use **DISPLAY GTZ,EXCLUDE** to display details for all currently active EXCLUDE statements that are specified by the **SETGTZ EXCLUDE** command or by corresponding GTZPRMxx PARMLIB member statements.

```
D GTZ,EXCLUDE
GTZ1003I  15.57.21 GTZ EXCLUDE 201
FOUND 45 MATCHING EXCLUDE STATEMENT(S)
------------------------------------------------------------------------
 STATEMENT:     1                   ORIGIN:        GTZPRM00
 EVENTDESC:     'CONVCON'
 PROGRAMTYPE:   NOPATH              PROGRAM:       ITPENTER
------------------------------------------------------------------------
 STATEMENT:     2                   ORIGIN:        GTZPRM00
 EVENTDESC:     'MCSOPER: *'
 PROGRAMTYPE:   NOPATH              PROGRAM:       AULOAD
------------------------------------------------------------------------
... messages deleted ...
```

Figure 14-2 Using D GTZ,EXCLUDE command

The **DISPLAY GTZ,DEBUG** command (Figure 14-3) is used to display details for all currently active DEBUG statements that are specified by the **SETGTZ DEBUG** command or corresponding GTZPRMxx PARMLIB member statements.

```
D GTZ,DEBUG
GTZ1004I 23.27.46 GTZ DEBUG
FOUND 1 MATCHING DEBUG STATEMENT(S)
-------------------------------------------------------------------
STATEMENT:      1                   ORIGIN: COMMAND
ACTION:         ABEND               REASON: x0064
ACTION COUNT:   2                   LIMIT: NOLIMIT
OWNER:          IBM_GTZ_FCT
```

Figure 14-3 Using D GTZ,DEBUG

Printing track data

The **GTZPRINT** print utility is used to write status and selected tracking and configuration data to the specified SYSOUT. The print utility is more useful than the **D GTZ,TRACKDATA** command when many events exists. Use the **D GTZ,STATUS** command to check how many recorded track events are currently in GTZ memory. You might need to use a filter to display only the track events that interest you.

The required SYSIN DD in `RECFM=FB` and `LRECL=80` format allows you to specify one or more `PRINT` statements. `PRINT` statements are equivalent to the **DISPLAY GTZ** command in syntax, except for using a `PRINT` prefix rather than the `GTZ` prefix.

For example, to print the tracking facility's status, the current tracked events, the current `EXCLUDE` and `DEBUG` filters, use these statements:

```
PRINT STATUS
PRINT TRACKDATA
PRINT EXCLUDE
PRINT DEBUG
```

The required SYSOUT DD specifies where to write the output for the commands. The required attributes for SYSOUT are `DSORG=PS`, `RECFM=FB LRECL=80`. The same Resource Access Control Facility (RACF) authorization as for the GTZQUERY service is required for the PRINT utility: READ access to the `XFACILIT` class resource GTZ.sysname.QUERY.

In addition, you can use the GTZQUERY system service to write your own analysis tool or health check. The GTZSHCK member in SYS1.SAMPLIB shows how to use the GTZQUERY macro from a METAL C program.

14.1.2 Installation

The Generic Tracker is part of z/OS 2.1. The GTZ address space starts automatically during IPL. However, by default, tracking is disabled and no GTZPRMxx member is selected.

14.1.3 Migration and coexistence considerations

The One-byte console ID tracker interface, the CNZTRKR macro, is changed to route all data to the Generic Tracker by using a **GTZTRACK** request. For all **CNZTRKR** calls, the following routing applies:

- `TRPL_Track_Info` value is routed to EVENTDESC
- `TRPL_Track_Value` value is routed to EVENTDATA
- `TRPL_Violators_Addr` address is routed to EVENTADDR
- `TRPL_Dont_ABEND` parameter is routed to NOABEND

In addition, the owner is set to IBMCNZ and the source is set to CNZTRKR.

The **`DISPLAY OPDATA,TRACKING`** and the **`SET CNIDTR=`** commands are not available in z/OS 2.1.

14.2 Documentation and reference

For complete description of the Generic Tracker, see the following publications:

- *z/OS V2R1.0 MVS Programming: Assembler Services Reference, Volume 1 (ABE-HSP), SA23-1369*

  `http://ibm.co/1zkxMSa`

- *z/OS V2R1.0 MVS System Commands, SA38-0666*

  `http://pic.dhe.ibm.com/infocenter/zos/v2r1/index.jsp`

- *z/OS V2R1.0 Diagnosis Tools and Service Aids,* GA32-0905, in the z/OS V2R1 Elements and Features Library (PDF files)

  `http://www-03.ibm.com/systems/z/os/zos/library/bkserv/v2r1pdf/`

15

Job Entry Subsystem 2 (JES2)

This chapter describes the JES2 enhancements, functions, and features that were introduced with IBM z/OS Version 2 Release 1 (2.1):

- 15.1, “In-stream symbolic substitution” on page 202
- 15.2, “Passing symbols by using the internal reader” on page 204
- 15.3, “JES Symbol Service” on page 206
- 15.4, “Requesting job notification through ENF 78” on page 207
- 15.5, “8-character job classes” on page 208
- 15.6, “SAF controls on input class use” on page 211
- 15.7, “JES2-related enhancements to JCL” on page 212
- 15.8, “Improved processing of input phase errors” on page 215
- 15.9, “Interpretation with conversion” on page 216
- 15.10, “Controlling where a job converts” on page 218
- 15.11, “Displaying JES2 initialization information” on page 218
- 15.12, “Performance improvements for SYSOUT processing” on page 219
- 15.13, “SSI 85 - Unauthorized services for spool access” on page 220
- 15.14, “Documentation and reference” on page 222

© Copyright IBM Corp. 2014. All rights reserved.

15.1 In-stream symbolic substitution

Symbolic substitution in the job control language (JCL) stream of a job is performed by the JCL converter during the processing of JCL statements. For programming flexibility and efficiency, starting with z/OS 2.1, symbolic substitution is also supported for data that is contained within in-stream JCL data sets. When an in-stream JCL data set is read by the JES, in-stream symbolic substitution is performed.

There are three types of symbols that can be used for in-stream substitution:

JCL symbols By default, JCL symbols are available only to the job converter and they are lost by the time the job runs. However, by using the `EXPORT` and `SET` JCL statements, JCL symbols can be made available in the job execution phase. Any inherited symbols that are defined by the **`EXPORT SYMLIST=`** parameter are implicitly exported.

Any JCL symbols that are inherited from a submitting job through the internal reader `SYMLIST` facility are implicitly exported. Exported JCL symbols can be accessed during the job execution phase by using the JCL Symbol Service (IEFSJSYM) or the JES Symbol Service (IAZSYMBL). The JCL Symbol Service is documented in *z/OS MVS Programming: Authorized Assembler Services Reference EDT-IXG*.

JES symbols JES symbols are a type of symbols that are introduced in z/OS 2.1. JES symbols are dynamic symbols that can be managed and manipulated by using the JES Symbol Service (IAZSYMBL). The JES Symbol Service is documented in *z/OS JES Application Programming*.

System symbols System symbols are specific to the MVS system. System symbols are defined in the `IEASYMxx` member of SYS1.PARMLIB and are described in the *z/OS MVS Initialization and Tuning Reference* (see “Related publications” on page 537 for a link).

15.1.1 Invocation and use

For JES2 to substitute in-stream symbols, all symbols that are used in the in-stream data set must be exported by using the EXPORT JCL statement. The EXPORT JCL statement makes a symbol available to the job execution phase. During the job execution phase, JES2 substitutes symbols as the program reads records from the in-stream data set. The syntax of the EXPORT statement is shown in Example 15-1.

Example 15-1 Syntax of the EXPORT JCL statement

```
EXPORT SYMLIST=(symbol1,symbol2,symbol3,...)
```

In addition, specifying **`EXPORT SYMLIST=*`** indicates that all defined symbols are exported.

The type of symbol substitution that is used for the in-stream data is controlled by the `SYMBOLS` keyword that is coded on the DD statement that defines the in-stream data set. Without the `SYMBOLS` keyword, JES does not substitute symbols, so the application interprets the data as it is entered in the in-stream data set. Table 15-1 on page 203 shows the definitions of the symbols.

Table 15-1 SYMBOLS keyword definitions

SYMBOLS= keyword	Description
SYMBOLS=JCLONLY	Names of JCL symbols and JES symbols found in the in-stream data set are replaced with their values.
SYMBOLS=EXECSYS	Substitution follows the SYMBOLS=JCLONLY rule. System symbols that are defined on the system during job execution can be used in the in-stream data.
SYMBOLS=CNVTSYS	Substitution follows the SYMBOLS=EXECSYS rule, with the exception that system symbols used for substitution are taken not from the system where the job is running but from the system where the job has undergone JCL conversion.

The symbols have the values that they had at the time of JCL conversion. The syntax rules are the same as the JCL symbol substitution by converter rules.

Example 15-2 shows an example of using the EXPORT JCL statement and the `SYMBOLS` keyword on the DD statement.

Example 15-2 Using the EXPORT JCL statement

```
//EXPJOB01 JOB ACCNT#,MSGLEVEL=(1,1),NOTIFY=&SYSUID,MSGCLASS=X
//*
//         EXPORT SYMLIST=(MEM)
//         SET    MEM=ASMP1
//*
//COPYMEM  EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DISP=SHR,DSN=PELEG.SRC
//SYSUT2   DD DISP=SHR,DSN=PELEG.SRC2
//SYSIN    DD *,SYMBOLS=JCLONLY
 COPY INDD=SYSUT1,OUTDD=SYSUT2
 SELECT MEMBER=&MEM
/*
//
```

We examined the job data sets on the spool after submitting the job. As you can see in Figure 15-1, the value of the *&MEM* variable was not substituted during JCL conversion.

```
 COPY INDD=SYSUT1,OUTDD=SYSUT2
 SELECT MEMBER=&MEM
```

Figure 15-1 SYSIN in-stream data set

However, as Figure 15-2 on page 204 shows, the value of the `ASMP1` value was substituted for the *&MEM* variable during the execution phase of the job.

```
                                          IEBCOPY MESSAGES AND CONTROL STATE
IEB1135I IEBCOPY  FMID HDZ2210  SERVICE LEVEL NONE     DATED 20130308 DFSMS
IEB1035I PELEGEXP  COPYMEM  08:29:03 TUE 16 APR 2013 PARM=''
 COPY INDD=SYSUT1,OUTDD=SYSUT2
 SELECT MEMBER=ASMP1
IEB1013I COPYING FROM PDS   INDD=SYSUT1   VOL=BH5ST1 DSN=PELEG.SRC
IEB1014I           TO PDS  OUTDD=SYSUT2   VOL=BH5ST6 DSN=PELEG.SRC2
IEB159I NO MEMBERS COPIED FROM INPUT DATA SET REFERENCED BY SYSUT1
IEB144I THERE ARE 28 UNUSED TRACKS IN OUTPUT DATA SET REFERENCED BY SYSUT2
IEB147I END OF JOB - 0 WAS HIGHEST SEVERITY CODE
```

Figure 15-2 SYSPRINT data set

15.1.2 Migration and coexistence considerations

When the EXPORT statement or the SYMBOLS and SYMLIST keywords are used in a job, the job becomes ineligible for processing on systems that run z/OS versions earlier than V2.1.

15.2 Passing symbols by using the internal reader

The internal reader in z/OS 2.1 supports passing symbols that are defined in one job to the jobs that it submits through the internal reader.

15.2.1 Invocation and use

The parent job uses the SYMLIST= keyword of the DD JCL statement to list the symbols that are passed to the submitted job. The submitted job is not required to export the inherited symbols. Symbols are implicitly exported in the submitted job. This allows the parent job and the submitted job to use a consistent set of symbols. The internal reader supports passing both JCL symbols and JES symbols.

Example 15-3 shows using the SYMLIST= keyword to pass the *&MEM* variable from parent job, SYMBJOB1, to the submitted job, SYMBJOB2. The SYMBJOB2 job is not required to use the EXPORT statement to export the &MEM symbol. You must still use the SYMBOLS= keyword in the SYSIN DD statement to tell JES2 to substitute symbols during the execution phase of the job.

Example 15-3 Example of passing symbols using the internal reader

```
//SYMBJOB1 JOB ACCNT#,MSGLEVEL=(1,1),NOTIFY=&SYSUID,MSGCLASS=X
//*
//         EXPORT SYMLIST=MEM
//         SET MEM=ASMP1
//*
//GENER    EXEC PGM=IEBGENER
//SYSUT1   DD DATA,DLM=$$
//SYMBJOB2 JOB ACCNT#,MSGLEVEL=(1,1),NOTIFY=&SYSUID,MSGCLASS=X
//COPYMEM  EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DISP=SHR,DSN=PELEG.SRC
//SYSUT2   DD DISP=SHR,DSN=PELEG.SRC2
//SYSIN    DD *,SYMBOLS=JCLONLY
 COPY INDD=SYSUT1,OUTDD=SYSUT2
```

```
 SELECT MEMBER=&MEM
/*
//
$$
//SYSUT2   DD SYSOUT=(A,INTRDR),SYMLIST=(MEM)
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//
```

Job correlator

The job correlator is a job attribute that is associated with every job on a system to uniquely identify a job. The job correlator is a 64-character printable value, consisting of two strings that are separated by a colon character (**:**). The first string is a 31-character system value, which applications must always treat as a single value. The second string is an optional 32-character user-defined value. To define the user portion of the job correlator, use the JES Symbol Service to assign a value to the SYS_CORR_USRDATA symbol before the job is submitted. The SYS_CORR_USRDATA symbol value must comply with the following rules:

- The value must be 0 - 32 characters in length (an empty value is valid and equivalent to a value of all blank characters).
- The first character must be from the subset A - Z (capitals only), @ (*at* character), # (number character), or $ (dollar sign character).
- Subsequent characters must be from the subset A - Z (capitals only), 0 - 9 (numerics), @ (*at* character), # (number character), $ (dollar sign character), and _ (underscore character).
- Embedded blank characters are not supported.

Example 15-4 shows a sample.

Example 15-4 Job correlator

```
//SYMBJOB1 JOB ACCNT#,MSGLEVEL=(1,1),NOTIFY=&SYSUID,MSGCLASS=X
//*
//         EXPORT SYMLIST=MEM
//         SET MEM=ASMP1
//*
//GENER    EXEC PGM=IEBGENER
//SYSUT1   DD DATA,DLM=$$
//SYMBJOB2 JOB ACCNT#,MSGLEVEL=(1,1),NOTIFY=&SYSUID,MSGCLASS=X
//COPYMEM  EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DISP=SHR,DSN=PELEG.SRC
//SYSUT2   DD DISP=SHR,DSN=PELEG.SRC2
//SYSIN    DD *,SYMBOLS=JCLONLY
 COPY INDD=SYSUT1,OUTDD=SYSUT2
 SELECT MEMBER=&MEM
/*
//
$$
//SYSUT2   DD SYSOUT=(A,INTRDR),SYMLIST=(MEM)
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//
```

Starting with z/OS 2.1, the internal reader signals the result of the job submission by using special JES symbols. If the job was successfully submitted, the following special JES symbols are set:

- SYS_CORR_LASTJOB

  This value is set for the job *correlator* of the job that was just submitted, including the user portion, if it is provided.

- SYS_LASTJOBID

  This value is set for the job *identifier* of the job that was just submitted.

If job submission fails, these SYS symbols are set to empty values.

The job correlator of a job can be displayed by using the **$DJ,JOBCORR** command, as shown in Example 15-5.

Example 15-5 $DJ,JOBCORR command

```
$DJ4000,JOBCORR
$HASP890 JOB(PELEGEXP)  JOBCORR=J0004000WTSCPLX7CB389CDF.......:
```

In addition, the following JES2 commands are enhanced to accept the job correlator as a limiting keyword:

```
$A J
$C J
$C O J
$D J
$D O J
$E J
$H J
$L J
$O J
$P J
$P O J
$S J
$T J
$T O J
```

15.2.2 Migration and coexistence considerations

The job correlator exists for every job on the system. However, only a managed application system (MAS) member at Version 2.1 or later can display it or access it programmatically.

The user portion of the job correlator can be set only if job is submitted to the MAS member in Version 2.1.

15.3 JES Symbol Service

The JES Symbol Service provides a single view of JCL and JES symbols. The JES Symbol Service consists of the IAZSYMBL invocation macro and the IAZSYMDF data definition macro. If the **EXTRACT** function does not find a JES symbol with a particular name, it searches for a JCL symbol with the same name.

The JES Symbol Service manages JES symbols, which can be used in the following ways:

- JES symbols can be used to pass data between applications that are running in the same job step. For example, one application could use the CREATE function to create a JES symbol with a particular name and value. A second application could then use the EXTRACT function to retrieve this value.
- JES symbols can be used to create JCL symbols for the submitted jobs. For example, an application can use the CREATE function to create a JES symbol with a particular name and value. This JES symbol and value can then be exported, using the INTRDR, as a JCL symbol for a submitted job.
- Several special purpose JES symbols can be used to pass information between applications and JES.
- JES symbols can be used internally by an application. As a method of communication, a JES symbol that is created by one program can be used by another program within the same job step.
- JES symbols can be passed on to a submitted job by including the symbol names in the `SYMLIST=` keyword of an internal reader (INTRDR) allocation. JES symbols that are passed in this way must be valid JCL symbol names, which consist of the following elements:
  - 1 - 8 characters from the subset A - Z (capitals only)
  - 0 - 9 (numerics)
  - @ (at character)
  - # (number sign character)
  - $ (dollar sign character)
- JES symbols can be used for in-stream symbol substitution in the same job step. JES in-stream substitution occurs when in-stream data set is read. During this substitution, the following symbols can be used:
  - JCL symbols that are exported by the converter
  - JES symbols that are dynamically created by the JES Symbol Service
  - System symbols

  The specific symbols to use for substitution are defined by using the `SYMBOLS` keyword parameter in the DD statement that defines the in-stream data set.
- JES symbols can be used to communicate information between applications and JES2.

For a detailed description of the JES symbol service macros, see *z/OS JES Application Programming* (see “Related publications” on page 537 for a link).

15.4 Requesting job notification through ENF 78

In z/OS 2.1, the internal reader facility can be used to request notification of a job that is being submitted. Applications can use job notification to track submitted jobs.

15.4.1 Invocation and use

Job notification is requested by defining the SYS_JOB_NOTIFY JES symbol before submitting a job through using the internal reader. When the job is no longer eligible for execution, JES sends a job completion notification by ENF 78. The value of SYS_JOB_NOTIFY can be up to 4096 bytes long. ENF 78 includes job identification information and the value of the SYS_JOB_NOTIFY symbol.

15.4.2 Interactions and dependencies

IBM z/OS Management Facility (z/OSMF) 2.1 uses the job notification function to track a job's progress through the system.

15.5 8-character job classes

Before z/OS 2.1, JES2 provided 38 predefined job classes:

- Started task control (STC)
- Time-sharing logons (TSU) job classes
- 36 one-character (A - Z and 0 - 9) job classes

Starting with Version 2.1, an unlimited number of user-defined job classes of 2 - 8 characters each are supported. There is no architectural limit on the number of 2- through 8-character job classes, but a practical limit can be imposed by performance, hardware, and other variables that exist for a given MAS.

15.5.1 Invocation and use

To add and delete 8-character job classes, use the **$ADD JOBCLASS** and **$DEL JOBCLASS** commands. The **$ADD JOBCLASS** command defines a job class or group to be used by JES2. The **$DEL JOBCLASS** command deletes a job class that was added with the **$ADD JOBCLASS** command. The **$DEL JOBCLASS** command cannot be used to delete the standard job classes (A - Z, 0 - 9, STC, and TSU).

An 8-character job class can be either active or inactive. jobs cannot be queued to an inactive class. Existing jobs cannot be altered to be in an inactive class. Existing jobs remain in the class and are eligible to run. To change a job class status to active or inactive, use the `ACTIVE=` keyword on the **$ADD JOBCLASS** and **$T JOBCLASS** commands. This parameter does not apply to the STC and TSU job classes.

Example 15-6 shows how to add a class named `CLASS1`, mark it as active, and associate it with group `XYZ`. Because active job classes cannot be deleted, the class is changed to the inactive state before deletion. Then, the CLASS1 class is deleted.

Example 15-6 $ADD JOBCLASS and $DEL JOBCLASS

```
$ADD JOBCLASS(CLASS1),ACTIVE=YES,GROUP=XYZ
$HASP837 JOBCLASS(CLASS1)
$HASP837 JOBCLASS(CLASS1)    ACTIVE=YES,GROUP=XYZ,MODE=JES,
$HASP837                     QAFF=(ANY),QHELD=NO,SCHENV=,
$HASP837                     XEQCOUNT=(MAXIMUM=*,CURRENT=0),
$HASP837                     XEQMEMBER(SC74)=(MAXIMUM=*,
$HASP837                     CURRENT=0)
$DEL JOBCLASS(CLASS1)
$HASP003 RC=(55),DEL
$HASP003 RC=(55),DEL JOBCLASS(CLASS1)  - REQUEST INVALID DUE TO
$HASP003             ACTIVE STATUS
$T JOBCLASS(CLASS1),ACTIVE=NO
$HASP837 JOBCLASS(CLASS1)
$HASP837 JOBCLASS(CLASS1)    ACTIVE=NO,GROUP=XYZ,MODE=JES,
$HASP837                     QAFF=(ANY),QHELD=NO,SCHENV=,
$HASP837                     XEQCOUNT=(MAXIMUM=*,CURRENT=0),
```

```
$HASP837                    XEQMEMBER(SC75)=(MAXIMUM=*,
$HASP837                    CURRENT=0),
$HASP837                    XEQMEMBER(SC74)=(MAXIMUM=*,
$HASP837                    CURRENT=0)
$DEL JOBCLASS(CLASS1)
$HASP837 JOBCLASS(CLASS1)     - ELEMENT DELETED
```

The **$T JOBCLASS** command is updated with the ACTIVE= and GROUP= keywords to control whether a group is active or inactive and to control class and group associations. Example 15-7 shows how to use the **$T JOBCLASS** command to associate classes CLASS1 and CLASS2 with the group XYZ and mark the classes as active.

Example 15-7 $T JOBCLASS command

```
$T JOBCLASS(CLASS1,CLASS2),GROUP=XYZ,ACTIVE=YES
```

A complete description of the **$ADD JOBCLASS**, **$DEL JOBCLASS** and **$T JOBCLASS** commands is available in *z/OS V2R1.0 JES2 Commands*, SA32-0990 (see "Related publications" on page 537 for a link).

Note: After an 8-character job class is added, JES2 no longer maintains the 36 class queue heads in the checkpoint section of the $HCT ($JQHEADI).

Assigning 8-character job classes to initiators

The syntax of the **$T INIT** command is updated to support 8-character job classes. When using 8-character job class names, you must use the form of the CLASS= keyword, which lists the job classes that are associated with the initiator in parentheses. See Example 15-8.

Example 15-8 $T INIT command with 8-character job class names

```
$T INIT(7),CLASS=A
$HASP892 INIT(7)   STATUS=INACTIVE,CLASS=A,NAME=7,ASID=0053
$T INIT(7),CLASS=(A,CLASS1,CLASS2)
$HASP892 INIT(7)
$HASP892 INIT(7)   STATUS=INACTIVE,CLASS=(A,CLASS1,CLASS2),
$HASP892           NAME=7,ASID=0053
```

In the same manner, the **$T OFF(n).JR** and **$T OFF(n).JT** commands, which control offload job receiver and transmitter characteristics, are updated to support 8-character class names.

Job class groups

To avoid a large number of 2- through 8-character job class names being associated with a single initiator or a device, you can create job class groups to manage these associations. In a manner similar to placing NJE nodes in subnets, job classes can be defined to a job class group by following these guidelines:

- A job class can be in no job class group or in only one job class group.
- A job class group is created when the first job class is added to the group.
- A job class group is deleted when the last job class is removed from the group.
- Deleting a job class also deletes the job class from its job class group.
- There is no limit to the number of job class groups that can be defined and no limit to the number of job classes in a group.

- Job class group names must be unique, from 2 - 8 alphanumeric characters, and cannot match any existing job class name.

The examples in Example 15-9, which follows, and Example 15-10 show how to use the **$ADD JOBCLASS** and **$T JOBCLASS** commands to associate job class with a group.

Example 15-9 shows how to associate initiator 7 with CLASS1, CLASS2, CLASS3, CLASS4, and CLASS5 by using the group name in the **$T INIT** command. Jobs that are submitted to all of these classes can run in initiator 7.

Example 15-9 Associate job class group with an initiator

```
$ADD JOBCLASS(CLASS1,CLASS2,CLASS3,CLASS4,CLASS5),GROUP=XYZ,ACTIVE=YES
$HASP837 JOBCLASS(CLASS1)
$HASP837 JOBCLASS(CLASS1)    ACTIVE=YES,GROUP=XYZ,MODE=JES,
$HASP837                     QAFF=(ANY),QHELD=NO,SCHENV=,
$HASP837                     XEQCOUNT=(MAXIMUM=*,CURRENT=0),
$HASP837                     XEQMEMBER(SC74)=(MAXIMUM=*,
$HASP837                     CURRENT=0)
$HASP837 JOBCLASS(CLASS2)
$HASP837 JOBCLASS(CLASS2)    ACTIVE=YES,GROUP=XYZ,MODE=JES,
$HASP837                     QAFF=(ANY),QHELD=NO,SCHENV=,
$HASP837                     XEQCOUNT=(MAXIMUM=*,CURRENT=0),
$HASP837                     XEQMEMBER(SC74)=(MAXIMUM=*,
$HASP837                     CURRENT=0)
... messages deleted ...
$HASP837 JOBCLASS(CLASS5)
$HASP837 JOBCLASS(CLASS5)    ACTIVE=YES,GROUP=XYZ,MODE=JES,
$HASP837                     QAFF=(ANY),QHELD=NO,SCHENV=,
$HASP837                     XEQCOUNT=(MAXIMUM=*,CURRENT=0),
$HASP837                     XEQMEMBER(SC74)=(MAXIMUM=*,
$HASP837                     CURRENT=0)
$T INIT(7),CLASS=(XYZ)
$HASP892 INIT(7)   STATUS=INACTIVE,CLASS=(XYZ),NAME=7,ASID=0053
```

Defining 8-character job classes in JES2PARM

The JOBCLASS statement is updated to support 8-character job classes. Example 15-10 shows an example.

Example 15-10 Define 8-character job class in JES2PARM

```
JOBCLASS(CLASS1)     ACTIVE=YES,       GROUP=XYZ,       MODE=JES,
                     AUTH=ALL,         BLP=NO,          LOG=NO,
                     REGION=0M,        CONDPURG=NO,     OUTPUT=YES,
                     SWA=ABOVE,        MSGCLASS=S,
                     TIME=(1440,0)
```

Submitting jobs to run in 8-character job class

The CLASS= keyword on the JCL job card is updated to support 8-character job class names. Example 15-11 on page 211 shows an example of the CLASS= keyword.

Example 15-11 Using CLASS= keyword with 8-character job class name

```
//PELEG14  JOB ACCNT#,MSGLEVEL=(1,1),NOTIFY=&SYSUID,CLASS=CLASS1
//STEP1    EXEC PGM=IEFBR14
//
```

Figure 15-3 shows the JES2 job log messages, which indicate that the job runs in CLASS1.

```
J E S 2  J O B  L O G  --  S Y S T E M  S C 7 4  --  N O D E  W T S C P L X 7

--- WEDNESDAY, 17 APR 2013 ----
IRR010I  USERID PELEG    IS ASSIGNED TO THIS JOB.
ICH70001I PELEG    LAST ACCESS AT 09:40:18 ON WEDNESDAY, APRIL 17, 2013
$HASP373 PELEG14  STARTED - INIT 7    - CLASS CLASS1   - SYS SC74
-                                        ---------TIMINGS (MINS.)---------
-JOBNAME  STEPNAME PROCSTEP    RC   EXCP    CPU    SRB   VECT   VAFF  CLOCK
-PELEG14  STEP1                00      8    .00    .00    .00    .00     .0
-PELEG14  ENDED.  NAME-                     TOTAL CPU TIME=   .00  TOTAL ELAP
$HASP395 PELEG14  ENDED
```

Figure 15-3 Job log messages when running in an 8-character job class

15.5.2 Interactions and dependencies

The System Display and Search Facility (SDSF) is updated to support 8-character job classes.

15.5.3 Migration and coexistence considerations

The 8-character job classes can be used only if all MAS members are at z/OS 2.1 or later.

The **$ADD JOBCLASS** command requires that all members of the job class are running z/OS 2.1 or later. The **$DEL JOBCLASS** command can be used only for classes that were created by the **$ADD JOBCLASS** command.

15.6 SAF controls on input class use

Resource Access Control Facility (RACF) profiles are added to verify a job's submitter or owner access to a job class. This applies to the traditional 36 job classes and 8-character job classes. It does not apply to the special STC and TSU job classes. A authorization is checked during input processing. This ensures that the user ID that is associated with the job has access to the job class associated with the job. A authorization is also checked if a user changes the execution class of a job when the job is held before execution.

15.6.1 Invocation and use

JES2 normally allows any job class to be used. An installation can control job class use by granting access based on either the submitter's profile or on the owner's profile, or both. The control is based on the presence or absence of two FACILITY class profiles and the following conditions:

- When the JES.JOBCLASS.OWNER profile is defined in the FACILITY class, job class profiles for owners are enforced.
- When the JES.JOBCLASS.SUBMITTER profile is defined in the FACILITY class, job class profiles for submitters are enforced.
- When both profiles are defined in the FACILITY class, a job class must pass both checks before it is considered valid.
- When either of these profiles is defined, JES2 checks READ access to the following profile in the JESJOBS class:

  JOBCLASS.nodename.jobclass.jobname

  Where *nodename* is the local NJE node name, *jobclass* is the job class to be associated with the job, and *jobname* is the job name of the job that is being submitted.

15.7 JES2-related enhancements to JCL

The JES2 support for the JOB, OUTPUT, and JCLLIB JCL statements is enhanced in z/OS 2.1 to provide simpler JCL processing and to reduce the dependency on JES2-specific JECL statements.

15.7.1 Invocation and use

Rather than using the /*JOBPARM SYSTEM= or SYSAFF= statements, it is now possible to specify the SYSTEM= and SYSAFF= keywords on the JOB card in these ways:

- Use the SYSTEM= keyword to indicate the systems that are eligible to process the job.
- Use the SYSAFF= keyword to indicate the JES2 members that are eligible to process the job.

The SYSTEM= and SYSAFF= keywords are mutually exclusive on the JOB card. Specifying both results in a JCL error. If SYSAFF= is specified on both the JOB card and in the /*JOBPARM statement, the keyword in the /*JOBPARM statement is ignored and a warning message is issued.

Example 15-12 shows an example of specifying the SYSAFF= keyword on the JOB card. In this example, a system affinity to JES2 member SC74 is requested.

Example 15-12 Using the SYSAFF= keyword on the JOB card

```
//PELEG14  JOB ACCNT#,MSGLEVEL=(1,1),NOTIFY=&SYSUID,CLASS=CLASS5,
// SYSAFF=SC74
//STEP1    EXEC PGM=IEFBR14
//
```

Example 15-13 on page 213 shows an example of specifying the SYSTEM= keyword on the JOB card. In the example, system SC75 is specified as an eligible system for the job.

Example 15-13 Using SYSTEM= keyword on the JOB card

```
//PELEG14  JOB ACCNT#,MSGLEVEL=(1,1),NOTIFY=&SYSUID,CLASS=CLASS5,
// SYSTEM=SC75
//STEP1    EXEC PGM=IEFBR14
//
```

OUTPUT statement enhancements

A MERGE= keyword is added on the OUTPUT JCL statement. The MERGE= keyword indicates whether the parameters that are specified on the OUTPUT JCL statement are merged with the output statements for the job. The default is MERGE=NO.

The JCL example in Example 15-14 shows how to use the OUTPUT JCL statements with MERGE=YES keyword.

Example 15-14 Using MERGE= keyword on the OUTPUT JCL statement

```
//TESTGENR JOB ACCNT#,MSGLEVEL=(1,1),NOTIFY=&SYSUID,MSGCLASS=X
//*
//BUILDING OUTPUT BUILDING='008',MERGE=YES
//ROOMDEPT OUTPUT ROOM='A1',DEPT='ITSO',DEFAULT=YES
//*
//STEP1    EXEC PGM=IEBGENER
//SYSUT1   DD DISP=SHR,DSN=PELEG.JOBS(IEBGENER)
//SYSUT2   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//*
//STEP2    EXEC PGM=IEBGENER
//ROOMDPT2 OUTPUT ROOM='A2',DEPT='ITSO',DEFAULT=YES
//SYSUT1   DD DISP=SHR,DSN=PELEG.JOBS(IEBGENER)
//SYSUT2   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//*
//STEP3    EXEC PGM=IEBGENER
//BUILDING OUTPUT BUILDING='009',MERGE=YES
//ROOM3    OUTPUT ROOM='A3'
//SYSUT1   DD DISP=SHR,DSN=PELEG.JOBS(IEBGENER)
//SYSUT2   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*,OUTPUT=*.ROOM3
//SYSIN    DD DUMMY
//
```

Table 15-2 on page 214 summarizes the definitions from the OUTPUT statements of the TESTGENR job, which is shown in Example 15-14.

DD SYSUT2 and DD SYSPRINT of STEP1 are defined with room A1 and department ITSO because of the job level ROOMDEPT OUTPUT statement, which is the default output statement for the job. DD SYSUT2 and DD SYSPRINT of STEP1 are defined with building 008 because the job level BUILDING OUTPUT statement is merged with the job level ROOMDEPT OUTPUT statement.

DD SYSUT2 and DD SYSPRINT of STEP2 are defined with room A2 and department ITSO because the ROOMDPT2 OUTPUT statement under STEP2 is the default OUTPUT statement for STEP2 and overrides ROOMDEPT.

DD SYSUT2 of STEP3 is defined with room A1 and department ITSO because of the job level ROOMDEPT OUTPUT statement, which is the default OUTPUT statement for the job. It is defined with building 009 because the BUILDING OUTPUT statement under STEP3 is merged with the ROOMDEPT OUTPUT statement. The BUILDING OUTPUT statement on STEP3 overrides the job level BUILDING OUTPUT statement.

DD SYSPRINT of STEP3 is defined with room A3 and a null department because it explicitly requests the output definitions of the ROOM3 OUTPUT statement under STEP3. When a DD explicitly references an OUTPUT statement, that OUTPUT statement overrides the default output statement. DD SYSPRINT of STEP3 is defined with building 009 because the BUILDING OUTPUT statement under STEP3 is merged with the ROOM3 output statement.

Table 15-2 Output definition results from OUTPUT statements in Example 15-14 on page 213

DD Name	Step name	Building	Department	Room
SYSUT2	STEP1	008	ITSO	A1
SYSPRINT	STEP1	008	ITSO	A1
SYSUT2	STEP2	008	ITSO	A2
SYSPRINT	STEP2	008	ITSO	A2
SYSUT2	STEP3	009	ITSO	A1
SYSPRINT	STEP3	009		A3

JCLLIB statement enhancements

A `PROCLIB=` keyword is added to the JCLLIB statement that requests a JES2 procedure library by its DDname, as defined in the JES2 procedure that is used to initialize JES2. Typically, JES2 procedure library DDnames in the JES2 procedure are in the `PROC`*nn* format, where *nn* is either 00 or 1 or 2 decimal numbers, 1 - 99. However, you can use any valid DDname if the name matches the DDname in the JES2 procedure or is specified in PROCLIB definitions. The system retrieves called cataloged procedures from the requested JES2 procedure library. Figure 15-4 shows an example of the output.

```
//ASMJOB1  JOB ACCNT#,MSGCLASS=X,MSGLEVEL=(1,1),NOTIFY=&SYSUID
//*
//         JCLLIB PROCLIB=PROC01
//*
//ASMACL   EXEC PROC=ASMACL,
//             PARM.C='NOLIBMAC',
//             PARM.L='MAP,LET,LIST,RMODE=SPLIT,REUS=NONE'
//C.SYSIN   DD DISP=SHR,DSN=PELEG.SRC(ASMP3)
//L.SYSLMOD DD DISP=SHR,DSN=PELEG.PDSE.LOAD(ASMP3)
//
```

Figure 15-4 Using JCLLIB PROCLIB= keyword

In addition, the **$D PROCLIB** command is enhanced to display the static PROCLIB concatenations in the JES2 procedure, as shown in Figure 15-5 on page 215.

```
$D PROCLIB
$HASP319 PROCLIB(PROC00)
$HASP319 PROCLIB(PROC00)    STATIC PROCLIB,
$HASP319                    DD(1)=(DSNAME=SYS1.PROCLIB,
$HASP319                    VOLSER=BH5CAT,UNIT=SYSALLDA),
$HASP319                    DD(2)=(DSNAME=SYS1.IBM.PROCLIB,
$HASP319                    VOLSER=Z21RA1,UNIT=SYSALLDA)
```

Figure 15-5 $D PROCLIB command

15.8 Improved processing of input phase errors

In z/OS 2.1, input phase JCL errors are handled by JES2 in the same way as any other JCL error. Most input errors do not prevent the conversion phase, and input phase errors are reported in the same way that the JCL errors that are detected. Warning messages might also be issued for input phase errors. In addition, JES2 now assigns JCL statement numbers to JECL statements.

15.8.1 Invocation and use

Figure 15-6 shows an example of the input phase error message *before* Version 2.1.

```
//IBMUSERA JOB (,2D07),MSGLEVEL=(1,1),CLASS=ABC,SYSAFF=(BAD)          JOB00767
************ ILLEGAL JOB CARD - VALUE OF CLASS= EXCEEDS 1 CHARACTER ***********
/*JOBPARM PROC=PROC99
******* NON-VALID JOBPARM  STMT - UNEXPECTED KEYWORD DETECTED - PROC     ******
//*
//STEP1 EXEC PGM=IEBDG,REGON=0M
//SYSPRINT DD SYSOUT=*
//DATASET1 DD SYSOUT=*
//SYSIN DD DATA,DLM=$$$$
******** NON-VALID DD       STMT - VALUE FOR DLM      KEYWORD NOT VALID *******
$HASP106 JOB DELETED BY JES2 OR CANCELLED BY OPERATOR BEFORE EXECUTION
------ JES2 JOB STATISTICS ------
           17 CARDS READ
            7 SYSOUT PRINT RECORDS
            0 SYSOUT PUNCH RECORDS
            0 SYSOUT SPOOL KBYTES
         0.00 MINUTES EXECUTION TIME
```

Figure 15-6 Input phase error messages before V2.1

Figure 15-7 shows an example of the input phase errors in V2.1.

```
12.43.45 JOB00042 IEFC452I IBMUSERA - JOB NOT RUN - JCL ERROR 533
------ JES2 JOB STATISTICS ------
           17 CARDS READ
           24 SYSOUT PRINT RECORDS
            0 SYSOUT PUNCH RECORDS
            1 SYSOUT SPOOL KBYTES
         0.00 MINUTES EXECUTION TIME
        1 //IBMUSERA JOB (,2D07),MSGLEVEL=(1,1),CLASS=ABC,SYSAFF=(BAD)
        2 /*JOBPARM PROC=PROC99
          //*
        3 //STEP1 EXEC PGM=IEBDG,REGON=0M
        4 //SYSPRINT DD SYSOUT=*
        5 //DATASET1 DD SYSOUT=*
        6 //SYSIN DD DATA,DLM=$$$$
 STMT NO. MESSAGE
        1 HASP110 value of CLASS= parameter is not valid
        1 HASP112 value of SYSAFF= parameter is not valid
        2 HASP107 UNEXPECTED KEYWORD DETECTED - PROC
        3 IEFC630I UNIDENTIFIED KEYWORD REGON
        6 HASP107 value for DLM keyword not valid
```

Figure 15-7 Input phase error messages in 2.1

15.9 Interpretation with conversion

The conversion phase of a submitted job runs in a subtask under the JES2 address space and performs the first pass of JCL processing. The conversion phase deals with `PROC` and `INCLUDE` statements and parses JCL into text units.

When a job is selected to run under an initiator, the interpretation phase runs under the initiator and fully validates the JCL, including all parameter values. Some JCL errors are not detected until the interpretation phase, after the initiator selects the job.

Starting with 2.1, JES2 provides a option to run the interpreter immediately after conversion (except **`LOCATE`** processing for data sets), during the conversion phase and before the job is selected for execution. When they are set to `On`, both the converter and interpreter run in a JES2 address space named *jesxCInn*, where *jesx* is the JES2 subsystem name and *nn* is the CI instance number.

15.9.1 Invocation and use

To enable interpretation after conversion, use the `INTERPRET=` keyword in one of these ways in the `JOBDEF JES2PARM` statement:

- When `INTERPRET=INIT` is set or defaulted to, JES2 runs interpretation under the initiator, as in previous versions.
- When `INTERPRET=JES` is set, the interpreter runs after the converter under the JES2CI address space.

CISUB_PER_AS= is a keyword for the JOBDEF statement that controls how many CI subtasks run in the JES2CI address space. The default is CISUB_PER_AS=5. Example 15-15 shows how to code the JOBDEF keywords in the JES2PARM member.

Example 15-15 Using INTERPRET= and CISUB_PER_AS= keywords

```
JOBDEF JOBNUM=3000,INTERPRET=JES,CISUB_PER_AS=10
```

The **$T JOBDEF** command can also be used to change when the interpreter runs. As shown in Example 15-16, the interpreter is changed to run after conversion under JES. When a job is submitted, JES2 starts the JES2CI01 address space to run the converter and interpreter. The JES2CI01 address space then remains up, waiting for requests.

Example 15-16 Using $T JOBDEF,INTERPRET=

```
$T JOBDEF,INTERPRET=JES
$HASP835 JOBDEF 957
$HASP835 JOBDEF  ACCTFLD=OPTIONAL,BAD_JOBNAME_CHAR=?,
$HASP835         CNVT_ENQ=FAIL,INTERPRET=JES,CISUB_PER_AS=5,
$HASP835         CNVT_SCHENV=IGNORE,JCLERR=YES,JNUMBASE=4465,
$HASP835         JNUMFREE=9410,JNUMWARN=80,JOBFREE=2411,
$HASP835         JOBNUM=3000,JOBWARN=80,PRTYHIGH=10,
$HASP835         PRTYJECL=YES,PRTYJOB=NO,PRTYLOW=5,PRTYRATE=0,
$HASP835         RANGE=(1,9999),RASSIGN=YES,JOBRBLDQ=NONE,
$HASP835         DUPL_JOB=DELAY
$HASP100 PELEBR14 ON INTRDR                          FROM TSU04450
PELEG
IRR010I  USERID PELEG    IS ASSIGNED TO THIS JOB.
IRR812I PROFILE ** (G) IN THE STARTED CLASS WAS USED 960
        TO START JES2CI01 WITH JOBNAME JES2CI01.
IEF196I         1 //IEESYSAS JOB TIME=NOLIMIT,REGION=0M,
IEF196I           // MSGLEVEL=1
IEF196I         2 //JES2CI01 EXEC IEESYSAS,PROG=HASJES2#
... messages deleted ...
```

A internal data set named $SWABLKS is created on the spool by JES2 to hold the control blocks that are created during interpretation. When a job is selected to run, these control blocks are copied into the initiator address space.

Exit programming changes

The following conditions might require code changes:

- CNVT code to read IOT CBs moved to subtask
  - Calls CBIO exit 8 rather than exit 7
- Exit 6 always considered to be in USER environment
  - R11 on entry is HCCT, not HCT
  - Ensure that $ENVIRON set correctly
- internal data set to store interpreter output
  - $SWABLKS DD name
  - Always created by z/OS 2.1 JES2 during input phase

These are the implications of setting INTERPRET=JES:

- The converter interpreter runs outside of the JES2 address space
  - Exits 6 and 8 cannot access JES2 private storage
- exit 59 – post interpreter exit and processing OUTPUT cards

15.9.2 Migration and coexistence considerations

All MAS members must be at z/OS 2.1, and **$ACTIVATE at LEVEL=Z11** must have been run to use this function. Otherwise, the option can be set, but it has no effect.

15.10 Controlling where a job converts

In z/OS 2.1, JES2 provides a option to select the job conversion system, based on the SCHENV= keyword in the JOB card.

15.10.1 Invocation and use

A keyword, CNVT_SCHENV=, is added to the JOBDEF statement with these options:

- When CNVT_SCHENV=IGNORE is coded or defaulted to, job conversion may be performed by any member.
- When CNVT_SCHENV=HONOR is coded, conversion is performed only on a member where the scheduling environment is available. If the scheduling environment is not available, the job waits for conversion.

15.10.2 Migration and coexistence considerations

MAS members that are running a version before 2.1 can select a job for conversion even if CVNT_SCHENV=HONOR is specified.

15.11 Displaying JES2 initialization information

A JES2 command (**$D INITINFO**) is provided to display JES2 initialization statistics, including these statistics:

- The command that is used to start JES2
- The JES2PARM initialization decks that are used, including number of cards read
- The JES2 STEPLIB concatenation

15.11.1 Invocation and use

To display the JES2 initialization statistics, issue the **$D INITINFO** command as shown in Example 15-17 on page 219.

Example 15-17 $D INITINFO command

```
$D INITINFO
$HASP825 INITINFO 477
$HASP825 INITINFO --- Command used to start JES2
$HASP825          S JES2,N=ZOS21,M=SPOOLZ21,PARM=(WARM,NOREQ)
$HASP825          --- HASPPARM data sets read
$HASP825          DSN=SYS1.PARMLIB(SPOOLZ21),VOLSER=J2SHR2,
$HASP825          CARDS=458,
$HASP825          DSN=SYS1.PARMLIB(DYEXIT21),CARDS=122,
$HASP825          DSN=CONSOLE,CARDS=1,
$HASP825          DSN=SYS1.PARMLIB(NULL),VOLSER=J2SHR2,CARDS=1
$HASP825          --- STEPLIB Concatenation
$HASP825          DSN=ZOS21.LINKLIB,VOLSER=J2COM1,
$HASP825          DSN=NULL.JES2000.LINKLPA,VOLSER=J2SPA1,
$HASP825          DSN=SYS1.SRVLIB.JES2000.LINKLPA,
$HASP825          VOLSER=J2SPA1,
$HASP825          DSN=SYS2.LINKLIB,VOLSER=ZDR21B,
$HASP825          DSN=SYS1.MIGLIB,VOLSER=ZDR21B
```

15.12 Performance improvements for SYSOUT processing

Various JES2 devices select SYSOUT output groups from JES2 for processing. For example, printers select SYSOUT for printing and SAPI applications select SYSOUT for printing to high-function printers or archiving to specialized storage. The process of selecting SYSOUT output groups from JES2 is called *SYSOUT work selection*. As the number of SYSOUT output groups that are stored in JES2 and the number of JES2 devices that process this work grow, the overhead of SYSOUT work selection can become unacceptable.

The SYSOUT work selection algorithms in JES2 are redesigned in 2.1 to provide better performance for applications that process SYSOUT output groups.

15.12.1 Invocation and use

There are two ways for an application to process SYSOUT output groups:

- Active work selection, where the application requests work from JES2 (**GET**).
- Passive work selection, where the application identifies its interest and JES2 notifies the application when work with required characteristics becomes available (**POST**).

Two keywords are added to the **$T OUTDEF** command to enable the performance optimizations:

- `$T OUTDEF,SAPI_OPT=YES`

  This keyword enables passive work selection optimization. This optimization has a member scope and is active only until this member is restarted. The default is NO.

- `$T OUTDEF,WS_OPT=YES`

  This keyword enables active work selection optimization. This type of optimization has MAS scope and persists until it is explicitly disabled. The default is NO.

The two optimization types are independent and can be enabled in any combination. No changes to SAPI applications are required.

Note: In environments where the majority of SYSOUT jobs are never processed, it is possible that the overhead demands of index maintenance is not offset by the performance improvements of work selection. In this case, it might be preferable not to implement the algorithm.

15.12.2 Installation

Active SYSOUT work selection optimization builds data structures in the JES2 checkpoint, JOE index, and uses more output group elements (JOEs). The number of additional JOEs required depends on the statistical distribution of attributes of existing output groups (JOEs). The number of additional JOEs is expected to be within a small percent range of the number of output groups on the system.

The **$D OUTDEF,JOEUSE** command is enhanced to display the number of JOEs used by the JOE index, as the output in Figure 15-8 shows.

```
$HASP836 OUTDEF 296
$HASP836 OUTDEF  CURRENT JOE UTILIZATION
$HASP836         TYPE         COUNT
$HASP836         -------- ---------
$HASP836         WORK             3
$HASP836         CHAR             2
$HASP836         INDEX            2
$HASP836         FREE           193
```

Figure 15-8 $D OUTDEF,JOEUSE command output

15.12.3 Migration and coexistence considerations

Active work selection optimization requires that all members of the JES2 MAS be at 2.1 level, and JES2 checkpoint must be at the z11 level (z/OS 1.11). When a down-level JES2 member joins the MAS (with a requisite compatibility APAR applied), active work optimization is suspended. This has no functional impact on any running application or JES2 feature.

Passive work selection optimization has a single member scope and is not affected by this consideration.

15.13 SSI 85 - Unauthorized services for spool access

The Job Modify Service SSI (SSI 85) is created to allow for modification of JES job data without supplying a JES-specific command. The Job Modify Service can be used to perform the following functions:

- Modify job characteristics (job class, priority, SYSAFF, service class, scheduling environment, or offload status)
- Hold a job
- Release a job
- Purge a job
- Cancel a job, with options to purge or dump

- Restart a job with cancel or step, and hold options
- SPIN a job
- Change a job's execution node
- Start a job

Additional security checks are done in the requester's address space for access to alter the job that is being processed. The JESJOBS RACF class is used and requires additional resource names for the major actions that can be performed. The following JESJOBS profile names are used:

- CANCEL.nodename.userid.jobname
- MODIFY.nodename.userid.jobname
- HOLD.nodename.userid.jobname
- RELEASE.nodename.userid.jobname
- PURGE.nodename.userid.jobname
- RESTART.nodename.userid.jobname
- SPIN.nodename.userid.jobname
- REROUTE.nodename.userid.jobname
- START.nodename.userid.jobname

SSI 85 supports the same set of filtering capabilities as the Extended Status SSI 80 (IAZSSST). However, only one function can be requested per SSI call. The requested function is attempted on all jobs that are selected by the filter.

15.13.1 Invocation and use

The SSI caller can request either asynchronous or synchronous operation.

Asynchronous requests occur when the job becomes available. No direct results of the action are provided. The request is queued to the JES2 Main task for all requested jobs, and control is returned to the SSI caller.

For synchronous requests, JES2 provides feedback on the results of performing the requested action. The request requires the job lock to attempt the action, which is attempted immediately. The request is queued to the JES2 main task for processing, and JES2 waits for the request to be completed on all requested jobs before it returns control to the SSI caller. The JES2 Main task routes requests for running jobs to the member where the job is running. For a synchronous request, the results are returned to the initiating member.

A job feedback block (SSJF) is returned for each job that is selected and processed by the Job Modify SSI service. The returned values are job name, job ID, original job ID, owner user ID, member name, system name, the job correlator, and a job processing status indicator. The indication differs by type of request:

- For an asynchronous request, the job processing status indicator indicates whether a problem was encountered while processing that job or the request was successfully queued to the JES2 Main task.
- For a synchronous request job, processing status indicates whether a problem was encountered while processing that job or indicates the results of performing the requested function on the job.

15.13.2 Interactions and dependencies

The IBM z/OS Management Facility (z/OSMF) 2.1 uses the SSI 85.

15.13.3 Migration and coexistence considerations

A z/OS 1.11, 1.12, or 1.13 MAS member needs APAR OA36155 to coexist in the MAS with z/OS 2.1.

15.14 Documentation and reference

For documentation of the described enhancements, see these publications:

- z/OS JES2, IBM Knowledge Center for z/OS 2.1.0

  http://ibm.co/1lOSakk
- z/OS JES2, IBM z/OS V2R1 Information Center:

  http://ibm.co/1qEZvck
  - *z/OS V2R1.0 JES Application Programming,* SA32-0987
  - *z/OS V2R1.0 JES2 Commands,* SA32-0990
  - *z/OS V2R1.0 JES2 Initialization and Tuning Guide,* SA32-0991
  - *z/OS V2R1.0 JES2 Initialization and Tuning Reference,* SA32-0992
- *z/OS MVS Programming: Authorized Assembler Services Reference EDT-IXG*

  http://ibm.co/1j9J2Me
- *z/OS V2R1.0 MVS JCL Reference,* SA23-1385

  http://ibm.co/1qEZvck
- *z/OS V2R1.0 MVS JCL User's Guide,* SA23-1386

  http://ibm.co/1qEZvck
- "z/OS V2R1 Removes Many JCL Symbol Limitations, *IBM Systems Magazine*," Jan 2014

  https://ibm.biz/BdFd7K

16

Job Entry Subsystem 3 (JES3)

This chapter describes the JES3 enhancements, functions, and features that were introduced with IBM z/OS Version 2 Release 1 (2.1).

- 16.1, “Dynamic spool deletion” on page 224
- 16.2, “Enhanced spool use data” on page 224
- 16.3, “In-stream data in JCL procedures” on page 226
- 16.4, “Multiple simultaneous logons in a Parallel Sysplex” on page 227
- 16.5, “8-character job classes on the JOB card” on page 228
- 16.6, “SAF control over job class” on page 228
- 16.7, “JES3-related enhancements to JCL” on page 229
- 16.8, “ENF 70 events” on page 230
- 16.9, “Documentation and reference” on page 231

© Copyright IBM Corp. 2014. All rights reserved.

16.1 Dynamic spool deletion

JES3 is enhanced in z/OS 2.1 to support dynamic spool deletion by using the ***MODIFY CONFIG** command or a hot start with refresh. This eliminates the IBM Parallel Sysplex-wide IPL that is required for a JES3 warm start when removing a spool extent in previous versions.

16.1.1 Invocation and use

Before you delete a spool data set, complete the following steps:

1. Place the data set in DRAIN status by running the ***F Q,DD=ddname,DRAIN** command. The command moves all single track table (STT) data to other spool volumes.
2. Stop all jobs with requirements for the spool extent in spool hold by running the ***F Q,DD=ddname,HOLD** command.
3. Offload all jobs by using **Dump Job**.
4. Process all Job0 output (issue ***I U,Q=WTR,J=0** or ***I U,Q=HOLD,J=0** to see whether any output exists).
5. Cancel any remaining jobs by running the ***F Q,DD=ddname,CANCEL** command.
6. Run the ***F J=jjj,CANCEL** command to stop any started tasks that could not be cancelled.
7. Delete the spool by running the ***F CONFIG** command. Alternatively, perform a hot start with refresh, using an updated initialization stream.

User exit IATUX73 is provided for processing unidentifiable (non-JES3) STT records.

16.1.2 Migration and coexistence considerations

All systems must be at the 2.1 JES3 level or later. APAR OA36848 is required on Version 1.13 JES3 local. If a lower-level system is found, existing WTOs IAT3426 and IAT2061are issued, followed by WTOR IAT2064. JES3 waits until the operator either resets the down-level system or cancels the request by responding to the WTOR. Systems at Version 1.12 or earlier must be IPLed again before they can connect.

16.2 Enhanced spool use data

In z/OS 2.1, JES3 implements a mechanism for tracking spool allocations in storage. This allows the ***INQUIRY,Q,SP=spart,U** command to use in-storage spool allocation data and return its answer sooner. Moreover, the **U** (use) parameter that is added to the ***INQUIRY,J=job,SD,U** command provides the capability to display the number of track groups that are used by the job in each spool data set. In addition, the **U** and **N=** parameters are added to the ***INQUIRY,Q,DD=ddn,U,N=nnn** command to provide the capability to display a list of jobs that use a spool data set or extent.

16.2.1 Invocation and use

The following examples show using the spool use commands. For complete documentation of the commands, see *z/OS V2R1.0 JES3 Commands* in the z/OS 2.1.0 IBM Knowledge Center:

https://ibm.biz/BdFd7R

Example 16-1 shows the output from the ***I Q SP=ALL U** command.

*Example 16-1 *I Q SP=spart U command*

```
*I Q SP=ALL U
IAT8583 DRAINED   TOTAL IN USE                 36 TRKGPS
IAT8583 DRAINED   TOTAL IN USE BY JES3          0 TRKGPS
IAT8583 DRAINED   TOTAL IN USE BY JOBS         36 TRKGPS
IAT8587 DRAINED   USERS FOUND=      0 JES3      14 JOBS;  10 DISPLAYED
IAT8527 DRAINED : JOB SYSLOG   (JOB32502)     19 TRKGPS,  <1%
... messages deleted ...
IAT8527 DRAINED : JOB BPXAS    (JOB32549)      1 TRKGPS,  <1%
IAT8527 DRAINED : JOB TCPIP    (JOB32553)      1 TRKGPS,  <1%
IAT8531 NO JOBS FOUND IN SPOOL PARTITION UNAVAIL
IAT8583 JES3PART  TOTAL IN USE              1,728 TRKGPS
IAT8583 JES3PART  TOTAL IN USE BY JES3         29 TRKGPS
IAT8583 JES3PART  TOTAL IN USE BY JOBS      1,699 TRKGPS
IAT8587 JES3PART  USERS FOUND=      4 JES3     598 JOBS;  10 DISPLAYED
IAT8527 JES3PART: JOB SYSLOG   (JOB32502)     76 TRKGPS,  <1%
IAT8527 JES3PART: JOB SYSLOG   (JOB32599)     44 TRKGPS,  <1%
... messages delted ...
IAT8527 JES3PART: JOB COMPRESS (JOB20400)      7 TRKGPS,  <1%
IAT8591 INQUIRY ON SPOOL SPACE USAGE COMPLETE
```

The ***I Q DD=ddn U** command (Example 16-2) displays a sorted list of the largest spool space users for the specified spool extents. A maximum of eight extent names can be specified when you use the **U** parameter. Track group totals are displayed for each extent specified.

*Example 16-2 *I Q DD=ALL U command*

```
*I Q DD=ALL U
IAT8751 NO JOBS FOUND IN SPOOL DATA SET JES3JCT
IAT8752 SPOOL1    TOTAL IN USE                596 TRKGPS
IAT8752 SPOOL1    TOTAL IN USE BY JES3         13 TRKGPS
IAT8752 SPOOL1    TOTAL IN USE BY JOBS        583 TRKGPS
IAT8753 SPOOL1    USERS FOUND=      4 JES3     358 JOBS;  10 DISPLAYED
IAT8754 SPOOL1  : JOB SYSLOG   (JOB32502)     35 TRKGPS,  <1%
... messages deleted ...
IAT8754 SPOOL1  : JOB RMF      (JOB32537)      3 TRKGPS,  <1%
IAT8752 SPOOL2    TOTAL IN USE                639 TRKGPS
IAT8752 SPOOL2    TOTAL IN USE BY JES3          7 TRKGPS
IAT8752 SPOOL2    TOTAL IN USE BY JOBS        632 TRKGPS
IAT8753 SPOOL2    USERS FOUND=      2 JES3     398 JOBS;  10 DISPLAYED
IAT8754 SPOOL2  : JOB SYSLOG   (JOB32599)     16 TRKGPS,  <1%
... messages deleted ...
IAT8754 SPOOL2  : JOB PHIL     (JOB26645)      3 TRKGPS,  <1%
IAT8752 SPOOL3    TOTAL IN USE                493 TRKGPS
IAT8752 SPOOL3    TOTAL IN USE BY JES3          9 TRKGPS
IAT8752 SPOOL3    TOTAL IN USE BY JOBS        484 TRKGPS
IAT8753 SPOOL3    USERS FOUND=      4 JES3     295 JOBS;  10 DISPLAYED
IAT8754 SPOOL3  : JOB SYSLOG   (JOB32502)     29 TRKGPS,  <1%
... messages deleted ...
IAT8754 SPOOL3  : JOB PRICHAR  (JOB27519)      3 TRKGPS,  <1%
IAT8752 SPOOL4    TOTAL IN USE                36 TRKGPS
IAT8752 SPOOL4    TOTAL IN USE BY JES3         0 TRKGPS
IAT8752 SPOOL4    TOTAL IN USE BY JOBS        36 TRKGPS
```

```
IAT8753 SPOOL4    USERS FOUND=      0 JES3      14 JOBS;  10 DISPLAYED
IAT8754 SPOOL4  : JOB SYSLOG   (JOB32502)     19 TRKGPS,  <1%
... messages deleted ...
IAT8754 SPOOL4  : JOB TCPIP    (JOB32553)      1 TRKGPS,  <1%
IAT8755 INQUIRY ON SPOOL DATA SET USAGE COMPLETE
```

The ***INQUIRY,J** command (Example 16-3) is updated to display information about a job's spooled data. The **U** parameter is added to display the distribution of a job's data across spool extents. The **U** parameter is valid for both the **SD** (spool data set) and **SH** (spool hold) versions of the command.

*Example 16-3 *I J=jjj SD U command*

```
*INQUIRY,J=32675,SD,U
IAT8674 JOB PELEG14  (JOB32675) P=01 CL=X        OUTSERV(PENDING WTR)
IAT8750 JOB PELEG14  (JOB32675) DD=SPOOL1     1 TRKGPS
IAT8750 JOB PELEG14  (JOB32675) DD=SPOOL3     1 TRKGPS
IAT8699 INQUIRY ON JOB STATUS COMPLETE,       1 JOB  DISPLAYED
```

Dump Jobs command extended to dump jobs by spool data set

To complete the support for showing which jobs have data on which spool data sets that is provided by the enhanced ***INQUIRY** commands, DJ processing now supports dumping jobs by spool data set criteria.

The ***START,DJ,DD=ddn** command dumps all jobs that have data on a specified spool extent. **DD=ddn** is a parameter that allows the caller to specify one or more spool extents. The **DD=ddn** parameter is also allowed for ***START,DJ** commands with the **RESET** parameter specified. The Dump Job DSP must be called for Out Mode by using the ***CALL,DJ,OUT=** command. If the Dump Job DSP is started for Input Mode by using the ***CALL,DJ,IN=** command, the **DD=** parameter on the ***START** command is flagged by message IAT7203 as an unrecognized operand, as Example 16-4 shows.

*Example 16-4 *START DJ DD= command*

```
*START,DJ,DD=SPOOL3
IAT7229 DJ05BA (JOB00027): SUCCESSFULLY DUMPED JOB RUNON (JOB00018)
IAT7450 JOB RUNON (JOB00018) PURGED
IAT7229 DJ05BA (JOB00027): SUCCESSFULLY DUMPED JOB WRITRECS (JOB00023)
IAT7450 JOB WRITRECS (JOB00023) PURGED
IAT7230 DJ05BA (JOB00027): DUMP PROCESSING COMPLETE FOR DD NAME SPOOL3
IAT7253 DJ05BA (JOB00027): 0000002 JOBS SUCCESSFULLY DUMPED TO TAPE
IAT7220 DJ05BA (JOB00027): FUNCTION COMPLETE ON UNIT 05BA
IAT7228 ISSUE START OR CANCEL FOR DJ (JOB00027) (05BA)
```

16.3 In-stream data in JCL procedures

JES3 is enhanced in z/OS 2.1 to allow in-stream data sets to be created when processing DD DATA or DD * JCL statements within JCL PROCs and INCLUDEs. The support is provided for batch jobs and started tasks.

16.3.1 Invocation and use

To use the support, create a JCL PROC with an in-stream data set or include an in-stream data set in your JCL by using the INCLUDE statement, as Example 16-5 shows.

Example 16-5 In-stream data in PROCs

```
//INSTRMJB JOB ACCNT#,MSGLEVEL=(1,1),NOTIFY=&SYSUID,MSGCLASS=X
//PROC1    PROC
//STEP1    EXEC PGM=IEBGENER
//SYSIN    DD DUMMY
//SYSPRINT DD SYSOUT=*
//SYSUT2   DD SYSOUT=*
//SYSUT1   DD *
IN-LINE PROC IN-STREAM RECORD1
IN-LINE PROC IN-STREAM RECORD2
/*
//         PEND
//*
//RUNPROC  EXEC PROC=PROC1
//
```

SYSIN data sets are created on the spool to hold the in-stream data. These data sets are not visible when browsing the JESJCLIN data set on spool. However, the SYSIN data sets are visible when the SDSF “Change include SYSIN to ON” option is set.

16.3.2 Migration and coexistence considerations

To support in-steam data in PROCs and INCLUDEs, the job must pass conversion phase under a z/OS 2.1 system. The converted job can then run on down-level systems.

16.4 Multiple simultaneous logons in a Parallel Sysplex

In z/OS 2.1, JES3 is enhanced to allow the same Time Sharing Option/Extensions (TSO/E) user ID to log on to multiple systems in the Parallel Sysplex

16.4.1 Invocation and use

Multiple simultaneous logons by the same TSO/E user ID on different systems in the Parallel Sysplex are now supported. When a TSO/E user is logged on to multiple systems, the following notification rules apply:

1. When a system was specified explicitly, notification messages are sent to that system.
2. When there is no explicit specification, the notification goes to the system from which the TSO/E user submitted the job.
3. When no system specification is found, a search is conducted, starting with the first system defined by MAINPROC. The first occurrence of the TSO/E user ID triggers the notification to be sent to that system.
4. When no user is logged on, the notification is sent to the global.

16.5 8-character job classes on the JOB card

Starting with z/OS V2.1 (JES3 level HJS7790), JES3 supports a job class name of more than one character on the JOB card, up to a maximum of eight characters. JES3 always supported up to eight characters for the job class name, but on the MAIN card (JECL) only. That capability continues to function as it has in earlier JES3 levels.

16.5.1 Invocation and use

The class name on the MAIN card overrides the class name on the job card. For example, the JCL in Figure 16-1 can now be replaced with the JCL in Figure 16-2. Both cause the job to run in the WLMCLASS job class.

```
//WRITRECS JOB MSGCLASS=T,MSGLEVEL=(1,1),CLASS=A
//*MAIN CLASS=WLMCLASS
//GO1      EXEC PGM=WRITRECS
//OUT1     DD SYSOUT=*
```

Figure 16-1 Using MAIN card to override the CLASS= keyword on the JOB card

```
//WRITRECS JOB MSGCLASS=T,MSGLEVEL=(1,1),CLASS=WLMCLASS
//GO1      EXEC PGM=WRITRECS
//OUT1     DD SYSOUT=*
```

Figure 16-2 An 8-character job class on the JOB card

16.5.2 Migration and coexistence considerations

In a JESPlex with main processors at mixed JES3 levels, the JES3 global address space must be at HJS7790 or later for multi-character CLASS names on the JOB card to function properly.

Furthermore, the converter and the interpreter service on a main processor at HJS7790 or later must be available within the JESPlex to convert and interpret the JCL. This CI service can be available from either CI dynamic support programs (DSPs) that are configured within the JES3 global address space or from a CI functional subsystem (FSS) address space if the CI service runs on a main processor at HJS7790 or later. If this CI service is unavailable, the jobs wait until the CI service becomes available.

16.6 SAF control over job class

Resource Access Control Facility (RACF) profiles are added to verify the access of a job's submitter or owner to a job class. This applies to the traditional 36 job classes and 8-character job classes. It does not apply to the STC and TSU special job classes. A authorization is checked during input processing. This ensures that the user ID that is associated with the job has access to the job class associated with the job. A authorization is also checked if a user changes the execution class of a job when the job is held before execution.

16.6.1 Invocation and use

JES2 normally allows any job class to be used. An installation can control job class use by granting access based either on the submitter's profile or on the owner's profile, or both. The control is based on the presence or absence of two FACILITY class profiles and these conditions:

- When the JES.JOBCLASS.OWNER profile is defined in the FACILITY class, job class profiles for owners are enforced.
- When the JES.JOBCLASS.SUBMITTER profile is defined in the FACILITY class, job class profiles for submitters are enforced.
- When both profiles are defined in the FACILITY class, a job class must pass both checks before it is considered valid.
- When either of these profiles is defined, JES3 checks READ access to the following profile in the JESJOBS class:

  ```
  JOBCLASS.nodename.jobclass.jobname
  ```

  Where:

nodename	Local NJE node name
jobclass	Job class to be associated with the job
jobname	Job name of the job that is being submitted

Examples

To deny any job from using CLASS=P, regardless of user ID, use this method:

1. Define a JOBCLASS.NODE1.P.* profile in the JESJOBS resource class.
2. Restrict all users to access of NONE for that entity.

To allow only payroll jobs that are named PAYRxxxx and running under the PAYRLLID user ID to use CLASS=P, use this method:

1. Define a JOBCLASS.NODE1.P.PAYR* profile in the JESJOBS resource class.
2. Permit the PAYRLLID user ID to have READ access to that entity.

16.6.2 Migration and coexistence considerations

The JES3 global main must always be at JES3 2.1 to use this enhancement.

16.7 JES3-related enhancements to JCL

JES3 is enhanced in z/OS 2.1 to support two keywords in the JOB JCL statement:

```
SYSTEM=
SYSAFF=
```

These keywords specify the JES3 system names that are eligible to process the job.

16.7.1 Invocation and use

The use of a SYSTEM= or SYSAFF= keyword overrides the //*MAIN SYSTEM= JECL statement. The same considerations that exist when you use //*MAIN SYSTEM= also apply to using SYSTEM= or SYSAFF= keywords. JGLOBAL and JLOCAL are also acceptable values for the SYSTEM= keyword.

Note: Using both SYSTEM= and SYSAFF= in the same JOB statement is not supported. JES3 ignores IND for SYSAFF= unless it is a valid JES3 system name.

16.8 ENF 70 events

In z/OS 2.1, JES3 issues ENF 70 events to notify listeners of job state changes. These are the supported qualifier codes:

ENF70_SELECT	Indicates that the job was selected by a processing phase conversion, execution, job transmitter, and so on.
ENF70_DESELECT	Indicates that the job was cleared by a processing phase
ENF70_CHANGE	Indicates that the job was queued for a phase; old and phases are identified. This event also includes being requeued to a class or service class queue before execution.
ENF70_PURGE	Indicates that the job was purged.

16.8.1 Invocation and use

Use the ENFREQ macro to listen for ENF 70 events. When an ENF 70 event is signaled, a parameter list that is mapped by the IAZENF70 mapping macro is passed to the listener. By default, JES3 signals ENF 70 events across all systems in the Parallel Sysplex. See *z/OS MVS Programming: Authorized Assembler Services Reference EDT-IXG, SA23-1373,* for more information about ENFREQ (see 16.9, “Documentation and reference” on page 231 for a link).

Use the JOBTRACK= keywords of the OPTIONS JES3 initialization statement to control the signaling scope of ENF 70 events. The default is JOBRACK=SYSPLEX. When JOBTRACK=JGLOBAL is specified, ENF 70 events are signaled only to listeners on the JES3 global. When JOBTRACK=OFF is specified, ENF 70 event signaling is disabled.

The ***MODIFY,CONFIG** command can be used to dynamically update job tracking inishdeck definitions. The ***MODIFY,JOBTRACK** command can be used to dynamically change the signaling scope of ENF 70 events after JES3 has started. Use the ***INQUIRY,OPTIONS** command to display the value of the **JOBTRACK** parameter setting that is in effect.

The syntax of the ***MODIFY,JOBTRACK** command is shown in Example 16-6.

*Example 16-6 *MODIFY,JOBTRACK command syntax*

```
*MODIFY,JOBTRACK=SYSPLEX|JGLOBAL|OFF
```

Example 16-7 on page 231 shows an example of using the ***MODIFY,JOBTRACK** command.

*Example 16-7 Using the *INQUIRY,OPTIONS and *MODIFY,JOBTRACK commands*

```
*I OPTIONS
IAT8646 OPTIONS INQUIRY RESPONSE
  DUMP=PRDMP, DUMPLINS=024576, DUPJOBNM=NO,
  WANTDUMP=YES, LIMIT=03, INTERVAL=10, INTRDR=000020,
  JOBNO=(000001,032767,032767), MT=OFF, SE=10,
  XCFGRPNM=WTSCPLX4, JOBTRACK=SYSPLEX
OPTIONS INQUIRY RESPONSE COMPLETE
*F JOBTRACK=JGLOBAL
IAT8101 JOBTRACK SET TO JGLOBAL
```

16.8.2 Migration and coexistence considerations

ENF 70 listeners can exist on back-level mains, but the JES3 Global main must be at Version 2.1. Otherwise, the ENF 70 events are not signaled.

16.9 Documentation and reference

For documentation and references for the described enhancements, see these publications:

- Running JES3 in a Parallel Sysplex environment, IBM z/OS 2.1.0 Information Center

  https://ibm.biz/BdFdjb
- z/OS JES3, IBM Knowledge Center for z/OS 2.1.0

  https://ibm.biz/BdFdj8
- z/OS JES3, IBM z/OS V2R1 Information Center, documentation

  http://pic.dhe.ibm.com/infocenter/zos/v2r1/index.jsp
- *z/OS MVS Programming: Authorized Assembler Services Reference EDT-IXG, SA23-1373,* for more information about ENFREQ

  https://ibm.biz/BdFd7n
- *z/OS V2R1 Elements and Features Library (PDF files)*

  http://www.ibm.com/systems/z/os/zos/library/bkserv/v2r1pdf/
  - *z/OS V2R1.0 MVS JCL Reference,* SA23-1385
  - *z/OS V2R1.0 MVS Programming: Authorized Assembler Services Reference, Volume 2 (EDT-IXG),* SA23-1373

17

Cross-system coupling facility and cross-system extended services

This chapter describes the various enhancements, functions, and features that were introduced for the cross-system coupling facility (XCF) and cross-system extended services (XES) in IBM z/OS Version 2 Release 1 (2.1).

- 17.1, “Serial rebuild with structure priority change” on page 234
- 17.2, “Documentation and reference” on page 238

© Copyright IBM Corp. 2014. All rights reserved.

17.1 Serial rebuild with structure priority change

This section describes the serial rebuild with structure priority change to the XCF component. It explains the benefit of using serial rebuild processing for coupling facility (CF) events and how to enable it for your IBM Parallel Sysplex.

Unplanned events such as CF failure and loss of connectivity result in rebuilds, break duplexing, and reduplexing that are processed in parallel. Performance problems during these rebuilds can affect availability, because the applications are unavailable during CF structure rebuild processing.

Coupling facility resource management (CFRM) active policy contention is generated when many CF structures are rebuilt at the same time. System functions that depend on important structures can be unavailable for long periods of time, and this problem might be caused by unimportant structures that are competing for CFRM attention.

During this rebuild process, both the systems and CF are highly stressed with XCF and users who are coordinating structure rebuild processing, which increases the contention. Also, couple data set data transfer (I/O) time can be significant, which adds to that. When many CF structures are rebuilt at the same time, performance problems are noticeable.

This function addresses the problem by performing *system-initiated* rebuilds in a more serial manner. It performs these actions in a policy-controlled priority order, which ensures that the most important structures are rebuilt faster.

The support is called *CFLossConnRecoveryManagement*, abbreviated as `CFLCRMGMT`.

The following CF structure rebuild processing triggers are enhanced:

- CF failure or loss of connectivity events
  - System loss of connectivity to a CF, which includes a CF failure
- CF gain connectivity events
- Policy-initiated start duplexing events
  - Policy-initiated CF structure duplexing actions, which includes duplexing a `DUPLEX(ENABLED)` structure after duplexing failover for a CF failure

With `CFLRCRMGMT`, system-initiated rebuilds are performed *almost* serially, but policy-initiated and CF gain-connectivity events are done in a *strictly* serial manner.

Because those rebuilds were previously done in parallel, it might take a long time for a very important structure to become available. Now, you can have this workload protected in a failure scenario, with a faster rebuild, rather than having it delayed due to contention caused by other, less important structures.

Benefits of enabling `CFLCRMGMT`:

- Ensure that the disruption from the CF structure rebuilds and break duplexing activity is as small as possible.
- Ensure that the most important structures in the installation's workload are available quickly.

Benefits are observed whenever you lose connection to a CF or you start a policy-initiated rebuild (start duplex for `DUPLEX=ENABLED` structures) when you gain connectivity to a CF.

CFLCRMGMT does not affect **SETXCF START,REBUILD,LOC=OTHER,CFNM=*cfname*** or **REBUILD,POPCF=*cfname*** requests, where the priorities are not honored. The **REBUILD OTHER** command is still done in parallel, but the **POPCF** does the processing serially, in an alphabetical order.

Tip: For CF maintenance, we suggest that you use **SETXCF START,REALLOCATE** combined with MAINTENANCE MODE rather than using the LOC=OTHER or POPULATECF rebuilds. This is because **REALLOCATE** already does serial rebuilds in a structure-by-structure manner, which causes less impact to the Parallel Sysplex.

17.1.1 Invocation and use

Message-based processing is enhanced with the CFLCRMGMT ability. With CFLCRMGMT, the message-based manager system is responsible for coordinating and initiating the processing for these purposes:

- Recover from a loss of connectivity to a CF
  - Done in a more serial manner than before
- Process a gain of connectivity to a CF or start duplexing
  - All processing strictly serial

After this function is enabled on *all* active systems in the Parallel Sysplex, the message-based manager system can enable CFLCRMGMT

Attention: Message-based processing enablement is required to turn on CFLCRMGMT. It is defined through the **ITEM NAME(MSGBASED) NUMBER(1)** parameter when you are defining the coupling facility resource management (CFRM) couple data set. For more information, see *z/OS MVS Setting up a Sysplex*, SA23-1399, in the IBM Knowledge Center for z/OS 2.1.0.

You can use it by defining the priority of your structures in the CFRM policy. The structure definition statements are updated to include an optional RECPRTY statement.

This statement specifies the rebuild priority to be given to the structure for CFLCRMGMT when the system loses connectivity to a CF. It goes 1 - 4, with one digital decimal value, with these values:

- 1 = highest priority
- 4 = lowest priority

When RECPRTY is not specified, the system defaults, with the following actions:

- RECPRTY(3) is assigned to all structures
- Lock structures are processed before list and cache structures
- RECPRTY does not apply to XCF signaling structures
- RECPRTY takes effect immediately with policy activation

In z/OS 2.1, when you issue a **D XCF, STR** command, it always shows a POLICY RECPRTY if you have it specified for that structure. If CFLCRMGMT is enabled, the command also shows a SYSTEM RECPRTY, which is either the value that is specified or the default value if you did not assign a value.

You can check your structures priorities by issuing the **D XCF** command, as shown in Example 17-1. In this example, we did not specify RECPRTY on the CFRM policy for SYSIGGCAS_ECS, and we specified RECPRTY(1) for ISGLOCK.

Example 17-1 Display XCF,STR command output

```
-D XCF,STR,STRNM=SYSIGGCAS_ECS
 IXC360I  15.04.37  DISPLAY XCF 909
 STRNAME: SYSIGGCAS_ECS
...
 SYSTEM RECPRTY  : 3

-D XCF,STR,STRNM=ISGLOCK
 IXC360I  10.06.22  DISPLAY XCF 032
 STRNAME: ISGLOCK
 ...
   POLICY RECPRTY : 1
```

When a system-initiated rebuild starts with CFLCRMGMT enabled, messages appear. Example 17-2 shows a log sequence where we lost connectivity from system SC74 to the CF CF7A.

Example 17-2 CF LOSSCONN log messages

```
IXC518I SYSTEM SC74 NOT USING 101
        COUPLING FACILITY  002827.IBM.02.00000000B8D7
                           PARTITION: 0E    CPCID: 00
        NAMED CF7A
       REASON: CONNECTIVITY LOST.
       REASON FLAG: 13300001.
IXC568I CF LOSSCONN RECOVERY PROCESSING INITIATED. 102
  MANAGER SYSTEM NAME      : SC74
  MANAGER SYSTEM NUMBER    : 0100023F
  MANAGEMENT LEVEL         : 01052010
...
IXC568I CF LOSSCONN RECOVERY MANAGEMENT SUCCESSFUL.
IXC568I CF LOSSCONN RECOVERY PROCESSING COMPLETED. 321
  EVENT MANAGEMENT PROTOCOL: MESSAGE-BASED
  MANAGER SYSTEM NAME      : SC74
  MANAGER SYSTEM NUMBER    : 0100023F
  MANAGEMENT LEVEL         : 01052010
```

17.1.2 Installation

This XCF function can be implemented by using the **SETXCF** command dynamically or via the COUPLExx PARMLIB member, as shown in Example 17-3 on page 237, Example 17-4 on page 237, Example 17-5 on page 237, and Example 17-6 on page 237.

When using the command dynamically, we suggest that you issue **RO *ALL** to make the change in all systems in the Parallel Sysplex. We also advise doing that only after all of your systems are running on z/OS 2.1.

To activate it via the COUPLExx PARMLIB member, an IPL is required.

Example 17-3 shows the use of the SETXCF command to enable the CFLCRMGMT feature.

Example 17-3 CFLCRMGMT enablement

```
-SETXCF FUNCTIONS,ENABLE=CFLCRMGMT
 IXC373I XCF / XES OPTIONAL FUNCTIONS ENABLED:
         CFLCRMGMT
 IXC548I CFRM EVENT MANAGEMENT ENVIRONMENT UPDATED
 EVENT MANAGEMENT PROTOCOL: MESSAGE-BASED
 REASON FOR CHANGE: MANAGEMENT LEVEL
 TRANSITION SEQUENCE NUMBER: 00000007
 TRANSITION TIME: 04/17/2013 14:24:05.530637
 MANAGER SYSTEM NAME: SC74
 MANAGER SYSTEM NUMBER: 01000239
 MANAGEMENT LEVEL: 01052010
```

Example 17-4 shows the use of the SETXCF command to enable the CFLCRMGMT feature.

Example 17-4 CFLCRMGMT disablement

```
-SETXCF FUNCTIONS,DISABLE=CFLCRMGMT
 IXC373I XCF / XES OPTIONAL FUNCTIONS DISABLED:
         CFLCRMGMT
 IXC548I CFRM EVENT MANAGEMENT ENVIRONMENT UPDATED
 EVENT MANAGEMENT PROTOCOL: MESSAGE-BASED
 REASON FOR CHANGE: MANAGEMENT LEVEL
 TRANSITION SEQUENCE NUMBER: 00000009
 TRANSITION TIME: 04/17/2013 14:27:00.569663
 MANAGER SYSTEM NAME: SC74
 MANAGER SYSTEM NUMBER: 01000239
 MANAGEMENT LEVEL: 01050107
```

Example 17-5 shows the use of the D XCF command to display the status of the CFLCRMGMT feature.

Example 17-5 D XCF,COUPLE output

```
-D XCF,C
 IXC357I  09.26.54  DISPLAY XCF 657
 OPTIONAL FUNCTION STATUS:
   FUNCTION NAME                  STATUS      DEFAULT
   DUPLEXCF16                     DISABLED    DISABLED
   SYSSTATDETECT                  ENABLED     ENABLED
   USERINTERVAL                   DISABLED    DISABLED
   CRITICALPAGING                 DISABLED    DISABLED
   DUPLEXCFDIAG                   DISABLED    DISABLED
   CFLCRMGMT                      ENABLED     DISABLED
```

Example 17-6 show the COUPLExx PARMLIB member syntax to enable CFLCRMGMT.

Example 17-6 COUPLExx syntax

```
FUNCTIONS ENABLE(CFLCRMGMT)
```

17.1.3 Interactions and dependencies

Message-based event and confirmation processing are a prerequisite to enabling a serial rebuild.

All systems in the Parallel Sysplex must enable `CFLCRMGMT` before CFRM can properly process structures in a serial manner.

There are no hardware dependencies. It works on any machine model that supports z/OS 2.1.

17.1.4 Migration and coexistence

There are no toleration or coexistence APARs or PTFs for this change. There is no rolldown to previous z/OS releases.

The function can be enabled one system at a time, although the structures cannot all be processed serially until all systems have enabled CFLCRMGMT. It must remain disabled until all systems within the Parallel Sysplex are at z/OS 2.1 or later.

17.2 Documentation and reference

For more information, see the following publications and web pages:

- *z/OS V2R1 z/OS MVS Programming: Sysplex Services Guide,* SA23-1400

  http://ibm.co/1jcyR9H
- *z/OS V2R1 z/OS MVS Programming: Sysplex Services Reference,* SA38-0658

  http://ibm.co/1jcyR9H
- *z/OS V2R1 z/OS MVS Setting up a Sysplex,* SA23-1399

  http://ibm.co/1pY8nvR
- *z/OS V2R1 z/OS MVS System Messages Vol 10 (IXC - IZP),* SA38-0677

  http://ibm.co/U2eUqq

18

Global resource serialization

This chapter describes the global resource serialization (GRS) enhancements, functions, and features that were introduced with IBM z/OS Version 2 Release 1 (2.1).

In z/OS 2.1 you can change enqueues (ENQs) from *exclusive* to *share* through a macro `ISGENQ` **`REQUEST=CHANGE,CONTROL=EXCLUSIVE`** parameter. The version also adds the support of a Fibre Channel Connection (FICON) channel-to-channel global resource serialization (GRS) Ring mode, which permits GRS-managed channel-to-channel (CTC) connections on an IBM zEnterprise EC12 (zEC12) system.

These changes are described in the following sections:

- 18.1, “ISGENQ macro, REQUEST=CHANGE” on page 240
- 18.2, “FICON channel-to-channel connections” on page 242
- 18.3, “Documentation and reference” on page 243

© Copyright IBM Corp. 2014. All rights reserved.

18.1 ISGENQ macro, REQUEST=CHANGE

Before z/OS 2.1, the ISGENQ **REQUEST=CHANGE** macro could change the status of a previous ISGENQ request only to upgrade from shared to exclusive. You can now use it to downgrade the ENQ from exclusive to shared. ISGENQ **REQUEST=CHANGE** provides now the **CONTROL=SHARED** option, with local ENQ support available in GRS modes and global ENQ support available in Star mode.

This change solves the inability to downgrade ENQs from exclusive to shared. It provides flexibility with ENQs for better processing, which makes it possible to increase throughput.

18.1.1 Invocation and use

Start by passing the CONTROL parameter to the ISGENQ **REQUEST=CHANGE** as the syntax in Example 18-1 shows.

Example 18-1 ISGENQ parameter syntax

```
,REQUEST=CHANGE,CONTROL=SHARED|EXCLUSIVE
```

CONTROL=SHARED changes the ENQ from exclusive to shared. CONTROL=EXCLUSIVE changes the ENQ from shared to exclusive. CONTROL=EXCLUSIVE is the default for compatibility.

If RESLIST=YES is specified, all requests must be for the same control. The CONTROL= keyword applies to the entire list of enqueue tokens. If CONTROL=SHARED is specified and there are ineligible global resources in the resource table, the entire request fails.

Regarding OWNINGTTOKEN, authorized requesters can direct it to another task.

CONTENTIONACT is not applicable, because changing from shared to exclusive still fails if there is contention.

The reason codes **xxxx0C10** and **xxxx0C11** are added for the ISGENQ macro, according to Table 18-1,

Table 18-1 Return and reason codes

Return code	Reason code	Equate symbol and meaning
0C	xxxx0C10	Equate symbol: ISGENQRsn_UnsupportedMode Meaning: The current GRS mode does not support this specific request.

Return code	Reason code	Equate symbol and meaning
0C	xxxx0C11	Equate symbol: ISGENQRsn_MasidNotSupported Meaning: The resource that was the target of the **REQUEST=CHANGE,CONTROL=SHARED** request, currently or at another time, contained MASID users. **REQUEST=CHANGE,CONTROL=SHARED** is not supported for resources that involve MASID requesters.

For more information about the ISGENQ macro and the reason code, see *MVS Authorized Assembler Services Reference EDT-IXG (see*

18.1.2 Interactions and dependencies

The GRS exits now support alternative serialization products.

In addition, the ability to downgrade an ENQ is used by a JCL keyword and JES jobclass attribute, **DSENQSHR**, that can be set on a per-job or per-jobclass basis. This allows better parallelism of batch jobs when they need to access the same data set resource. It is described better in Chapter 29, “Scheduler and Device Allocation” on page 379.

Note: This feature requires JES2 on z/OS 2.1, and it is not available in GRS Ring mode. **DSENQSHR=DISALLOW** is the default on JES3.

Example 18-2 shows the use of DSNENQSHR=ALLOW in a batch allocation environment.

Example 18-2 Use of DSNENQSHR=ALLOW for batch allocation

```
//DSNSHRDN JOB MSGLEVEL=(1,1),DSENQSHR=ALLOW
//STEP1 EXEC PGM=IEFBR14
//DD1 DD DSN=BK.DS1,DISP=OLD <-- Obtain ENQ EXCL for STEP1
//STEP2 EXEC PGM=IEFBR14
//DD2 DD DSN=BK.DS1,DISP=SHR <-- Downgrade from EXCL to SHR after STEP1
//STEP3 EXEC PGM=IEFBR14
//DD3 DD DSN=BK.DS1,DISP=SHR
```

Example 18-3 shows how a multi-step job using generation data groups would ENQ to obtain exclusive access to the GDG in one step and then share it in a later step.

Example 18-3 Sample use for GDGs

```
//DSNSHRDN JOB MSGLEVEL=1
//STEP1 EXEC PGM=IEFBR14
//DDG1 DD DSN=BK.GDG(+1),DISP=(NEW,CATLG), <-- Obtain ENQ EXCL for STEP1
// SPACE=(TRK,1),VOL=SER=DASD01
//STEP2 EXEC PGM=IEFBR14
//DDG2 DD DSN=BK.GDG(+1),DISP=OLD
//STEP3 EXEC PGM=IEFBR14
//DDG3 DD DSN=BK.GDG(+1),DISP=SHR <-- Downgrade after STEP2
```

18.1.3 Migration and coexistence considerations

The GRS enqueue change is tolerated on systems that run earlier version of z/OS through the APAR OA36207, which available for z/OS 1.12 and 1.13. A mixed-level GRS complex encounters wait states without it. These earlier-level systems can participate in the change request, but the change to **CONTROL=SHARED** must be initiated on a z/OS 2.1 system.

18.2 FICON channel-to-channel connections

z/OS 2.1 supports GRS-managed FICON CTCs. If your installation must continue using GRS-managed CTCs (GRS Star is still strongly advised, such as Parallel Sysplex communications for GRS Ring), migrating from ESCON to FICON CTCs becomes possible.

The IBM zEnterprise EC12 (zEC12) system does not support ESCON channels. Therefore, GRS-managed CTCs need to have support for FICON to still be possible on a zEC12 system.

18.2.1 Invocation and use

GRS Star is still recommended for global ENQs, because it provides better performance and scalability. If your installation needs to be configured as a ring, Parallel Sysplex communications is recommended.

The `GRSCNFxx` PARMLIB member syntax remains unchanged.

The existing `ISG046E` has a return code that indicates if the FICON CTC path is lost, as shown in Example 18-4. The GRS displays it as disabled but does not vary it offline and automatically enables the CTC when the path is found.

Example 18-4 ISG046E error message about lost FICON CTC path

```
ISG046E CTC ddd DISABLED DUE TO HARDWARE ERROR – CODE=10
```

18.2.2 Installation

See the August 2012, Issue 26, of the *Hot Topics* newsletter for a detailed explanation and example of migrating to FICON CTC support. (See 18.3, “Documentation and reference” on page 243 for a link.)

Rolling IPLs that following these steps are required to validate the FICON CTCs through a `GRSCNFxx`:

1. Prepare the system for IPL, stopping all applications.
2. Quiesce the system from the ring by issuing the command shown in Example 18-5.

Example 18-5 GRS quiesce command

```
VARY GRS(sysname),QUIESCE
```

3. Stop the system.
4. Purge it from the ring, as shown in Example 18-6.

Example 18-6 GRS purge command

```
VARY GRS(sysname),PURGE
```

5. Re-IPL, using the `GRSCNFxx` member.
6. Check the CTCs to confirm system and CTC status by issuing the **`D GRS`** command.

Repeat steps 1 - 6 for each remaining system in the ring.

18.2.3 Migration and coexistence considerations

The GRS-managed CTC support is hardware-tolerant. It is fully delivered for these earlier releases via APAR OA 38230 and the following program temporary fixes (PTFs):

- z/OS 1.11: PTF UA66451
- z/OS 1.12: PTF UA66452
- z/OS 1.13: PTF UA66453

18.3 Documentation and reference

For more information, see the following publications and web pages:

- *z/OS Hot Topics* newsletter, August 2012, Issue 26

  http://www.ibm.com/systems/z/os/zos/bkserv/hot_topics.html
- *z/OS MVS JCL Reference,* SA23-1385

  http://ibm.co/1qEZvck
- *z/OS MVS JCL User's Guide,* SA23-1386

  http://ibm.co/1mwWOYA
- *z/OS MVS Planning: Global Resource Serialization,* SA23-1389

  http://ibm.co/1suvDib
- *z/OS MVS Programming: Assembler Services Reference IAR-XCT,* SA23-1370

  http://ibm.co/1suvDib
- *z/OS MVS Programming: Authorized Assembler Services Reference EDT-IXG,* SA23-1373

  http://ibm.co/1suvDib

19

System Management Facilities

This chapter describes the enhancements, functions, and features that were introduced for System Management Facilities (SMF) with z/OS Version 2 Release 1 (2.1):

- 19.1, “SMF 30 Instruction Counts” on page 246
- 19.2, “IFASEXIT to stop reading before the end of the logstream” on page 248
- 19.3, “DSPSIZEMAX for SMF logstreams” on page 251
- 19.4, “Documentation and reference” on page 252

© Copyright IBM Corp. 2014. All rights reserved.

19.1 SMF 30 Instruction Counts

The meaning of CPU time becomes less precise with each generation of IBM System z hardware. Additional factors, such as the z/OS image processor use and the cache level of the source data, have a big impact on the CPU time that is used.

The CPU Measurement Facility (CPU MF) counters provide valuable data. You can use this tool, for example, to calculate your workloads' *relative nest intensity* (RNI), which lets you know how much you benefit from the processor's cache architecture. The basic counter set includes details such as the number of executed cycles and instructions for each logical processor physical unit (PU).

In z/OS 2.1, System Management Facilities (SMF) record type 30 provides instruction counts in a Counter Data section (SMF30CDS), sourced from the CPU Measurement Facility HIS (hardware instrumentation services) address space basic counters. With it, record 30 now provides another metric that tends to be less affected by the machine generation and workload (therefore, repetitive) to measure how much work is being done.

19.1.1 Invocation and use

To get instruction counts in the Counter Data section, the system must activate at least the basic counter set from the CPU MF, using HIS. For more information about how to configure and initialize collection by counters, see *Setting Up and Using the IBM System z CPU Measurement Facility with z/OS* (see 19.4, "Documentation and reference" on page 252 for a link).

To enable SMF to produce the Counter Data section on record type 30, you need to specify the `SMF30COUNT` parameter for your `SMFPRMxx` member, and then use the **`SET SMF=xx`** command to activate it, as shown in Example 19-1. The `SMFPRMxx` default is `NOSMF30COUNT`.

Example 19-1 SMF30COUNT in SMFPRMxx

```
SYS1.PARMLIB(SMFPRMVI) - 01.01                 Columns 00001 00072
=>                                                Scroll ===> CSR
************************* Top of Data ******************************
  ACTIVE                       /*ACTIVE SMF RECORDING*/
  DSNAME(SYS1.&SYSNAME..MAN1,SYS1.&SYSNAME..MAN2)
  ...
  MEMLIMIT(NOLIMIT)           /* NO ABOVE THE BAR STORAGE LIMIT*/
  JWT(0010)                   /* 522 AFTER 10 MINUTES*/
  SID(&SYSNAME(1:4))          /* SYSTEM ID IS &SYSNAME  */
  LISTDSN                     /* LIST DATA SET STATUS AT IPL*/
  LASTDS(MSG)                 /*DEFAULT TO MESSAGE          */
  NOBUFFS(MSG)                /*DEFAULT TO MESSAGE          */
  SMF30COUNT                  /*PRODUCES SMF30CDS           */
```

```
SET SMF=VI
IEE974I 17.42.24 SMF DATA SETS 288
          NAME                VOLSER SIZE(BLKS) %FULL  STATUS
        P-SYS1.SC74.MAN1      PAG740      3000    95  ACTIVE
        S-SYS1.SC74.MAN2      PAG740      3000     0  ALTERNATE
IEE388I SMF NOW RECORDING ON VOLSER PAG740, DSN=SYS1.SC74.MAN1 TIME=
                17.42.24
IEE536I SMF      VALUE VI NOW IN EFFECT

D SMF,O
IEE967I 17.42.34 SMF PARAMETERS 292
        MEMBER = SMFPRMVI
        SMFDLEXIT(USER3(IRRADU86)) -- DEFAULT
        SMFDLEXIT(USER2(IRRADU00)) -- DEFAULT
        SMFDPEXIT(USER3(IRRADU86)) -- DEFAULT
        SMFDPEXIT(USER2(IRRADU00)) -- DEFAULT
        EMPTYEXCPSEC(NOSUPPRESS) -- DEFAULT
        NOPERMFIX -- DEFAULT
...
        SMF30COUNT -- PARMLIB
        NOBUFFS(MSG) -- PARMLIB
        LASTDS(MSG) -- PARMLIB
        LISTDSN -- PARMLIB
```

Figure 19-1 Set SMF command and Display SMF command

If the basic counter set deactivates, a disruption occurs and the following happens:

- The system continues processing while instruction counts are unavailable.
- The disruption flag (SMF30_InstCapDisruption) is set to ON.
- Instructions during disruption are missing from SMF record type 30 instruction counts.

Counter Data section

This section contains instruction counts and is similar to the Processor Accounting section fields for CPU times, with the following triplet information:

Offset	SMF30CDO
Length	SMF30CDL
Number	SMF30CDN

These are the SMF 30 Counter Data section fields:

- SMF30_Inst_CP_Task
- SMF30_Inst_CP_NonPreemptSRB
- SMF30_Inst_CP_PreemptSRB
- SMF30_Inst_Offload
- SMF30_Inst_OffloadOnCP
- SMF30_Inst_CP_Enclave
- SMF30_Inst_Offload_Enclave
- SMF30_Inst_OffloadOnCP_Enclave
- SMF30_Inst_CP_DepEnc
- SMF30_Inst_Offload_DepEnc
- SMF30_Inst_OffloadOnCP_DepEnc
- SMF30_InstCaptDisruption
- SMF30_InstCaptLimited

Table 19-1 shows comparable CPU time fields with Counter Data section fields for instruction counts.

Table 19-1 CPU time vs instruction counts

CPU time	Instruction counts
SMF30CPT	SMF30_Inst_CP_Task
	SMF30_Inst_CP_PreemptSRB
	SMF30_Inst_OffloadOnCP
	SMF30_Inst_CP_Enclave
	SMF30_Inst_OffloadOnCP_Enclave
	SMF30_Inst_CP_DepEnc
	SMF30_Inst_OffloadOnCP_DepEnc
SMF30ASR	SMF30_Inst_CP_PreemptSRB
SMF30CPS	SMF30_Inst_CP_NonPreemptSRB
SMF30ENC	SMF30_Inst_CP_Enclave
	SMF30_Inst_OffloadOnCP_Enclave
SMF30DET	SMF30_Inst_CP_DepEnc
	SMF30_Inst_OffloadOnCP_DepEnc
SMF30_TIME_ON_IFA	SMF30_Inst_Offload
SMF30_TIME_ON_SUP	

As you might have noticed, some CPU time fields are the sum of the equivalent instruction count fields, and the reverse is also true.

19.1.2 Installation

1. Activate the CPU Measurement Facility basic counter set, using HIS.
2. Specify the `SMF30COUNT` keyword on your SMFPRMxx member, and activate it by using either the **`SET SMF=xx`** or **`SETSMF`** command.

19.1.3 Interactions and dependencies

To use the System Management Facilities with z/OS 2.1, you must have IBM System z10 Enterprise Class (z10EC) or Business Class (z10BC) or newer hardware.

19.2 IFASEXIT to stop reading before the end of the logstream

IFASEXIT is one interface that enables you to extract data from the SMF logstream. With it, you can get records from one logstream at a time. It runs as a logstream exit routine that is invoked in JCL by using the **`SUBSYS DD`** parameter.

In previous z/OS releases, the IFASEXIT macro always read until the end of the logstream, regardless of the specified date and time. z/OS 2.1 adds a Smart Endpoint (SMEP) parameter that is used to designate a *smart endpoint* value, which results in a calculated time for when to stop browsing the SMF logstream.

The ability to stop reading the logstream data at a designated point before the actual end of it gives you the possibility, depending on your installation, to greatly reduce the IFASEXIT processing time. This produces the greatest benefit when the logstream contains large amounts of data or if you specify a SMEP date that is a significant time before the end of the logstream.

19.2.1 Invocation and use

Example 19-2 shows the syntax of the **SUBSYS DD** statement for IFASEXIT.

Example 19-2 Syntax of SUBSYS JCL for IFASEXIT

```
//ddname DD DSNAME=log.stream.name,
// SUBSYS=(LOGR,IFASEXIT[,'SUBSYS-options1']
// [,'SUBSYS-options2'])
```

SMEP *hhmm*

The **SUBSYS-options2** parameter is used to designate a smart endpoint value, which is specified as *hhmm*. The *hhmm* value describes the number of hours and minutes to use to calculate the endpoint. The total value cannot exceed 2 hours (0200).

hh	A number between 00-02
mm	When *hh* is fewer than 02, this is a number from 00 - 59. When *hh* is 02, the *mm* value must be 00.

If you do not specify the SMEP, the default for IFASEXIT is to read records all the way to the end of the logstream.

SMEP processing uses the following rules to find the logical endpoint in the logstream:

- The smart endpoint time value is calculated by adding the value that is specified by *hhmm* to the time that is specified in the TO=*ending time* value of the **SUBSYS-options1** parameter.
- For each system ID (SID) specified (also in the **SUBSYS-options2** parameter), a table entry is created and marked complete when the SID reaches the smart endpoint time.
- After all of the SIDs are accounted for, the reading of the logstream ends.

Example 19-3 shows an IFASEXIT example with the SMEP parameter.

Example 19-3 SMEP use example

```
//ANTSMF01 JOB MSGLEVEL=(1,1),REGION=0M,CLASS=A
//************************************************
//* RUN IFASEXIT TESTCASE   SMEP(0200) W/ SID(SC74)
//************************************************
//SMFDMP1 EXEC PGM=IEBGENER
//SYSUT1 DD DSNAME=IFASMF.SMFPERM,STREAM1,
//          LRECL=32756,RECFM=VB,
//          BLKSIZE=32760,
// SUBSYS=(LOGR,IFASEXIT,'FROM=(2012/001,00:00),TO=(2012/120,12:00)',
//        'SID(SC74),SMEP(0200)')
//SYSUT2 DD DSN=VILARINS.ANTSMF01.OUTPUT,
```

```
//          UNIT=SYSALLDA,VOL=SER=,
//          DISP=(NEW,PASS),
//          LRECL=32756,RECFM=VB,
//          BLKSIZE=32760,
//          SPACE=(CYL,(1,1),RLSE)
//SYSIN    DD DUMMY
//SYSPRINT DD SYSOUT=A
//*********************************************
//* PRINT TO THE OUTPUT DATASET
//*********************************************
//PRINT1  EXEC PGM=IDCAMS,REGION=4096K
//SMFDS    DD DSN=VILARINS.ANTSMF01.OUTPUT,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD DUMMY
//SYSIN DD *
 PRINT INFILE(SMFDS)
/*
```

This example uses the `IFASMF.SMFPERM.STREAM1` active logstream, which contains all SMF records from 2012. From the **`SUBSYS=`** parameter, we use `LOGR` to set up the input parameter list to call the IFASEXIT exit. IFASEXIT runs through all records, gathering all SMF records for the SID specified (SC74). All other systems are ignored.

Without SMEP, it reads every record in that logstream, so all the records from 2012. With the `SMEP(0200)` keyword, the processing goes from January 1, 2012 until February 1, 2012. The reading stops 2 hours after hitting the **`TO=`** parameter. The final step prints the records to verify that all SMF records within this time frame are recorded in the output data set.

IFA192I now has message inserts when the **`SUBSYS`** options fails validation while processing IFASEXIT syntax checking, as shown in Figure 19-2.

```
-IFA192I SUBSYS OPTIONS FAILED VALIDATION -
   ONLY NUMERIC CHARACTERS ARE FOLLOW FOR SMEP.
   SMEP PARAMETER IS OUT OF THE VALID RANGE.
   SMEP PARAMETER MUST BE IN THE CORRECT FORMAT - SMEP(HHMM)
   ONLY ONE SID CAN BE SPECIFIED
```

Figure 19-2 IFA192I messages

For more information, see *z/OS MVS System Management Facilities (SMF)*, SA38-0667 (see 19.4, “Documentation and reference” on page 252 for a link).

19.2.2 Migration and coexistence

APAR OA37604 is toleration. It can be installed on earlier z/OS release systems to allow the SMEP keyword to be tolerated, although the smart endpoint processing does not take effect:

- An system that is running an earlier version without OA37604 installed fails IFASEXIT jobs that contain the SMEP keyword.
- An earlier-version system with OA37604 installed allows the SMEP keyword but does not process the smart endpoint function. The SMEP keyword is ignored.

19.3 DSPSIZEMAX for SMF logstreams

SMF logstream recording might need to buffer data when, for example, there is an external logger failure. When that happens, it is possible that 2G of storage can be used by SMF for each logstream that is disconnected. This buffer space shortage can cause data loss.

To avoid this situation, you can now set a limit on the maximum size of the buffer area that would be used by SMF. The SMFPRMxx DSPSIZMAX keyword for logstreams is similar to BUFSIZMAX for data sets.

19.3.1 Invocation and use

This option is enabled by setting the DSPSIZMAX keyword on the PARMLIB SMFPRMxx member or by using the **SETSMF** command, as shown in Example 19-4. It can be specified for all logstreams or per LSNAME or DEFAULTLSNAME. The value range for this keyword is from 128 - 2048 M. It defaults to 2048 M (2 G). SMF record type 23 brings a segment, SMF23LGS, which contains values that allow you to verify a shortage of logstream buffers.

Example 19-4 DSPSIZMAX syntax

```
SYS1.PARMLIB(SMFPRMVI) - 01.03               Member SMFPRMVI saved
>                                                 Scroll ===> CSR
************************ Top of Data ******************************
 ACTIVE                        /*ACTIVE SMF RECORDING*/
 DSNAME(SYS1.&SYSNAME..MAN1,SYS1.&SYSNAME..MAN2)
 NOPROMPT                      /* NO PROMPT  JUST DO IT  */
 FLOOD(ON)
 REC(PERM)                     /*TYPE 17 PERM RECORDS ONLY*/
 INTVAL(10)
...
 SID(&SYSNAME(1:4))           /* SYSTEM ID IS &SYSNAME  */
 LISTDSN                      /* LIST DATA SET STATUS AT IPL*/
 LASTDS(MSG)                  /*DEFAULT TO MESSAGE          */
 NOBUFFS(MSG)                 /*DEFAULT TO MESSAGE          */
 DSPSIZMAX(1G)                /*LIMITS BUFFER AREA FOR LS   */
 DEFAULTLSNAME(IFASMF.DEFAULT,DSPSIZMAX(1G))
 LSNAME(IFASMF.STREAM1,TYPE(30),DSPSIZMAX(512M))
...

or via SETSMF command:

SETSMF DSPSIZMAX(1G)
SETSMF LSNAME(IFASMF.DEFAULT,DSPSIZMAX(1G))
SETSMF DEFAULTLSNAME(IFASMF.DEFAULT,DSPSIZMAX(512M))

-D SMF,O
...
     SYS(EXITS(IEFU83)) -- PARMLIB
     SYS(TYPE(30,70:79)) -- PARMLIB
     DSPSIZMAX(0001G) -- PARMLIB
     SMF30COUNT -- PARMLIB
     NOBUFFS(MSG) -- PARMLIB
...
```

19.3.2 Migration and coexistence

If you are sharing SMFPRMxx across multiple releases:

- OA35175 for z/OS Versions 1.12 and 1.13 has static DSPSIZMAX support. It cannot be activated dynamically by **SETSMF** or **SET SMF=xx** commands. It is set only during IPL.
- OA35619 for z/OS Versions 1.10 and 1.11 is toleration. It recognizes but ignores the keyword.

19.4 Documentation and reference

For more information about topics in this chapter, check the following PDF files that are available to download from the z/OS V2R1 Elements and Features Library:

http://www.ibm.com/systems/z/os/zos/library/bkserv/v2r1pdf/

- *z/OS MVS Initialization and Tuning Reference,* SA23-1380
- *z/OS MVS System Commands,* SA38-0666
- *z/OS MVS System Management Facilities (SMF)*, SA38-0667

Also check this IBM Redbooks publication:

Setting Up and Using the IBM System z CPU Measurement Facility with z/OS, REDP-4727

http://www.redbooks.ibm.com/abstracts/redp4727.html?Open

20

Resource Measurement Facility

This chapter describes the enhancements, functions, and features that were introduced for the Resource Measurement Facility (RMF) feature with z/OS Version 2 Release 1 (2.1):

- 20.1, “zIIP enablement” on page 254
- 20.2, “XML support for additional RMF postprocessor reports” on page 255
- 20.3, “RMF zHybrid SMF” on page 261
- 20.4, “RMF zHybrid Microsoft Windows support” on page 267
- 20.5, “Documentation and reference” on page 272

© Copyright IBM Corp. 2014. All rights reserved.

20.1 zIIP enablement

Performance monitoring products can increase the total cost of ownership (TCO) of your z/OS environment. To minimize this effect, these products started using the IBM System z Integrated Information Processor (zIIP).

With RMF in z/OS 2.1, the Monitor III data gatherer (RMFGAT) offloads part of its work to zIIPs, which frees general purpose central processors (CPs) and decreases installation cost.

20.1.1 Invocation and use

The RMFGAT is enabled for zIIP use by default.

When you have at least one zIIP processor online, RMFGAT partially offloads its work to this engine automatically.

This use can be controlled initially by the Monitor III PARMLIB **ZIIPUSE** parameter. However, you do not need to specify it, because it is the default. Example 20-1 shows the Monitor III start log without the parameter specified.

Example 20-1 RMF III start options log

```
ERB100I RMF: ACTIVE
...
ERB103I III: OPTIONS IN EFFECT
ERB103I III:   MASTER  -- DEFAULT
ERB103I III:   ZIIPUSE  -- DEFAULT
ERB103I III:   NOLOCK  -- MEMBER
ERB103I III:   NOSGSPACE  -- MEMBER
ERB103I III:   ZFS  -- MEMBER
...
ERB103I III:   DATASET(NOSWITCH)  -- MEMBER
ERB103I III:   DATASET(STOP)  -- MEMBER
ERB103I III:   CYCLE(1000)  -- MEMBER
ERB100I III: ACTIVE
...
```

20.1.2 Installation

Because the **ZIIPUSE** parameter is the default, there is nothing necessary to activate it. Just start RMFGAT with your current member. If you want to disable or enable it, issue the command shown in Example 20-2.

Example 20-2 RMF III ZIIPUSE enablement/disablement command

```
F RMF,F III,ZIIPUSE|NOZIIPUSE
```

You can also specify the option parameter on the PARMLIB member `ERBRMF04`, as shown in Example 20-3 on page 255.

Example 20-3 ERBRMF04 option ZIIPUSE

```
SYNC(00)       /* MINTIME SYNCHRONIZATION */
SYSOUT(A)      /* MESSAGES TO SYSOUT CLASS A */
WSTOR(32)      /* SIZE OF INSTORAGE BUFFER (IN MB) */
ZIIPUSE        /* PARTIAL USE OF zIIP ENGINES */
IOSUB          /* I/O SUBSYSTEM GATHERING ACTIVE */
CFDETAIL       /* COUPLING FACILITY DETAILS */
CACHE          /* ACTIVATE CACHE GATHERING */
VSAMRLS        /* ACTIVATE VSAM RLS GATHERING */
OPD            /* ACTIVATE OMVS PROCESS DATA GATHERING */
```

20.1.3 Interactions and dependencies

The z/OS system image needs at least one available zIIP processor. Otherwise, the monitoring product will run on CPs.

20.1.4 Migration and coexistence

RMF Monitor III Data Gatherer zIIP use is enabled automatically in z/OS 2.1.

20.2 XML support for additional RMF postprocessor reports

Since z/OS 1.11, some of the RMF postprocessor (PP) reports have been generated in XML format as an alternative to the standard text format. XML format enables browser-based display and the capability of XML parsing. In z/OS 2.1, RMF contains all of the basic PP reports in XML. This provides a better user interface and a standardized format for API access for SMF type 7x data.

The RMF Distributed Data Server (DDS) has long been one of the key components for the IBM Parallel Sysplex-wide access to Monitor III data. Since RMF for z/OS 1.12, DDS also supports HTTP requests to retrieve RMF PP data from RMF PP reports. Because the data is returned as XML document, a web browser (RMF Data Portal) can act as a portal to retrieve and display RMF PP data.

The RMF Spreadsheet Reporter and RMF XML Postprocessor Toolkit work with XML reports.

RMF had its standard text output limited to a width of 132 bytes. Because of that, some reports appeared somehow squeezed. This can be observed particularly in overview (OVW) reports with a high number of OVW conditions. The basic text format does not take advantage of any presentation capability, such as resizing windows and scrolling back and forth. Depending on the report type, layouts were different. Therefore, API programs need to be tailored according to each specific report type in order to extract data out of the reports. As an alternative to that basic text type, RMF PP reports since z/OS 1.11 can be generated in XML format.

The following reports are available in XML for each z/OS release:

V1R11 (1.11)	Processor activity Crypto activity FICON director activity OMVS kernel activity ESS disk systems activity Overview reports
V1R12 (1.12)	WLMGL workload activity DEVICE activity
V1R13 (1.13)	PAGING activity SDELAY serialization delay
V2R1 (2.1)	SDEVICE shared device activity XCF cross-system coupling activity CACHE subsystem activity CF coupling facility activity CHANnel path activity ENQueue activity HFS hierarchical file system IOQ I/O queuing activity PAGESP page data set activity VSTOR virtual storage activity report

As you can see, z/OS 2.1 makes nearly all standard RMF PP reports available in XML.

Attention: Summary and exception reports, plus interval reports based on data that is collected during a monitor II background session, are not available in XML format.

The diagram in Figure 20-1 on page 257 shows the hierarchy of XML tags for PP reports:

- A <postprocessor> tag starts a section that contains the data for one report type (such as CPU or CRYPTO) that belongs to one interval.
- Each <postprocessor> section contains header information (report name and description, system name, date, time, and version information).
- One or more sections follow that contain tables and lists name-value pairs that represent the report data.

  The complete description for the syntax of the XML data generated by the RMF PP can be found in the *RMF Programmer's Guide* in Chapter 3, "How to interpret an XML document returned by the DDS."
- The diagram shows one hierarchy level of tags allowed after the <postprocessor> tag.
- The <postprocessor> tag contains all header information for one report that belongs to one interval.
- This header information is followed by the performance data, which is surrounded by several types.
- Within these <segments>, the data is organized into tables or name-value pairs.

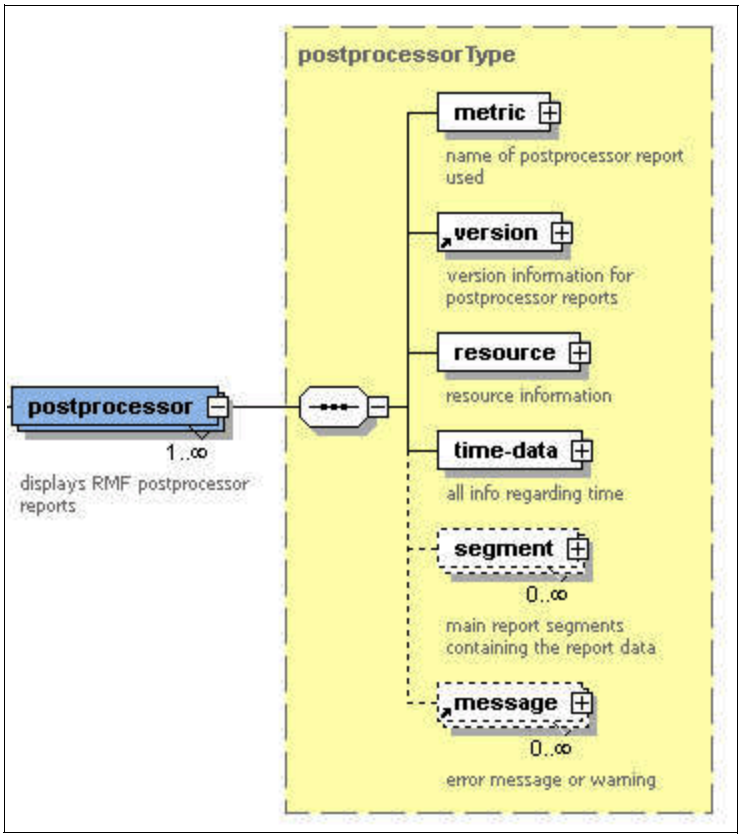


Figure 20-1 XML syntax hierarchy

The complete description of the syntax of the XML data generated by the RMF PP is in the *Resource Measurement Facility Programmer's Guide*, Chapter 3, "How to interpret an XML document returned by the DDS" (see 20.5, "Documentation and reference" on page 272). Example 20-4 shows XML syntax that follows the format shown in Figure 20-1.

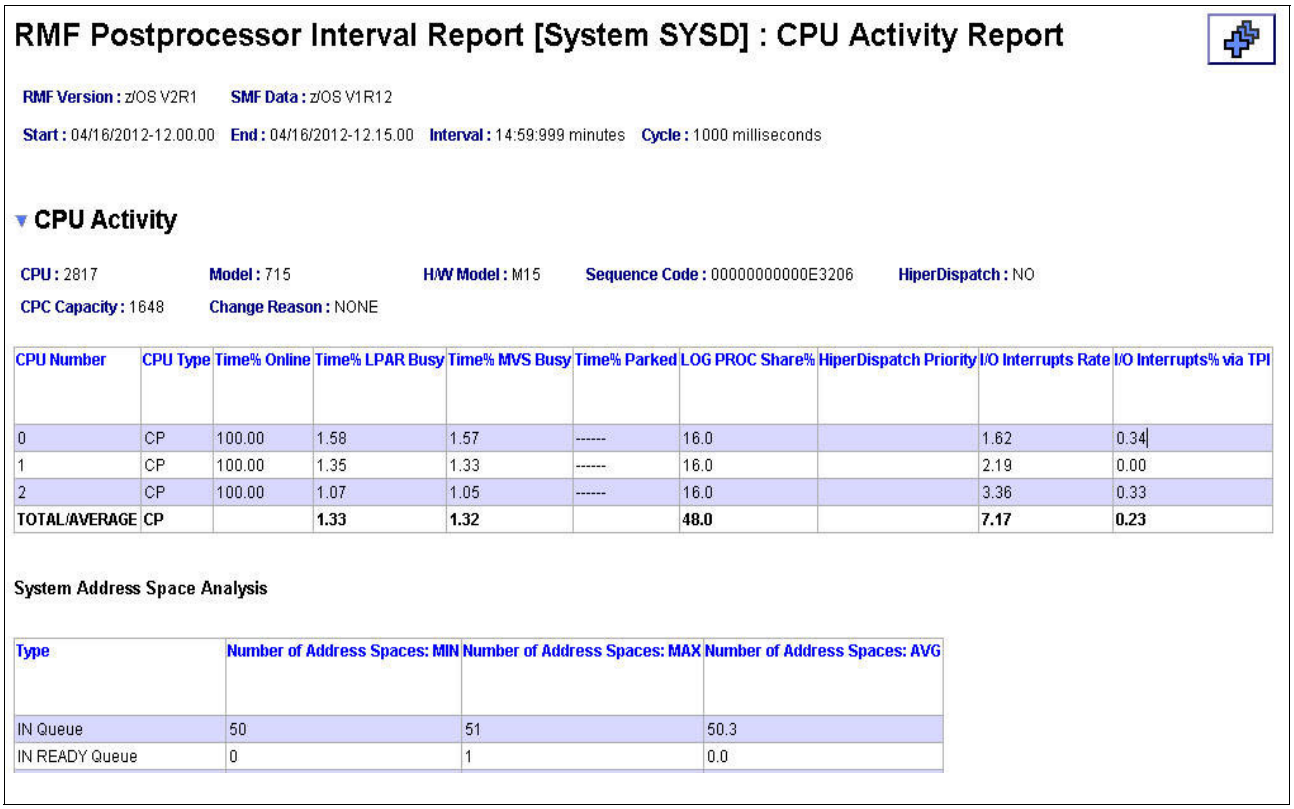
Example 20-4 XML syntax example for CPU Activity Report header information

```
<postprocessor>
  <metric id="CPU">
    <description>CPU Activity Report</description>
    <type>Interval</type>
  </metric>
  <version>
    <smf-data>z/OS V1R12</smf-data>
    <rmf-report>z/OS V2R1</rmf-report>
  </version>
  <resource>
    <resname>SYS5</resname>
    <restype>SYSTEM</restype>
  </resource>
  <time-data>
    <display-start locale="en-us">04/16/2012-12.00.00</display-start>
    <display-end locale="en-us">04/16/2012-12.15.00</display-end>
    <report-interval unit="minutes">14:59:999</report-interval>
    <cycle unit="milliseconds">1000</cycle>
  </time-data>
```

Figure 20-2 is an example of a CPU Activity Report that is displayed in a web browser by using the XSL stylesheet that is provided by the RMF.

It shows the following information:

Report header	Name of report (CPU Activity Report), interval start and end time, RMF and SMF versions
Segment	CPU Activity in this example (can be also Partition Data, Group Capacity, and Cluster Report for the CPU Activity Report)
Name-value pairs	CPU, model, HiperDispatch
Tables	CPU times and rates, system address space analysis



RMF Postprocessor Interval Report [System SYSD] : CPU Activity Report

RMF Version : z/OS V2R1 SMF Data : z/OS V1R12

Start : 04/16/2012-12.00.00 End : 04/16/2012-12.15.00 Interval : 14:59:999 minutes Cycle : 1000 milliseconds

CPU Activity

CPU : 2817 Model : 715 H/W Model : M15 Sequence Code : 00000000000E3206 HiperDispatch : NO

CPC Capacity : 1648 Change Reason : NONE

CPU Number	CPU Type	Time% Online	Time% LPAR Busy	Time% MVS Busy	Time% Parked	LOG PROC Share%	HiperDispatch Priority	I/O Interrupts Rate	I/O Interrupts% via TPI
0	CP	100.00	1.58	1.57	------	16.0		1.62	0.34
1	CP	100.00	1.35	1.33	------	16.0		2.19	0.00
2	CP	100.00	1.07	1.05	------	16.0		3.36	0.33
TOTAL/AVERAGE	**CP**		**1.33**	**1.32**		**48.0**		**7.17**	**0.23**

System Address Space Analysis

Type	Number of Address Spaces: MIN	Number of Address Spaces: MAX	Number of Address Spaces: AVG
IN Queue	50	51	50.3
IN READY Queue	0	1	0.0

Figure 20-2 CPU Activity Report in a web browser

20.2.1 Invocation and use

Generation of PP reports in XML format is controlled by JCL DD statements that contain these names:

- XPRPTS (single-system)
- XPRPTS (single-system)
- XPXSRPTS (Parallel Sysplex)
- XPOVWRPT (Overview)

Table 20-1 on page 259 shows the XML DDNAMES properties.

Table 20-1 XML Postprocessor DDNAMES

DDNAME	Contents	Allocations	Notes
XPRPTS	Combined single-system report in XML format	One DD statement for one data set to contain all single reports for each interval during the session	There is no dynamic allocation for this DDNAME. You must define it explicitly if you want to get all reports in XML format into one data set or output class. If you define this DDNAME, no MFRnnnnn files are created. If you define this DDNAME and PPRPTS, no XML output in file XPRPTS is created.
XPOVWRPT	Combined overview report in XML format	One DDNAME for one data set to contain all overview reports for each system included in the input data	There is no dynamic allocation of this DDNAME, you must define it explicitly if you want to get all overview reports in XML format into one data set or output class. If you define this DDNAME, no PPORPnnn files are created.
XPXSRPTS	Combined Parallel Sysplex-wide report in XML format	One DDNAME for one data set to contain all Parallel Sysplex reports for each interval included in the input data	There is no dynamic allocation of this DDNAME, you must define it explicitly if you want to get all overview reports in XML format into one data set or output class. If you define this DDNAME, no MFRnnnnn files are created. If you define this DDNAME and PPXSRPTS, no XML output in file XPXSRPTS is created.

If one of those three statements is specified in the JCL for postprocessor, XML output is generated. If both requests for text and XML are given at the same time, the following rules apply:

- If you specify a DD card for cumulative text output, such as PPRPTS or XPRPTS, only text output is generated to PPRPTS, and no XML is produced.
- If single-system reports are requested on the REPORTS control statement but they are not XML-enabled and the DD card XPRPTS is defined, no output is generated for them.

The DDNAME XPXSRPTS was introduced in z/OS 1.12 to support XML-formatted PP Parallel Sysplex reports. For this DDNAME, similar rules apply:

- If you specify the PPXSRPTS along with XPXSRPTS, no XML output is generated.

Example 20-5 is a sample RMF postprocessor job to generate XML output for the system, Parallel Sysplex, and overview reports, passing all three DDNAMES.

Example 20-5 RMF postprocessor sample JOB that uses XML DDNAMES

```
//ERBSAMPP JOB (ACCT),'PGMRNAME',CLASS=A,REGION=32M,...
//RMFPP EXEC PGM=ERBRMFPP
//MFPINPUT DD DISP=SHR,DSN=<Input_SMF_Data>
//MFPMSGDS DD SYSOUT=*
//XPRPTS DD SYSOUT=*      CUMULATIVE XML OUTPUT OF SYSTEM REPORTS
//XPOVWRPT DD SYSOUT=*    DTO. OF OVERVIEW REPORTS
//XPXSRPTS DD SYSOUT=*    DTO. OF SYSPLEX REPORTS
//SYSOUT DD SYSOUT=*
//SYSIN DD *
NOSUMMARY
REPORTS(CPU,CRYPTO,FCD,SDELAY)
SYSRPTS(CF,WLMGL)
OVW(BATCH (SSCHRT(S.BATCH)))
OVW(BATCHPRD(SSCHRT(S.BATCHPRD)))
SYSOUT(A)
```

In this example, all three types of DD statements for XML output are used:

- XPRPTS uses the REPORTS control statement
- XPOVWRPT uses the OVW control statement
- XPXSRPTS uses the SYSRPTS control statement

SDSF output lists the DDNAMES for the generated XML output for the requested system, Parallel Sysplex, or overview reports after the RMF PP job ends, depending on these stipulations:

- If the JCL states SYSOUT=* for the XPRPTS, XPXSRPTS, or XPOVWRPT DD cards, the XML output can be found on the JES spool after the job is finished.
- If the XML output is routed to a permanent data set, you can find it in that data set after the job terminates.

RMF Postprocessor XML Toolkit

In z/OS 2.1, the RMF includes the Postprocessor XML Toolkit, which assists you in browsing PP XML reports with your Internet browser. The reports are formatted by use for RMF XSL stylesheets. Here is a quick installation walkthrough:

- The toolkit is part of the RMF product. Application files and the installation utility are provided in the ERBXMLTK member of the SERBPWSV library. Download it as a binary file: erbxmltk.msi
- Install the MSI package by using the Microsoft Windows installer, either by double-clicking the package file or by issuing this command:

  ```
  msiexec /package erbxmltk.msi [/qn]
  ```

  The Windows installer guides you through installation.

The Postprocessor XML Toolkit is installed in the IBM RMF Performance Management program group.

The installation process extracts all files that are necessary to format and display the XML reports (XSL stylesheet, JavaScript, and bitmap files) to the toolkit directory. In the directory, you can find tips on how to use the XML format reports.

To view an XML PP report, follow these steps:

1. Run an RMF PP job and direct the XML output data to a data set (DDNAMES: XRPTS, XPOVWRPT, or XPXSRPTS).
2. Download the XML output data set to the Postprocessor XML Toolkit directory on your workstation with the `.xml` file extension. Be sure to download the data set that contains the XML output of the postprocessor reports in ASCII format to the Postprocessor XML Toolkit directory.
3. Use your Internet browser to open the XML PP reports within the Postprocessor XML Toolkit.

The reports are formatted into an HTML document when you open them within the Postprocessor XML Toolkit by using your browser. The created XML PP reports contain a link to the stylesheet files that are required to format the reports. These stylesheets are available in a subdirectory of the PP XML Toolkit.

In the subdirectory example, you can find more information about how to use RMF PP XML reports.

RMF Spreadsheet Reporter - XML Support

The RMF Spreadsheet Reporter provides built-in support for the PP XML-formatted reports. You can request the XML format by using the general option to “Use XML Report Format.” This causes the following results:

- The report selection list in the Reports tab displays only the report types that can be generated in XML format.
- The generated JCL contains the DDNAMES for the XML format (XPRPTS and XPOVWRPT).
- The default file type for local report listings changes from `.lis` to `.xml`.

The XML-formatted reports can be instantly displayed within the Spreadsheet Reporter. After a listing is associated with the XML file type, the View action opens a browser window with an unformatted report. Report headers are displayed as name-value pairs, and the report body is in tabular format.

Internally, the XML uses the following stylesheets, which are in the Spreadsheet Reporters Listing directory:

- `ddsml-pp.xsl`
- `ddsml.css`

20.3 RMF zHybrid SMF

RMF zHybrid resource monitoring persistence introduces a function as part of RMF Cross Platform Monitoring (RMF XP). It has been included with z/OS since Version 1.13. RMF XP collects performance data from platforms other than z/OS: IBM AIX®, Linux, and Microsoft Windows. This data can now be stored persistently as the SMF record type 104.

The IBM zEnterprise System integrates z/OS and distributed environments under the same hardware. It requires that powerful management functions and performance tools are available for the platforms, because they are for z/OS.

With SMF storing performance data collected by RMF XP under the record type 104, the same infrastructure that is already running in z/OS can be used to archive performance data for the AIX, Linux, and Windows platforms. This allows system programmers to analyze cross-platform performance data by using almost the same set of postprocessing tools as they use for z/OS.

Data collection from remote IBM AIX and Linux systems is done by the GPM4CIM server. Previously, it was not capable of storing the data persistently for later analysis. With z/OS 2.1 RMF, the GPM4CIM server can write the SMF record type 104 to the active SMF buffer.

Important: To write SMF record type 104, the task started on the GPM4CIM server needs at least READ access to the BPX.SMF profile of the `FACILITY` class. This is specified with your System Authorization Facility (SAF) interface.

Alternatively, a copy of the SMF record image can be written to the SMF buffer of the RMF Parallel Sysplex data server. This is not an RMF XP capability. It uses the existing infrastructure.

Record type 104 stores performance data from all distributed platforms, separated by subtype ranges. According to each platform-specific resource model, one subtype number is associated with a resource type respective metric. To provide enough space for future extensions, a range of 20 subtypes is reserved for each platform.

Table 20-2 shows the ranges for SMF record 104 allocated subtypes.

Table 20-2 SMF record 104 subtype range

Platform	Subtype range
IBM AIX	1 - 12 (13 - 19 reserved)
IBM Linux on System x®	20 - 31 (32 - 39 reserved)
IBM Linux on System z	40 - 53 (54 - 60 reserved)
Microsoft Windows	60 - 64 (65 - 80 reserved)

Table 20-3 on page 263 gives an overview about the currently allocated SMF 104 subtypes for the supported platforms:

Table 20-3 SMF 104 allocated subtypes

Platform	Subtype	Metric
AIX on IBM System p®	1	AIX_ActiveMemoryExpansion
	2	AIX_Processor
	3	AIX_ComputerSystem
	4	AIX_Disk
	5	AIX_NetworkPort
	6	AIX_FileSystem
	7	AIX_Memory
	8	AIX_OperatingSystem
	9	AIX_Process
	10	AIX_ShareEthernetAdapter
	11	AIX_ActiveMemorySharing
	12	AIX_VirtualTargetDevice
Linux on System x	20	Linux_IPProtocolEndpoint
	21	Linux_LocalFileSystem
	22	Linux_NetworkPort
	23	Linux_OperatingSystem
	24	Linux_Processor
	25	Linux_UnixProcess
	26	Linux_Storage
	30	Linux_KVM
	31	Linux_Xen
Linux on System z	40	Linux_IPProtocolEndpoint
	41	Linux_LocalFileSystem
	42	Linux_NetworkPort
	43	Linux_OperatingSystem
	44	Linux_Processor
	45	Linux_UnixProcess
	46	Linux_Storage
	50	Linux_zCEC
	51	Linux_zLPAR
	52	Linux_zChannel
	53	Linux_zECKD

Platform	Subtype	Metric
Windows on System x	60	Windows_LocalFileSystem
	61	Windows_NetworkPort
	62	Windows_OperatingSystem
	63	Windows_Processor
	64	Windows_Storage

For more information about SMF 104 and its subtypes, see *z/OS V2R1.0 MVS System Management Facilities (SMF), SA38-0667-02*, in the z/OS V2R1 Elements and Features Library (see 20.5, “Documentation and reference” on page 272 for a link).

The structure of the SMF record type 104 follows the general guidelines for SMF records, including the common header. All sections can be addressed by using triplets, where a triplet describes individual sections with offset, length, and number. This enables you to extract data from the SMF record with standard SMF processing tools. Figure 20-3 shows the layout for the SMF 104 header.

Offsets		Name	Length	Format	Description
Common header for SMF record type 104					
0	0	SMF104LEN	2	binary	Record length. This field and the next field (total of four bytes) form the RDW (record descriptor word). See “Standard SMF record header” on page 154 for a detailed description.
2	2	SMF104SEG	2	binary	Segment descriptor (see record length field).
4	4	SMF104FLG	1	binary	System indicator: **Bit** — **Meaning When Set** **0** — New record format **1** — Subtypes used **2** — Reserved **3-6** — Version indicators* **7** — System is running in PR/SM mode *See “Standard SMF record header” on page 154 for a detailed description.
5	5	SMF104RTY	1	binary	Record type 104 (X'68').
6	6	SMF104TME	4	binary	Time since midnight, in hundredths of a second, that the record was moved into the SMF buffer.
10	A	SMF104DTE	4	packed	Date when the record was moved into the SMF buffer, in the form 0*cyyddd*F. See “Standard SMF record header” on page 154 for a detailed description.
14	E	SMF104SID	4	EBCDIC	System identification (from the SMFPRMxx SID parameter).
18	12	SMF104SSI	4	EBCDIC	Subsystem identification (GPM).
22	16	SMF104STY	2	binary	Record subtype.
24	18	SMF104TRN	2	binary	Number of triplets in this record. A triplet is a set of three SMF fields (offset/length/number values) that defines a section of the record. The offset is the offset from the RDW.
26	1A		2		Reserved.
28	1C	SMF104PRS	4	binary	Offset to RMF XP product section from the RDW.
32	20	SMF104PRL	2	binary	Length of RMF XP product section.
34	22	SMF104PRN	2	binary	Number of RMF XP product sections.
Header extension for all subtypes					
36	24	SMF104ICS	4	binary	Offset to image control section from the RDW.

SMF Header with common layout

Figure 20-3 SMF header layout

SMF 104 RMF Product section is slightly different from the SMF 70-79 records, because a subset of information is not available for the platforms, such as cycle length. However, the SMF 104 RMF Product section contains information that is not available in a z/OS environment, such as SMF104CPX, SMF104OSL, and SMF104PLT.

The SMF type 104 record contains performance data that is collected from one or more systems images. There is one image control section for each image. Beyond general information, the image control section describes the corresponding metric sections with the measured elements and their metric values. Figure 20-4 shows the record mappings for this record.

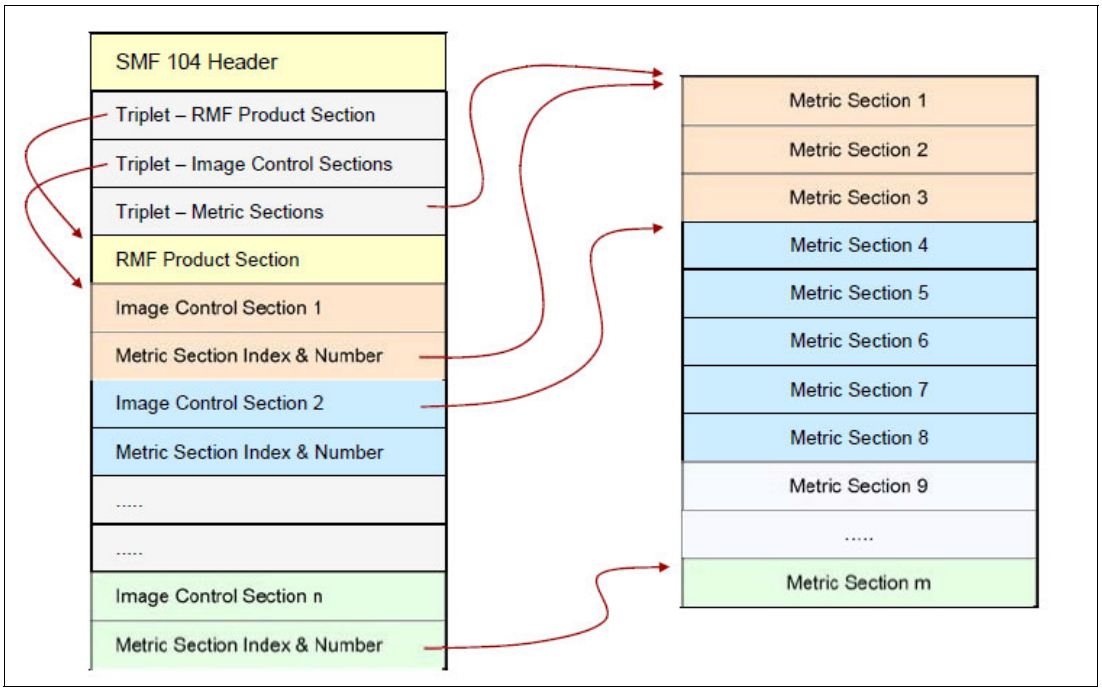


Figure 20-4 SMF record type 104 mappings

All image control sections are located consecutively, behind the RMF product section. In the same way, all metric sections are behind the last image control section. The metric section index and number within the image control section helps you identify the metric sections that belong to a certain image.

Because one SMF record is limited to 32756 bytes, an individual record can be split. All metric sections for a certain image might not be in the same record. However, one record is self-contained. A meaningful report can be produced by using only one record, with no need to combine multiple records from the same subtype for reporting purposes.

The RMF ERBSCAN utility supports the SMF type 104. Beyond the header sections, the content of the image sections and of the metric sections is displayed in hexadecimal and character formats. The individual metric values are stored in double floating point format. ERBSCAN translates all of the values to the decimal format and displays it right behind the hex dump section.

20.3.1 Invocation and use

To activate SMF 104 record writing by GPM4CIM, you must specify the RECORD option within the RMF XP configuration file. The default is NORECORD. The scope of this option is

global, meaning that the option is applied to all record subtypes. To change this dynamically, issue the command in Example 20-6.

Example 20-6 Modify GPM4CIM command

```
F GPM4CIM,RECORD|NORECORD
GPM253I SMF RECORDING IS NOW ON/OFF
```

Example 20-7 shows a sample RMF XP configuration file (gpm4A.cfg for AIX), including the RECORD parameter. RECORD(NO) is the default.

Example 20-7 Sample RMF XP configuration file

```
MAXSESSIONS_HTTP(20) /* MaxNo of concurrent HTTP requests */
HTTP_PORT(8805) /* Port number for HTTP requests */
HTTP_ALLOW(*) /* Mask for hosts that are allowed */
HTTP_NOAUTH(*) /* No server can access without auth.*/
INTERVAL(300) /* Length of the monitoring interval */
AIX_COMPLEX(WEBPLEX)                        /* Name of system complex */
AIX_IMAGE(p6rmf1.bb.com.br:5988)            /* Hostname of member */
AIX_IMAGE(p6rmf2.bb.com.br:5988)            /* */
RECORD(YES) /* Write SMF Records */
```

You can control the SMF 104 recording at the subtype level through the SMFPRMxx PARMLIB member. You can also switch it dynamically by using the **SET SMF** command. The RMF Sysplex Data Server receives and holds real-time copies of SMF records, which can be applied to the SMF 104 records. Example 20-8 shows an SMFPRMxx member that includes record 104 subtypes (TYPE statement).

Example 20-8 SMFPRMxx including 104 subtypes

```
CTIVE                         /* ACTIVE SMF RECORDING*/
DSNAME(SYS1.&SYSNAME..MAN1,
SYS1.&SYSNAME..MAN2,
SYS1.&SYSNAME..MAN3)
MAXDORM(3000)                  /* WRITE AN IDLE BUFFER AFTER 30 MIN */
MEMLIMIT(20000M)               /* MEMLIMIT ABOVE THE BAR BHIM */
STATUS(010000)                 /* WRITE SMF STATS AFTER 1 HOUR */
JWT(0900)                      /* 522 AFTER 15 HOURS */
SID(&SMFID)                    /* SYSTEM ID IS SYSBLD */
LISTDSN                        /* LIST DATA SET STATUS AT IPL */
INTVAL(15)
SYNCVAL(00)
SYS(TYPE(30,42,70:79,103,104(1:12,20:31,40:53,60:64),108),
EXITS(IEFU83,IEFU84,IEFU85,IEFACTRT,IEFUJV,IEFUSI,
IEFUJP,IEFUSO,IEFUJI,IEFUTL,IEFU29,IEFUAV),
INTERVAL(SMF,SYNC),NODETAIL)
```

Example 20-9 shows how to control SMF buffering on the subtype level with the **SMFBUF** parameter on the RMF procedure.

Example 20-9 RMF procedure that controls the SMF buffer of RMF SDS

```
//RMF PROC
//IEFPROC EXEC PGM=ERBMFMFC,REGION=32M,TIME=1440,
// PARM='SMFBUF(RECTYPE(30,70:79,104(1:12,20:31,40:53,60:64)))'
```

20.3.2 Interactions and dependencies

Software dependencies:

- AIX
  - AIX 5.3 or later
  - Packages:
    sysmgt.cimserver.pegasus
    sysmgt.cim.providers
    sysmgt.cim.smisproviders
- Linux
  - Red Hat Linux 5.3 or later
  - SUSE Linux 10.1 or later
  - Packages:
    - Top-pegasus or sblim-sfcb
    - Sblim-gather-provider (sblim = standard-based Linux instrumentation for manageability)
- Windows
  - Windows Server 2008 SP2
  - IBM Systems Director Platform Agent for Windows V6.3

Hardware dependencies:

- Requires hardware that can run any of those operating systems.

20.3.3 Installation

Installation support is included in z/OS 2.1 RMF. Be sure to check these two things, though:

- Make sure that SMF record type 104 is included in your `SMFPRMxx SYS(TYPE)` statement or that it is not excluded in your `SYS(NOTYPE)` statement.
- Include the SMF type 104 in the `PARM` statement of the RMF procedure.

20.4 RMF zHybrid Microsoft Windows support

A cross-platform performance monitoring solution was needed for all IBM System z Blade Center Extension (zBX)-related operating systems. Microsoft Windows was the only platform that was not supported by RMF XP, which was introduced in z/OS Version 1.13.

The following platforms were supported in that initial release:

- IBM AIX on System p
- IBM Linux on System x
- IBM Linux on System z

In z/OS 2.1, RMF XP can collect and display performance data from Windows running on IBM System x to make it available to view in GUI components, such as the RMF Data Portal and IBM System z Management Facility (z/OSMF) resource monitoring.

This information is available through SMF record type 104, subtypes 60 - 64. Capacity planning and accounting can now be performed for the Windows platform, and system programmers can analyze cross-platform performance data by using the same performance monitoring method.

The core component of RMF XP is the GPM4CIM started task. Similar to the existing Distributed Data Server for z/OS (DDS, GPMSERVE-started task), the GPM4CIM server can receive HTTP requests and send back responses in structured XML documents.

It runs in the z/OS UNIX environment, so at least one z/OS image is required to use the RMF XP component.

With z/OS 2.1, RMF XP uses the CIM client API to collect performance data from Windows systems in the same way that it does for AIX or Linux systems.

20.4.1 Invocation and use

To start the GPM4CIM server, RMF provides the GPM4CMI procedure as a member in the SYS1.PROCLIB. It runs under the z/OS UNIX environment through BPXBATCH and can be changed for what you need. Multiple instances can run in parallel, one task per platform.

The **cfg** parameter in the PARM statement points to the GPM4CIM configuration (`.cfg`) file. Because one instance of GPM4CIM is needed for each platform, no unique configuration file is used. That is why it comes with a variable name that defines which OS that it denotes, according to the following definitions:

A	AIX on System p
X	Linux on System x
Z	Linux on System z
W	Windows on system x

For example, the configuration file for Windows on System x is `gpm4`**`W`**`.cfg`.

In Figure 20-5, you can see the sample GPM4CIM proc for the Windows platform, which assigns W to the OS variable. Example 20-10 shows a sample `gpm4W.cfg` file.

```
//GPM4CIM PROC OS=W
//STEP1 EXEC PGM=BPXBATCH,TIME=NOLIMIT,REGION=0M,
// PARM='PGM /usr/lpp/gpm/bin/gpm4cim cfg=/etc/gpm/gpm4&OS..cfg'
//STDENV DD PATH='/etc/gpm/gpm4cim.env'
//STDOUT DD PATH='/var/gpm/logs/gpm4cim&OS..out',
// PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
// PATHMODE=(SIRUSR,SIWUSR,SIRGRP)
//STDERR DD PATH='/var/gpm/logs/gpm4cim&OS..trc',
// PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
// PATHMODE=(SIRUSR,SIWUSR,SIRGRP)
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
// PEND
```

Figure 20-5 GPM4CIM Windows proc example

Example 20-10 gpm4W.cfg example

```
INTERVAL(300)                   /* Monitoring interval (seconds)     */
WIN_COMPLEX(SAPPLEX)            /* User defined name of WIN complex  */
                                /* Linux images following here       */
WIN_IMAGE(SAP1.US.IBM.COM:5988)
WIN_IMAGE(SYS2.US.IBM.COM:5988)
```

The GPM4CIM parameters are supplied with the platform-specific configuration files `/etc/gpm/gpm4&OS.cfg` (`&OS` = A, X, Z, or W). You can use this to run one separate GPM4CIM instance per platform or multiple instances of it for the same platform. If you want to do that, create a separate copy of the configuration file for each instance.

The following parameters are specifically for GPM4CIM:

- `AIX_COMPLEX`
- `LNX_COMPLEX`
- `LNZ_COMPLEX`
- `WIN_COMPLEX`

Complex corresponds to the free selectable name for the system complex and these other four parameters:

- `AIX_IMAGE`
- `LNX_IMAGE`
- `LNZ_IMAGE`
- `WIN_IMAGE`

Image corresponds to the host name or IP address of one or more images that are running the specified operating system.

Note: All images within the same complex must run on the same operating system.

After you connect with your browser to GPM4CIM, the system complex appears as a top-level resource. Now you can navigate through the resource tree by clicking every individual resource that is marked as a hyperlink, as shown in Figure 20-6.

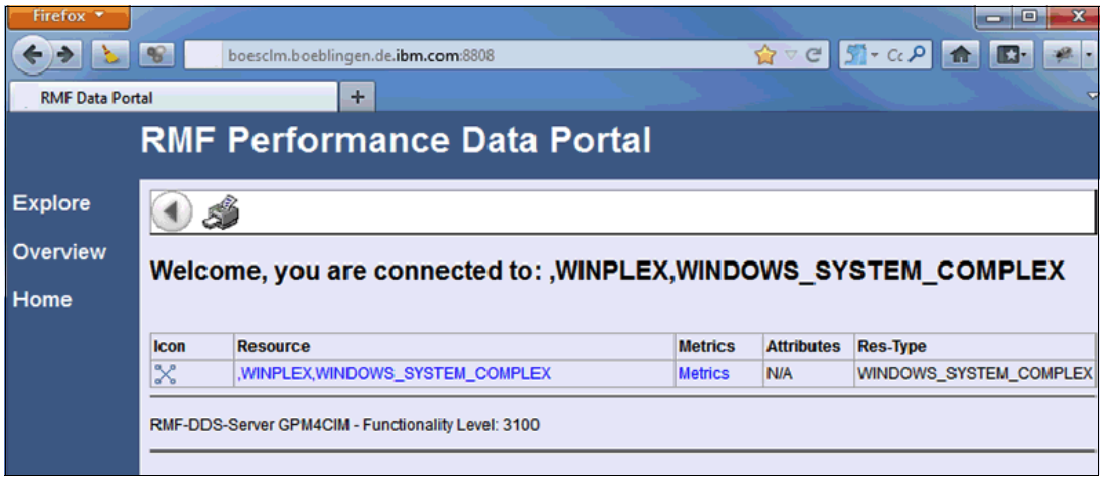


Figure 20-6 RMF Performance Data Portal navigation

When you click the Metrics column for a resource, the list of metrics that are applicable for this resource is displayed. After you select a certain metric, the metric's value or values for the current interval are displayed instantly. Figure 20-7 on page 270 shows the metrics section of the RMF Performance Data Portal.

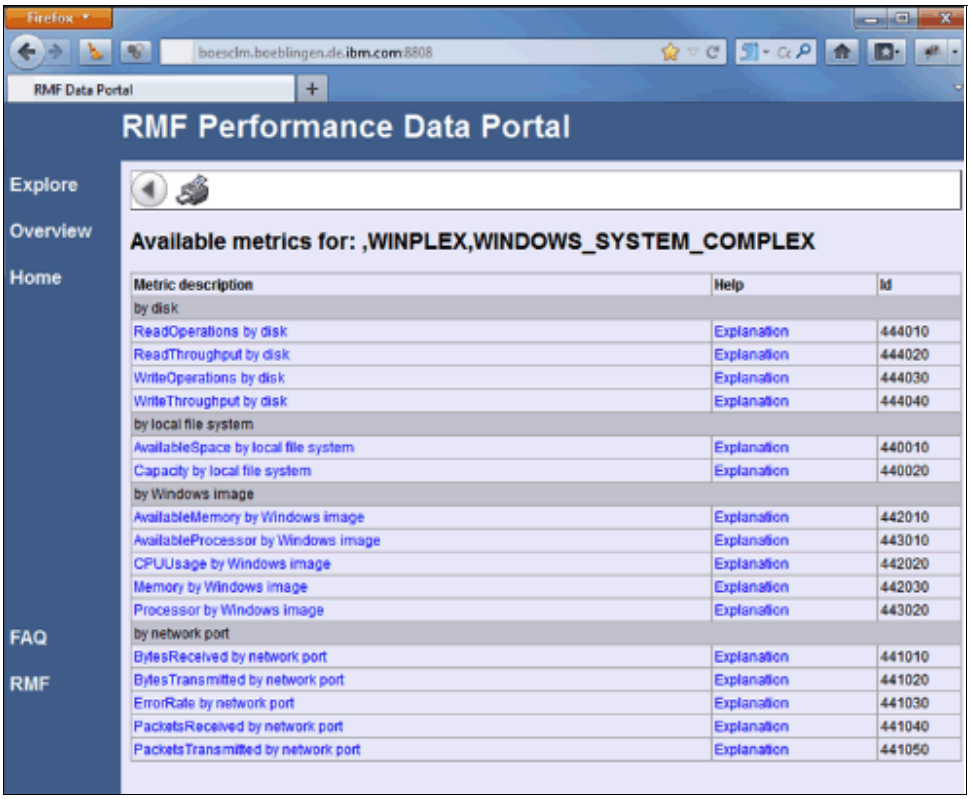


Figure 20-7 Windows metrics

There are two types of metrics:

- Single-valued, which consist of just one value in terms of a string
- List-valued, which consist of a list of name-value pairs

A certain value must be identical for the same resource, regardless of whether it is represented as single value or as part of a list. Figure 20-8 shows metric values, both single-valued and list-valued.

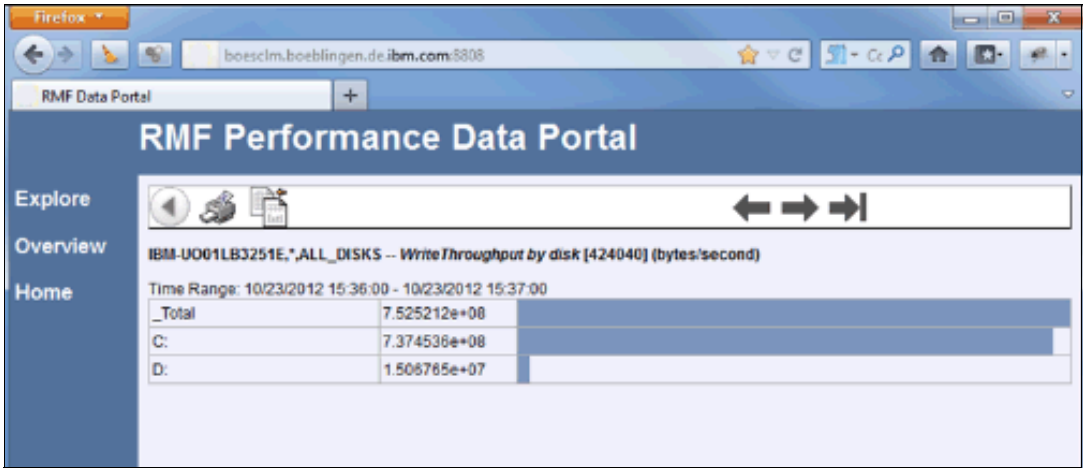


Figure 20-8 Metric values

All metrics are promoted to the system complex as top-level resources (see Figure 20-9). After a metric is selected from this context, all resource names are prefixed with the system name to ensure that a specific resource can be identified uniquely within the system complex.

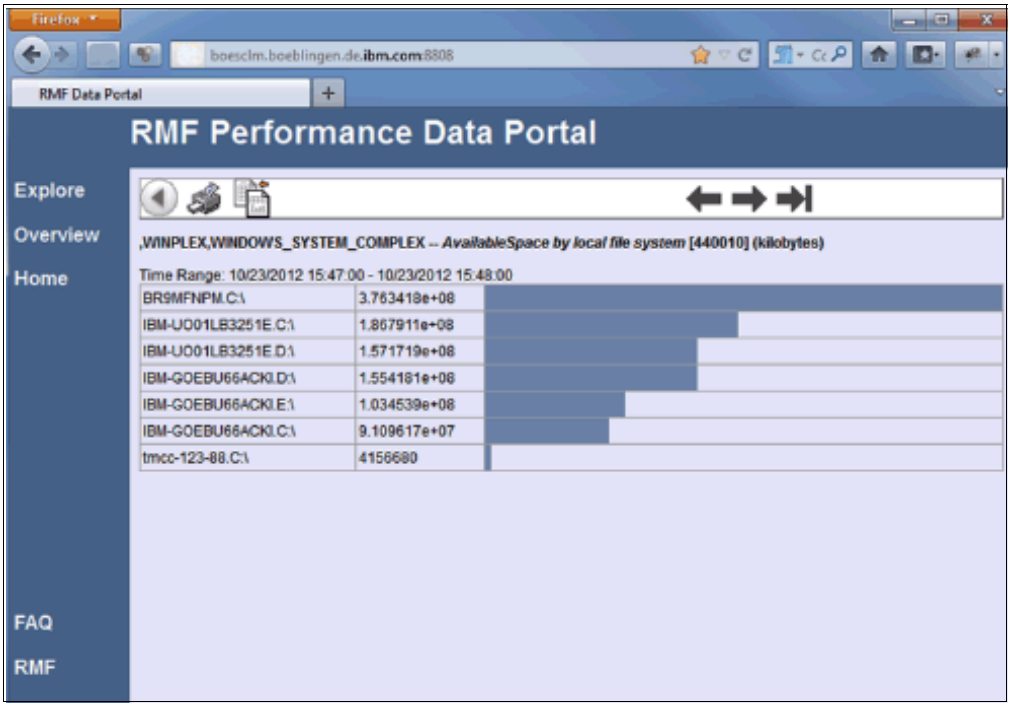


Figure 20-9 Metrics scope

20.4.2 Installation

All parts are installed in the file system directory `/usr/lpp/gpm` path:

- `/usr/pp/gpm/bin`
  - `gpm4cim`
  - `libpmccli.so`
  - `gpm4cim_setup.sh`
- `/usr/lpp/gpm/etc`
  - `gpm4cim.env`
  - `gpm4A.cfg`
  - `gpm4A.ini`
  - `gpm4X.cfg`
  - `gpm4X.ini`
  - `gpm4Z.cfg`
  - `gpm4Z.ini`
  - *`gpm4W.cfg`*
  - *`gpm4W.ini`*

20.5 Documentation and reference

For more information, see the following publications and web pages:

- *Resource Measurement Facility Programmer's Guide,* SC34-2667
  https://ibm.biz/BdFdnL
- *Resource Measurement Facility Reference Summary,* SX33-9034
  https://ibm.biz/BdFdnU
- *Resource Measurement Facility User's Guide,* SC34-2664
  https://ibm.biz/BdRWnT
- *z/OS RMF Report Analysis,* SC34-2665
  https://ibm.biz/BdFdnt
- *z/OS MVS System Management Facilities (SMF),* SA38-0667 (search by SA number)
  http://www.ibm.com/systems/z/os/zos/library/bkserv/v2r1pdf/

21

IBM Time Sharing Option Extensions and TSO/E REXX

This chapter describes the various enhancements, functions, and features that were introduced for Time Sharing Option Extensions (TSO/E) and TSO/E REXX with IBM z/OS Version 2 Release 1 (2.1):

- 21.1, “TSO/E logon failure messages” on page 274
- 21.2, “TSO/E REXX EXECIO enhancements” on page 275
- 21.3, “TSO/E REXX TRAPMSG function” on page 277
- 21.4, “64-bit support for the REXX STORAGE function” on page 280
- 21.5, “Enhanced TSO/E REXX LISTDSI function” on page 281
- 21.6, “Multiple TSO/E login support for JES3” on page 286
- 21.7, “TSO/E LOGOFF updates RACF TSO segment only on a change” on page 286
- 21.8, “Documentation and reference” on page 286

© Copyright IBM Corp. 2014. All rights reserved.

21.1 TSO/E logon failure messages

Although this is an MVS device allocation change, it is presented here as it affects allocation messages that are displayed during TSO logon processing.

MVS Device Allocation is used in many environments, such as batch jobs, z/OS subsystems, and during TSO logon processing. Allocation messages that are issued are sent to the JOBLOG by default. TSO users can use the System Display and Search Facility (SDSF) for example) to view the JCL messages that are created during TSO logon processing. However, when TSO logon fails because of an allocation error, the only indication that is displayed on the TSO user's terminal is that the logon attempt failed.

To see an error that occurred during logon processing, the JOBLOG messages had to be examined from another login session (for example, by a system programmer with access to the user's JOBLOG) to determine the error.

With z/OS 2.1, device allocation messages that are issued during TSO logon processing are sent to both the JOBLOG and the TSO console. This enables you to see logon failure messages unassisted, which potentially allows you to correct the problem without having to access the JOBLOG.

21.1.1 Invocation and use

All TSO logons automatically receive allocation failure messages. No invocation is necessary.

Figure 21-1 shows an allocation failure message. In this example, the KARAN1.TEST.DATA data set is not found.

```
ICH70001I KARAN1   LAST ACCESS AT 14:10:51 ON THURSDAY, JULY 18, 2013
KARAN1 LOGON IN PROGRESS AT 14:18:53 ON JULY 18, 2013
IEFA107I KARAN1 IKJTEST IKJTEST TESTDD - data set KARAN1.TEST.DATA NOT FOUND
LOGON FAILED ALLOCATION UNSUCCESSFUL
KARAN1 LOGGED OFF TSO AT 14:18:53 ON JULY 18, 2013
IKJ56400A ENTER LOGON OR LOGOFF-
```

Figure 21-1 Allocation failure message

Note: The IEFA107I message replaced IEF212Is:

```
IEF212I jobname (procstep) stepname ddname[+ xxx] - data set NOT FOUND
IEFA107I jobname procstep stepname ddname relpos - data set dsname NOT FOUND
```

Figure 21-2 on page 275 is an example of an allocation failure due to an offline user catalog.

```
IKJ56455I SPFUSER LOGON IN PROGRESS AT 14:20:55 ON MAY 3, 2012
IEFA102I UNABLE TO LOCATE SDB.DS01
FROM PRIVATE CATALOG IBMUSER.SDB.UCAT
IEF361I SPFUSER ISPFPROC SDBISPF1 - UNABLE TO ALLOCATE / OPEN PRIVATE CATALOG
OR ALLOCATE CVOL
IKJ56457I LOGON FAILED ALLOCATION UNSUCCESSFUL
IKJ56470I SPFUSER LOGGED OFF TSO AT 14:21:01 ON MAY 3, 2012
IKJ56400A ENTER LOGON OR LOGOFF-
```

Figure 21-2 Allocation failure due to offline catalog

The following types of messages are not sent to the TSO console:

- Allocation success messages (IEF236I, IEF237I, IGD101I, IGD103I, IGD104I, IEF285I, and so on)
- JCL processing messages (IEFC001I)
- JCL syntax error message: (IEFC605I UNIDENTIFIED OPERATION FIELD or IEFC621I EXPECTED CONTINUATION NOT RECEIVED)

21.2 TSO/E REXX EXECIO enhancements

Over the years, IBM clients have submitted many requests for the capability to use **EXECIO** to handle I/O to data sets that contain records with Variable Spanned (VS, VBS) RECFM and data sets that have undefined (U) RECFM. This includes the ability to handle spanned files that are generated by SMF or to read load library type undefined files.

In this release, EXECIO support is extended, so it is possible to read or write RECFM=VS, VBS, or U type data sets under REXX.

Note: RECFM=VS/VBS files do not support update mode (DISKRU).

21.2.1 Invocation and use

There is no change to the EXECIO syntax. However, there are enhanced capabilities.

Example 21-1 uses EXECIO to read records from an input RECFM=VS file and write them to a file that has RECFM=VBS (assuming LRECL input that is less than or equal to 240).

Example 21-1 EXECIO use for VS and VBS RECFM data sets

```
/* REXX */
"ALLOC FI(INVS) DA('userid.test.vs') SHR REUSE"
ALLOCRC = RC
"ALLOC FI(OUTVBS) DA('userid.test.newvbs') SPACE(1) TRACKS " ,
   " LRECL(240) BLKSIZE(80) RECFM(V B S) DSORG(PS) REUSE"
ALLOCRC = MAX(RC,ALLOCRC)
execio_rc = 0                          /* Initialize             */
error = 0                              /* Initialize             */
IF ALLOCRC = 0 THEN
  do
    /*************************************************************/
    /* When spanned records are read, each logical record is the  */
```

```
    /* collection of all spanned segments of that record on DASD. */
    /*************************************************************/
    "execio * DISKR INVS (STEM inrec. FINIS"  /* Read all records */
    if rc /= 0 then
      error = 1                          /* Read Error occurred    */
  end
ELSE
  do
    say 'File allocation error ...'
    error = 1                           /* Error occurred          */
  end
IF error = 0 then                    /* If no d is ok              */
  DO
    "execio "inrec.0" DISKW OUTVBS (STEM inrec. FINIS"  /* Write all
                                     records read to the file */
    if rc=0 then
      do
        say 'Output to VBS file completed successfully'
        say 'Number of records copied ===> ' inrec.0
      end
    else
      do
        say 'Error writing to VBS file '
        error = 1                       /* Error occurred          */
      end
  END
```

Example 21-2 uses EXECIO to read a member of the RECFM=U data set and change the first occurrence of TSOREXX within each record to TSOEREXX before rewriting the record. If a record is not changed, it does not need to be rewritten.

Example 21-2 Use of EXECIO with a RECFM=U member

```
/* REXX */
/* Alloc my Load Lib data set having RECFM=U BLKSIZE=32000 LRECL=0 */
"ALLOC FI(INOUTDD) DA('apar2.my.load(mymem)') SHR REUSE"
readcnt = 0                          /* Initialize rec read cntr    */
updtcnt = 0                          /* Initialize rec update cntr  */
error = 0                            /* Initialize flag             */
EoF = 0                              /* Initialize flag             */
do while (EoF=0 & error=0)           /* Loop while more recs/no err */
  "execio 1 DISKRU INOUTDD (STEM inrec." /* Read a rec for update  */
  if rc = 0 then                     /* If read ok                  */
    do  /* Replace 1st occurrence of 'TSOREXX' in record by 'TSOEREXX'
                                   and write it back                */
      readcnt = readcnt + 1          /* Records read                  */
      z = POS('TSOREXX ',inrec.1,1)    /* Find target within rec   */
      if z /= 0 then                 /* If found, replace it         */
         do
           inrec.1 = SUBSTR(inrec.1,1,z-1)||'TSOEREXX'|| ,
                  SUBSTR(inrec.1,z+LENGTH('TSOEREXX'))  /*Replace it*/
           "execio 1 DISKW INOUTDD (STEM inrec." /* Rewrite the update
                                       made to the last record read*/
       if rc > 0 then                /* If error                     */
             error=1                 /* Indicate error               */
           else
```

```
            updtcnt = updtcnt + 1   /* Incr update count          */
          end
        else                        /* Else nothing changed, nothing
                                       to rewrite                */
          NOP                       /* Continue with next record  */
      end
    else                            /* Else non-0 RC              */
      if rc=2 then                  /* if end-of-file             */
        EoF=1                       /* Indicate end-of-file       */
      else
        error=1                     /* Else read error            */
  end                               /* End do while               */
  "execio 0 DISKW INOUTDD (FINIS"   /* Close the file             */
  if error = 1 then
    say '*** Error occurred while updating file '
  else
    say updtcnt' of 'readcnt' records were changed'
  "FREE FI(INOUTDD)"
  exit 0
```

21.3 TSO/E REXX TRAPMSG function

TRAPMSG is a TSO/E REXX function that is used in conjunction with OUTTRAP to permit REXX to trap REXX messages (IRX messages) in some instances.

Before this, IRX messages could not be trapped. Now, you can use `TRAPMSG('on')` to tell REXX to treat REXX msg output the same way as any other output for purposes of trapping.

REXX messages that are issued by nested execs and by host commands that are invoked by REXX (**`EXECIO`**) can now be trapped into an OUTTRAP variable, rather than always being written to the panel. CLIST error messages from CLISTs started by REXX can also be trapped.

21.3.1 Invocation and use

The TRAPMSG function uses the following syntax:

`TRAPMSG()`	Returns current setting.
`TRAPMSG('ON')`	Enables output trapping for IRX messages.
`TRAPMSG(‘'OFF')`	Disables output trapping for IRX messages. The default is OFF.

Example 21-3 is a REXX program exec that starts **`EXECIO`** without allocating the `indd` file. EXECIO returns RC=20 and an error message. By trapping the message with OUTTRAP, the exec can decide what to do with the error. This same technique can be used to trap the IRX0250E message if EXECIO was to take an ABEND, such as a space B37 ABEND.

Example 21-3 TRAPMSG() example

```
/* REXX */
MSGTRAP_STAT = TRAPMSG('ON')      /* SAVE CURRENT STATUS AND SET
                   TRAPMSG ON TO ALLOW REXX MSGS TO BE TRAPPED  */
OUTTRAP_STAT = OUTTRAP('LINE.')   /* ENABLE OUTTRAP              */
/*************************************************************/
```

```
/* INVOKE TSO HOST CMD, EXECIO, AND TRAP ANY ERROR MSGS ISSUED  */
/*************************************************************/
"EXECIO 1 DISKR INDD (STEM REC. FINIS"

IF RC = 20 THEN                  /* IF EXECIO ERROR  OCCURRED   */
  DO I=1 TO LINE.0
    SAY '==> ' LINE.I            /* WRITE ANY ERROR MSGS        */
  END
OUTTRAP_STAT = OUTTRAP('OFF')    /* DISABLE OUTTRAP             */
MSGTRAP_STAT = TRAPMSG('OFF')    /* TURN IT OFF                 */
EXIT 0
```

The output from the TRAPMSG() example is shown in Figure 21-3.

```
==> The input or output file INDD is not allocated. It cannot be opened for
I/O.
==> EXECIO error while trying to GET or PUT a record.
```

Figure 21-3 Output from Example 21-3 on page 277

Example 21-4 contains two REXX execs. A REXX exec (exec1) turns on OUTTRAP and TRAPMSG and invokes a second REXX exec (exec2) as a command. The second REXX exec gets an IRX040E message due to an invalid function call. Exec1 is able to trap the message that was issued from exec2. The output is shown in Figure 21-4.

If exec1 had made the bad function call, it could not trap the error message because a function message is considered to be at the same level as the exec. This is similar to the fact that an exec can use OUTTRAP to trap SAY statements from an exec that it invokes, but it cannot trap its own SAY output.

Example 21-4 REXX exec1 and exec2

```
/* REXX - EXEC1 */
TRAPIT = OUTTRAP('LINE.')
TRAPMSG_STAT = TRAPMSG('ON')
‘%EXEC2’
DO I=1 TO LINE.0   /* DISPLAY ANY OUTPUT TRAPPED FROM EXEC2 */
  SAY '==> ' LINE.I
END
TRAPIT = OUTTRAP('OFF')
TRAPMSG_STAT = TRAPMSG('OFF')
EXIT 0
-------------------------------------------------------------
/* REXX - EXEC2 */
SAY 'IN EXEC2 ...'
TIME = TIME('P')    /* INVALID TIME OPERAND, GET MSG IRX0040I*/
RETURN TIME
```

```
==>  IN EXEC2 ...
==>       3 +++ TIME = TIME('P')    /* INVALID TIME OPERAND, GET MSG IRX0040I*/
==>  IRX0040I Error running EXEC2, line 3: Incorrect call to routine
```

Figure 21-4 Display of trapped messages from exec2

If a function or subroutine is invoked by the CALL keyword and the function or subroutine ends with a non-zero return code, although the output is trapped, additional code is required to deal with the function or subroutine error. When an exec is invoked as a command, as shown in Example 21-4 on page 278, a non-zero return code does not terminate the invoking exec (exec1).

In the following example, a CALL statement is used from exec1 to invoke exec2. In this case, a SYNTAX error is raised with the CALL exec2 statement. Exec1 would end because the function or subroutine call would return a non-zero return code (in this case, exec2) before the display of the trapped output from exec2. In Example 21-5, logic is added in exec1 to handle the SYNTAX error to demonstrate that the output from exec2 was trapped, including the IRX messages. Figure 21-5 shows the output.

Example 21-5 REXX exec1 and exec2 with CALL

```
/* REXX - exec1 */
SIGNAL ON SYNTAX                    /* Trap syntax errors   */
trapit = OUTTRAP('line.')
trapmsg_stat = TRAPMSG('ON')
continu = 'continu1'
call exec2
continu1:
do i=1 to line.0   /* Display any output trapped from exec2 */
  if i=1 then      say '---------- start of trapped output ----------'
  say '==> ' line.i
  if i=line.0 then say '---------- end   of trapped output ----------'
end
trapit = OUTTRAP('OFF')
trapmsg_stat = TRAPMSG('OFF')
exit 0

SYNTAX:
 SIGNAL ON SYNTAX
 Interpret SIGNAL continu
-------------------------------------------------------------------------------
/* REXX - exec2 */
say 'In exec2 ...'
time = TIME('P')    /* Invalid time operand, get msg IRX0040I*/
return time
```

```
---------- START OF TRAPPED OUTPUT ----------
==>  IN EXEC2 ...
==>       3 +++ TIME = TIME('P')    /* INVALID TIME OPERAND, GET MSG IRX0040I*/
==>  IRX0040I Error running REX2, line 3: Incorrect call to routine
---------- END   OF TRAPPED OUTPUT ----------
```

Figure 21-5 Display of trapped message from exec2 with CALL

21.4 64-bit support for the REXX STORAGE function

z/OS can address 64-bit storage, which provides vastly expanded addressable areas. Until this release, REXX could not read or write to these areas.

With z/OS 2.1, the REXX STORAGE function can view or change storage within 64-bit addressable areas. The STORAGE function is extended to handle 64-bit addresses in addition to current 24- and 31-bit addresses.

21.4.1 Invocation and use

The REXX STORAGE function now supports 64-bit addresses that are represented by 9 - 17 characters, consisting of 8 - 16 hexadecimal characters and an optional underscore (_) character that separates the high-order half and low-order half of the 64-bit address.

Example 21-6 shows the use of a valid 64-bit address with the STORAGE function. In this case, 60 bytes are read from 64-bit address IEF_80000010.

Example 21-6 Using a valid 64-bit address with the STORAGE function

```
storet = STORAGE(000001EF_80000010,60)
```

Example 21-7 illustrates valid and invalid 64-bit addresses.

Example 21-7 Valid and invalid 64-bit address for the REXX STORAGE function

```
Hex Address passed   Binary Address
  to STORAGE           used by STORAGE        Comment
  ================     ==================     ==================
         _00000010     '0000000000000010'x    - Valid 64-bit addr.
                                                (Padded to left with
                                                0's to 64-bits.)
                                                Addresses same area as
                                                31-bit '00000010'x addr.
        0_00000010     '0000000000000010'x    - Valid 64-bit addr.
                                                Addresses same area
                                                as _00000010.
        0_80000010     '0000000080000010'x    - Valid 64-bit addr.
                                                Addr is 2GB beyond
                                                the 0_00000010 addr.
        000001EF10     '000000000001EF10'x    - Valid 64-bit addr.
      1EF_80000010     '000001EF80000010'x    - Valid 64-bit addr.
       1EF80000010     '000001EF80000010'x    - Valid 64-bit addr
                                                without "_" separator.
 000001EF_80000010     '000001EF80000000'x    - Valid 64-bit addr.
 000001EF_10            Invalid Addr          - Right half of 64-bit
                                                addr <8 chars.
 00000001EF_000010      Invalid Addr          - Left half of addr >8
                                                chars, right half <8
                                                chars.
0000001EF_80000010      Invalid Addr          - More that 16 hex chars
                                                Also, left half more
                                                than 8 chars.
 00001EF8000001000      Invalid Addr          - More that 16 hex chars
```

21.5 Enhanced TSO/E REXX LISTDSI function

LISTDSI is a TSO/E external function that retrieves information about a data set's allocation, protection, and directory and stores this information in REXX variables.

There are variables and improvements with LISTDSI in this release:

- LISTDSI has arguments:

  `noracf` Indicates whether a check for RACF authority has been done or not. If not, the data set will not be opened by LISTDSI, for example, to read directory information.

  `multivol` Indicates whether size calculations include all volumes when a data set includes more than one. The `SYSNUMVOLS` and `SYSVOLUMES` variables are not affected by this setting.

- LISTDSI now provides information about all volumes of a multi-volume data set, not just the first volume. The following variables are added to support this change:

  `SYSNUMVOLS` The number of volumes for a multi-volume data set.

  `SYSVOLUMES` List of volumes, up to `SYSNUMVOLS`, delimited by a space. Up to 412 characters. The first six characters always match SYSVOLUME.

- LISTDSI variables have been added for PDSE data sets:

  `SYSALLOCPAGES` Indicates the number of pages that are allocated to a PDSE.

  `SYSUSEDPERCENT` Indicates the percentage of pages used from the pages allocated for a PDSE. A 0 - 100 number, rounded down to the nearest integer value.

- The following LISTDSI variables have been added for EAV data sets:

  `SYSCREATETIME` Indicates the time that a data set was created in the hh:mm:ss format, where hh is hours since midnight, mm is minutes since midnight, and ss is seconds since midnight. This variable is set only for EAV data sets and can be used with the `SYSCREATE` variable to determine the date and time when a data set was created.

  `SYSCREATEJOB` For EAV data sets, if available, indicates the name of the job that created the data set.

  `SYSCREATESTEP` For EAV data sets, if available, indicates the name of the job step that created the data set.

21.5.1 Invocation and use

This section provides use details about the LISTDSI updates with this release.

LISTDSI provides information about multi-volume data sets

LISTDSI now provides information about all volumes of a multi-volume data set, not just the first. The number of volumes is returned in variable SYSNUMVOLS and the list of volumes in SYSVOLUMES. The existing SYSVOLUME returns the name of the first volume,

The sample exec in Example 21-8 on page 282 calls LISTDSI for a multi-volume PDSE.

Example 21-8 Sample LISTDSI for multi-volume PDSE

```
/*REXX*/
DSN = "'KARAN.TEST.PDSE'"
DSINFO = LISTDSI(DSN DIRECTORY SMSINFO)
SAY 'SYSDSNAME    =  ' SYSDSNAME
SAY 'SYSVOLUME    =  ' SYSVOLUME
SAY 'SYSNUMVOLS   =  ' SYSNUMVOLS
SAY 'SYSVOLUMES   =  ' SYSVOLUMES
```

Figure 21-6 shows the results.

```
SYSDSNAME    =   KARAN.TEST.PDSE
SYSVOLUME    =   BH5ST4
SYSNUMVOLS   =   2
SYSVOLUMES   =   BH5ST4 BH5ST5
```

Figure 21-6 Sample LISTDSI for multi-volume PDSE output

LISTDSI multivol or nomultivol arguments

The **`multivol`** or **`nomultivol`** arguments indicate whether size calculations should include all volumes when a data set's allocation includes more than one volume.

If the **`multivol`** argument is passed to the LISTDSI function, variables that include size calculations include the statistics for all volumes rather than just the first one. If `multivol` is not specified on the function (default) or `nomultivol` is specified, the calculations are done only for the first volume.

In Example 21-9, a data set that is allocated uses LISTDSI to query information about a sequential data set on two volumes without the multivol argument. In this case, the calculations that are returned in variables are for only the first volume, as shown in Figure 21-7 on page 283. Regardless of whether `multivol` or `nomultivol` is specified, the SYSNUMVOLS and SYSVOLUMES variables are populated.

Example 21-9 LISTDSI for multi-volume data set with NOMULTIVOL specified

```
/*REXX*/
DSN = "'KARAN.TEST.PS'"
DSIARGS = "NOMULTIVOL"
DSINFO = LISTDSI(DSN DSIARGS)
SAY 'SYSDSNAME    =  ' SYSDSNAME
SAY 'SYSBLKSIZE   =  ' SYSBLKSIZE
SAY 'SYSUNITS     =  ' SYSUNITS
SAY 'SYSALLOC     =  ' SYSALLOC
SAY 'SYSUSED      =  ' SYSUSED
SAY 'SYSPRIMARY   =  ' SYSPRIMARY
SAY 'SYSSECONDS   =  ' SYSSECONDS
SAY 'SYSVOLUME    =  ' SYSVOLUME
SAY 'SYSEXTENTS   =  ' SYSEXTENTS
SAY 'SYSNUMVOLS   =  ' SYSNUMVOLS
SAY 'SYSVOLUMES   =  ' SYSVOLUMES
```

```
SYSDSNAME    =   KARAN.TEST.PS
SYSBLKSIZE   =   80
SYSUNITS     =   BLOCK
SYSALLOC     =   1248
SYSUSED      =   1248
SYSPRIMARY   =   78
SYSSECONDS   =   1
SYSVOLUME    =   BH5ST3
SYSEXTENTS   =   16
SYSNUMVOLS   =   2
SYSVOLUMES   =   BH5ST3 BH5ST5
```

Figure 21-7 LISTDSI output for multi-volume data set with NOMULTIVOL specified

Example 21-10 uses LISTDSI with `multivol` specified, so calculations are returned in variables for all volumes on which the data set has an allocation. The output is shown in Figure 21-8.

Example 21-10 LISTDSI for multi-volume data set with MULTIVOL specified

```
/*REXX*/
DSN = "'KARAN.TEST.PS'"
DSIARGS = "MULTIVOL"
DSINFO = LISTDSI(DSN DSIARGS)
SAY 'SYSDSNAME    =  ' SYSDSNAME
SAY 'SYSBLKSIZE   =  ' SYSBLKSIZE
SAY 'SYSUNITS     =  ' SYSUNITS
SAY 'SYSALLOC     =  ' SYSALLOC
SAY 'SYSUSED      =  ' SYSUSED
SAY 'SYSPRIMARY   =  ' SYSPRIMARY
SAY 'SYSSECONDS   =  ' SYSSECONDS
SAY 'SYSVOLUME    =  ' SYSVOLUME
SAY 'SYSEXTENTS   =  ' SYSEXTENTS
SAY 'SYSNUMVOLS   =  ' SYSNUMVOLS
SAY 'SYSVOLUMES   =  ' SYSVOLUMES
```

As shown in Figure 21-8, the calculations for the variables `SYSALLOC`, `SYSUSED`, and `SYSEXTENTS` include all the volumes for the data set.

```
SYSDSNAME    =   KARAN.TEST.PS
SYSBLKSIZE   =   80
SYSUNITS     =   BLOCK
SYSALLOC     =   2496
SYSUSED      =   2496
SYSPRIMARY   =   78
SYSSECONDS   =   1
SYSVOLUME    =   BH5ST3
SYSEXTENTS   =   32
SYSNUMVOLS   =   2
SYSVOLUMES   =   BH5ST3 BH5ST5
```

Figure 21-8 LISTDSI output for multi-volume data set with MULTIVOL specified

LISTDSI racf or noracf argument

The racf or noracf arguments are provided on the LISTDSI function call.

Specifying noracf means that LISTDSI will not determine the RACF status for the specified data set. This implies that LISTDSI will not attempt to open the data set to gather additional information, even if an open is necessary based on another keyword. For example, for a PDS, if DIRECTORY is specified, LISTDSI would open the data set to get directory information, but it will not if noracf is specified.

Specify noracf if you do not want LISTDSI to query RACF about whether a data set is protected. (Default is RACF.)

Example 21-11 is a REXX exec that uses the LISTDSI function with the noracf argument. As can be seen in the output that is shown in Figure 21-9, the SYSRACFA variable is blank. Additionally, although the directory argument was specified, because the data set is not opened no directory information is returned.

Example 21-11 LISTDIS with noracf argument

```
/*REXX*/
DSN = "'KARAN.CNTL'"
DSIARGS = "DIRECTORY NORACF"
DSINFO = LISTDSI(DSN DSIARGS)
SAY 'SYSDSNAME    =  ' SYSDSNAME
SAY 'SYSDSORG     =  ' SYSDSORG
SAY 'SYSRACFA     =  ' SYSRACFA
SAY 'SYSADIRBLK   =  ' SYSADIRBLK
SAY 'SYSUDIRBLK   =  ' SYSUDIRBLK
SAY 'SYSMEMBERS   =  ' SYSMEMBERS
SAY 'SYSREASON    =  ' SYSREASON
SAY 'SYSMSGLV1    =  ' SYSMSGLVL1
SAY 'SYSMSGLV2    =  ' SYSMSGLVL2
```

```
SYSDSNAME    =   KARAN.CNTL
SYSDSORG     =   PO
SYSRACFA     =
SYSADIRBLK   =   ??????
SYSUDIRBLK   =   ??????
SYSMEMBERS   =   ??????
```

Figure 21-9 LISTDSI output with noracf argument

LISTDSI PDSE enhancements

For PDSE data sets, the SYSALLOCPAGES variable returns the number of pages that are allocated to a PDSE, and the SYSUSEDPERCENT variable returns the percentage of pages used. These variables along with the existing SYSUSEDPAGES are always returned for a PDSE data set.

Example 21-12 on page 285 shows a REXX exec that displays the SYSALLOCPAGES and SYSUSEDPERCENT for a PDSE data set. The output is shown in Figure 21-10 on page 285.

Example 21-12 LISTDSI to list SYSALLOCPAGES and SYSUSEDPERCENT

```
/*REXX*/
DSN = "'SYS1.SIEALNKE'"
DSIARGS = "DIRECTORY SMSINFO"
DSINFO = LISTDSI(DSN DSIARGS)
SAY 'SYSDSNAME    =  ' SYSDSNAME
SAY 'SYSDSORG     =  ' SYSDSORG
SAY 'SYSDSSMS     =  ' SYSDSSMS
SAY 'SYSALLOCPAGES = ' SYSALLOCPAGES
SAY 'SYSUSEDPAGES =  ' SYSUSEDPAGES
SAY 'SYSUSEDPERCENT = ' SYSUSEDPERCENT
```

```
SYSDSNAME    =   SYS1.SIEALNKE
SYSDSORG     =   PO
SYSDSSMS     =   PROGRAM_LIBRARY
SYSALLOCPAGES =  39732
SYSUSEDPAGES =   35818
SYSUSEDPERCENT =  90
```

Figure 21-10 LISTDSI output showing SYSALLOCPAGES and SYSUSEDPERCENT

LISTDSI extended address volume (EAV) enhancements

If it is available, LISTDSI captures additional information about a data set that was created on an EAV volume. For a data set created on an EAV volume (EATTR=OPT), LISTDSI returns data set creation time in the `SYSCREATETIME` variable, the creating job name in the `SYSCREATEJOB` variable, and the creating step name in the `SYSCREATESTEP` variable.

Example 21-13 is a sample REXX exec that uses LISTDSI to display these variables for a physical sequential data set that is allocated with EATTR=OPT on a EAV volume. The output is shown in Figure 21-11.

Example 21-13 LISTDSI with EAV-related variables

```
/*REXX*/
DSN = "'KARAN.TEST.EAVPS'"
DSINFO = LISTDSI(DSN)
SAY 'SYSDSNAME    =  ' SYSDSNAME
SAY 'SYSEATTR     =  ' SYSEATTR
SAY 'SYSEADSCB    =  ' SYSEADSCB
SAY 'SYSCREATE    =  ' SYSCREATE
SAY 'SYSCREATETIME = ' SYSCREATETIME
SAY 'SYSCREATEJOB  = ' SYSCREATEJOB
SAY 'SYSCREATESTEP = ' SYSCREATESTEP
```

```
SYSDSNAME    =   KARAN.TEST.EAVPS
SYSEATTR     =   OPT
SYSEADSCB    =   YES
SYSCREATE    =   2013/204
SYSCREATETIME =  11:45:30
SYSCREATEJOB  =  KARANA
SYSCREATESTEP =  DEFEAV
```

Figure 21-11 LISTDSI output

21.6 Multiple TSO/E login support for JES3

In previous releases, a TSO user could not log on to more than one system simultaneously when JES3 was used. Now, it is possible to the same TSO user to be logged on to multiple systems with the same user ID. It provides the same function that has been available for JES2.

When a TSO user is logged on to multiple systems, the following notification rules apply:

1. When a system was specified explicitly, notification messages are sent to that system.
2. When no explicit specification is made, the notification goes to the system from which the TSO user submitted the job.
3. When no system specification is found, a search is conducted, starting with the first system that is defined through MAINPROC. The first occurrence of the TSO user ID triggers the notification to be sent to that system.
4. When no user is logged on, the notification is sent to the global.

Attention: Multiple TSO logon support is available only on systems with JES3 V2.1 or later versions.

21.7 TSO/E LOGOFF updates RACF TSO segment only on a change

During LOGOFF processing previously, TSO/E always rewrote the TSO/E information to the RACF database TSO segment. With this change, during LOGOFF processing, TSO/E writes only the TSO segment if it has changed during the TSO/E session where the user is issuing the LOGOFF. This avoids any unnecessary writes and exclusive enqueues on the RACF database.

21.8 Documentation and reference

For more information, see the following IBM publications, which are available to download as PDF files from the z/OS V2R1 Elements and Features Library (search by SA number):

http://www.ibm.com/systems/z/os/zos/library/bkserv/v2r1pdf/

- *z/OS V2R1.0 JES3 Messages*, SA32-1007
- *z/OS V2R1.0 TSO/E Customization*, SA32-0976
- *z/OS V2R1.0 TSO/E REXX Reference*, SA32-0972

22

Workload manager

The Workload Manager is an IBM z/OS component that handles the installation-declared transaction goals, as stated in a service definition policy. This chapter describes the Workload Manager enhancements that were introduced with z/OS Version 2 Release 1 (2.1):

- 22.1, “Extend IWMCLSFY macro interface to AMODE64” on page 288
- 22.2, “Performance blocks (PBs) above the 2 GB bar” on page 288
- 22.3, “Fast path classification” on page 289
- 22.4, “Workload Manager I/O priority group” on page 290
- 22.5, “Improved accuracy for the IWMEQTME service” on page 292
- 22.6, “More Workload Manager application environments” on page 293
- 22.7, “Documentation and reference” on page 294

© Copyright IBM Corp. 2014. All rights reserved.

22.1 Extend IWMCLSFY macro interface to AMODE64

With z/OS 2.1, the Workload Manager includes a 64-bit classify work request service API (IWM4CLSY), which allows a transaction manager to call this service in AMODE 31 or AMODE 64. This service also supports parameter values that are located above the 2 GB bar and can be used to replace the former IWMCLSFY macro.

22.1.1 Invocation and use

This enhancement results in performance gains because there are fewer CP cycles used by the transaction manager into AMODE31 switching from AMODE64 transaction managers. This improvement also saves virtual storage below the 2 GB bar.

IWM4CLSY in z/OS 2.1 also includes five classification attributes to improve the association of DB2 parallel query and DDF transactions with service classes at classification rules:

- Client accounting information, not the PM address space accounting information
- Client IP address, 39 characters that contain the source client IPv6 address (for example: 2001:0DB8:0000:0000:0008:0800:200C:417A)
- Client transaction name
- Client user ID
- Client workstation name, which contains the value of the client workstation name or host name from the client information that is specified for the connection

These classification attributes are associated with the real client, the originator of the query, as provided by the client and specified for the connection. These attributes are different in a DDF transaction, for example, from the equivalent ones in the DDF address space.

In z/OS 2.1, maximum length for these two classification attributes is increased from 8 to 128 characters:

- Package, which contains the package name for a set of associated SQL statements
- Procedure name, which contains the IBM DB2 stored SQL procedure name that is associated with the transaction

22.2 Performance blocks (PBs) above the 2 GB bar

With z/OS 2.1, Workload Manager introduces 12 AMODE64 execution delay monitoring services. Four are changed to support performance blocks (PBs) above 2 GB.

22.2.1 Invocation and use

In versions before z/OS 2.1, the PBs are allocated in common (sometimes private) storage below the 2 GB bar, which reduces the available area below the 2 GB bar for application programs. Also, several execution delay monitoring services are available only for AMODE31 transaction manager callers.

With z/OS 2.1, Workload Manager includes 12 AMODE64 execution delay monitoring services (using the IWM4MCRE macro), and four previous ones are changed to support the PBs above the 2 GB bar:

IWM4ECRE	Create an enclave.
WM4MABN	Record an abnormal event.
IWM4MCHS	Change the state of a work request.
IWM4MCRE	Create a monitoring environment.
IWM4MDEL	Delete a monitoring environment.
IWM4MXTR	Delay a monitoring extract service.
IWM4MINI	Monitor an environment extract service.
IWM4MNTF	Notify of work execution.
IWM4MRLT	Relate monitoring environment.
IWM4MSTO	Stop a work unit.
IWM4MSTR	Indicates the start of a work unit.

The AMODE64 Workload Manager execution delay monitoring services are identical to the AMODE31 services, except that they can also run in AMODE 64.

In z/OS 2.1, to control the location of the PBs (above or below the 2 GB bar), the `MONTKN64` keyword of the IWM4MCRE is introduced. MONTKN64 is mutually exclusive with the previous `MONTKN` keyword. However, for flexibility purposes, it is still possible to create PBs below the 2 GB bar with the IWM4MCRE service, even when the `MONTKN64` keyword is used.

The first users of having PBs above the 2 GB bar are IBM DB2 parallel query and distributed data facility (DDF).

22.2.2 Migration and coexistence considerations

Take special care when execution delay monitoring services are being used by a transaction manager and a database manager in relation to the location of the PB (above or below the 2 GB bar).

22.3 Fast path classification

Before z/OS 2.1, in a workload with massive IBM WebSphere Application Server transactions, holding the cross memory local lock (CML) at the Workload Manager address space can cause severe contention.

z/OS 2.1 implements the fast path classification. That is, the caller of the IWM4ECRE service optionally receives the service class token, which can be reused on consecutive calls as an input parameter. In this case, there is no need to obtain the CML to serialize against a policy update. As a consequence, the performance improves by reducing CML contention when creating enclaves.

This lock is held along the search for a service class that matches a arriving transaction. It is in effect during the process of the WM4ECRE macro service that creates an enclave and classifies it to a service class.

Notice that the IWM4ECREA macro service has a double function:

- Select a service class to an arriving transaction though the classification rules.
- Create an enclave (ENCB).

22.3.1 Invocation and use

To implement the fast path classification, the transaction manager uses IWM4ECRE with these parameters:

SERVCLS=servcls An optional output parameter when *classify* has been specified. It receives the service class token that matches the classification attributes. The token allows the caller to use fast path classification for other incoming transactions that have the same classification attributes as the current transaction. This token is passed back to that Workload Manager with the INSERVCLS parameter.

INSERVCLS=servcls An optional input parameter that contains the service class token of a previous classification call. However, the transaction manager caller must ensure that the classification attributes of the transaction match the service class token. If the service class token is still valid, an enclave is created and associated with the service and, optionally, with a report class information that is in the token. If the service class token is not valid, IWM4ECRE performs the full classification by using the information of the CLSFY parameter block and return code 4, with reason code IwmRsnCodeNewServcls.

22.4 Workload Manager I/O priority group

There are situations in the system where, during installation, settings should be selected to protect key business transactions. Before z/OS 2.1, you could choose long-term Workload Manager options, such as these two:

- Storage protection
- CP protection

With z/OS 2.1, it is possible to assign long-term I/O protection with transactions that are extremely I/O sensitive by assigning a service class with the HIGH I/O priority group option. The Workload Manager dynamically adjusts the I/O priority, based on how well goals are met and whether the I/O response time can contribute to achieving the goal.

22.4.1 Invocation and use

When the I/O priority of a certain transaction should be modified, Workload Manager first needs to determine how other transactions are affected. To do this, it checks the set of devices (named device set) that are used most by each service class transaction to access their data sets. This provides the access pattern information. The relationship between service class transactions and the device set is refreshed periodically every 10 minutes. Then, Workload Manager dynamically changes I/O priorities between service classes by using the same device set information.

If the access patterns change during the 10-minute interval and there are observable I/O delays, Workload Manager modifies only the I/O priorities when the device set is refreshed. In the worst case, this might take up to 10 minutes. Therefore, the I/O priority is not raised immediately and goals might be missed. In other words, Workload Manager does not micro-manage the I/O priorities and changes a service class period's I/O priority infrequently.

The solution is to introduce the concept of I/O priority groups for long-term I/O protection. Then, I/O-sensitive service classes are assigned to the HIGH group, and they always have higher I/O priority than service classes in the NORMAL group. NORMAL (default) and HIGH

are the only valid groups. Define the priority in the Service Class panel of the Workload Manager policy application. The HIGH group is allowed only if a dynamic I/O priority management is enabled. If it is, the Workload Manager dynamically manages your I/O priorities, based on service class goals and importance. See Figure 22-1. Specify Enable I/O Priority Groups (YES).

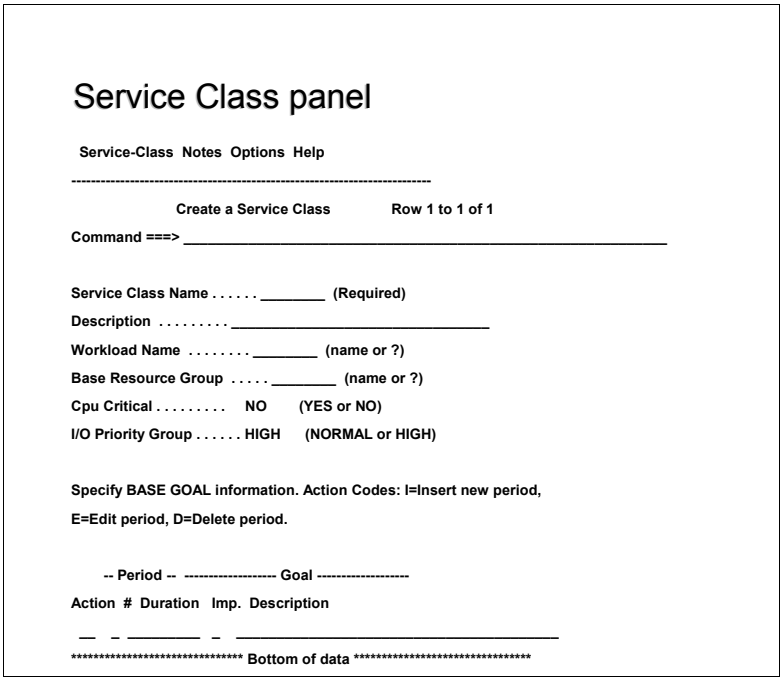


Figure 22-1 Service Class panel with I/O priority group.

API enhancements

Implementation of I/O priority groups also results in Workload Manager API enhancements in z/OS 2.1. These APIs are the ones that are used by performance monitors in getting performance data from the Workload Manager:

IWMRQRY — This query gets information about address space-related general execution delays. The answer area that is mapped by IWMWRQAA is enhanced according to REQFASD. A flag is added to the RQAEFLG1 field:

RqaeIOPrioHigh (Bit 6), which is the same as RasdIOPrioHigh

IWMPQRY — This interface returns a representation of the active policy. The answer area that is mapped by IWMSVPOL is extended. A flag is added to SVPOLCFL of the service class definition section, SVPOLCD:

SVPOLIPG (Bit 7), which is an indicator for I/O priority group

SMF 99 subtypes 2 and 6 have information about I/O priority groups.

The RMF post processor Workload Activity report displays the I/O priority groups information at the head of a service class page.

22.4.2 Migration and coexistence considerations

- For toleration and coexistence, APAR OA37824 is required on z/OS 1.12 and 1.13 systems because dynamic I/O priority management is a Parallel Sysplex-wide function.

Systems earlier than V2.1 are enabled by OA37824 to recognize the HIGH I/O priority group.

- I/O priorities are honored at the disk system level. Therefore, APAR OA37824 is required on any z/OS Version 1.12 or 1.13 system that shares the disk system with a z/OS 2.1 system, even if they are not in the same Parallel Sysplex.
- Assigning service classes to the HIGH I/O priority group is possible only with the z/OS 2.1 Workload Manager's Interactive System Productivity Facility (ISPF) application or IBM z/OS Management Facility (z/OSMF) Version 2.1.
- As soon as one service class is assigned to I/O the HIGH priority group, the function level of the service definition is changed to LEVEL029.
- Any service definition at function level 29 cannot be extracted, displayed, modified, installed, or activated by a Workload Manager application that is earlier than z/OS 2.1.

22.5 Improved accuracy for the IWMEQTME service

This enhancement applies to applications that capture the accumulated processor time when running under an enclave. With z/OS 2.1, the Workload Manager introduces a parameter on the IWMEQTME macro service, which is an enclave CPU time query. Optionally, it produces an exact value for the processor accumulated time that is returned by this service.

22.5.1 Invocation and use

Any dispatchable unit processor time is derived and accumulated only by MVS when this dispatchable unit is preempted or enters a wait state. Then, the total enclave dispatchable unit processor time that is returned by the IWMEQTIME service is the total accumulated time for the enclave until the requesting dispatchable unit was dispatched for the last moment. All dispatchable unit active processor time since then is missing, leading to slightly smaller results.

To solve the problem, z/OS 2.1 introduces an optional input keyword, `CURRENT_DISP=YES`, in the IWMEQTIME macro to indicate whether the Workload Manager should call the dispatcher (CALLDISP macro) before querying the enclave processor times. Calling the dispatcher forces MVS to include in the returned dispatchable unit the processor time at ENCB, which is the time that has elapsed since the last dispatching of the requesting dispatchable unit. When using this keyword, the requesting dispatchable unit must be a task, not an enclave service request block (SRB).

This function is optional because calling the dispatcher is costly. The default is not to call the dispatcher to avoid performance degradation for other users. Therefore, use this function only when accuracy is essential.

This option benefits single dispatchable unit enclaves, mostly. For enclaves with multiple dispatchable units, such as task control blocks (TCBs) or SRBs that are running in parallel, the benefit is marginal because the call to the dispatcher updates the time for only the current (calling) task, not the others that are running dispatchable units. Example 22-1 shows how to use this option.

Example 22-1 CURRENT_DISP option

```
CURRENT_DISP=NO (default)
CURRENT_DISP=YES
```

Figure 22-2 shows an example of Assembler source code that activates the option.

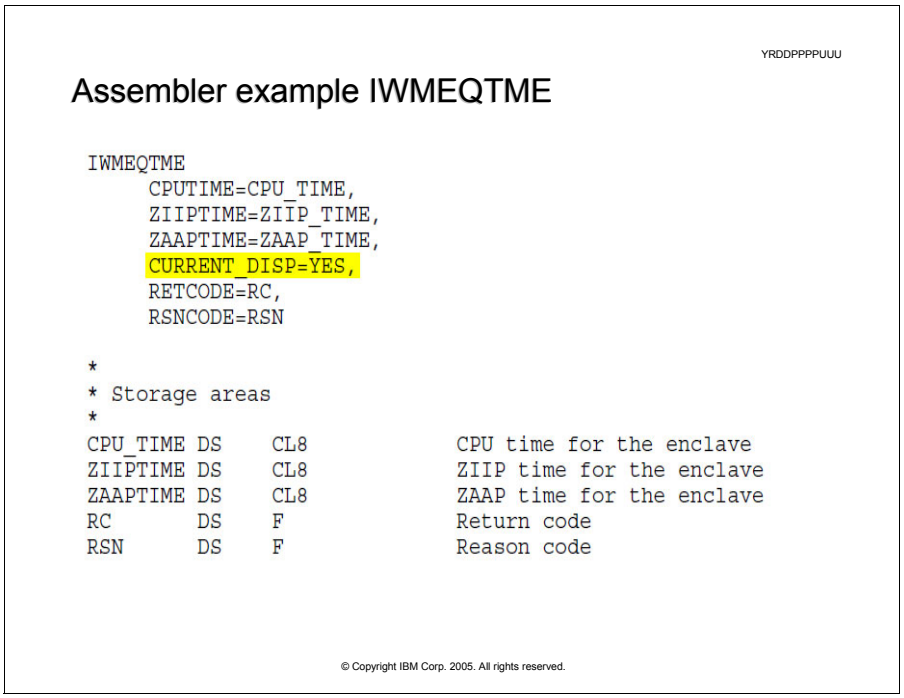


Figure 22-2 IWMEQTME macro example

The first user of this function is IBM WebSphere Application Server software.

22.5.2 Migration and coexistence considerations

Invoking IWMEQTME with the **CURRENT_DISP=YES** parameter while in SRB mode results in ABEND 05D-10.

Invoking IWMEQTME with the **CURRENT_DISP=YES** parameter while I/O and external interrupts are disabled results in ABEND 05D-08.

22.6 More Workload Manager application environments

The limit of application environments (AE) defined in the Workload Manager service definition is raised from 999 to 3000.

22.6.1 Invocation and use

Before z/OS 2.1, the maximum number of application environments was 999. Especially with large IBM DB2 parallel queries in SAP environments, the limitation to 999 application AEs in the Workload Manager service definition could become too low.

In z/OS 2.1, this number is increased to 3000. The Workload Manager task in z/OSMF is enhanced accordingly.

22.6.2 Installation

If you need more than 999 AEs, you must follow these guidelines to allocate a Workload Manager couple data set (CDS), which can hold the required number of AE objects:

- Allocate CDSes by using the Values task in the Workload Manager ISPF application. If a service definition with more AEs than the number that is allowed for the current Workload Manager CDS would be installed, the Workload Manager ISPF application displays message IWMAM047: Workload Manager CDS is too small to hold the service definition.
- You can also allocate a Workload Manager CDS by some JCL, as provided in the IWMFTCDS member of SYS1.SAMPLIB:
  - Enter a number for the APPLENV item, up to 3000.
  - Run the job on any z/OS 2.1 system in the IBM Parallel Sysplex.

22.6.3 Migration and coexistence considerations

As soon as more than 999 AEs are defined with the z/OS 2.1 Workload Manager ISPF or with z/OSMF 2.1, the function level of the service definition is raised to LEVEL029, with this limitation and capability:

- Any service definition at function level 29 cannot be extracted, displayed, modified, installed, or activated by a Workload Manager ISPF Application that is earlier than z/OS 2.1.
- If a service definition at LEVEL029 is installed to the Workload Manager couple data set by z/OS 2.1, systems with z/OS V1.12 and 1.13 of the same Parallel Sysplex can activate the policy.

22.7 Documentation and reference

See the following IBM publications for more information:

- *z/OS MVS Planning: Workload Management*, *Version 2 Release 1,* SC34-2662-01

  http://ibm.co/1kbhst6
- *z/OS MVS Programming: Workload Management Services, Version 2 Release 1*, SC34-2663-01

  http://ibm.co/1tt1UdB
- *z/OS MVS System Management Facilities (SMF)*, SA38-0667-01

  http://ibm.co/1oLdADr
- *z/OS RMF Report Analysis*, SC34-2665

  http://ibm.co/U5VFfQ

23

IBM Data Facility Sort

This chapter describes the various enhancements and functions that were introduced with IBM Data Facility Sort (DFSORT) in IBM z/OS Version 2 Release 1 (2.1). DFSORT is high-performance sort, merge, copy, analysis, and reporting software for z/OS, which you can use to sort, merge, and copy data sets.

DFSORT provides versatile data handling capabilities at the record, field, and bit level. You can use it either to do simple tasks, such as alphabetizing a list of names, or as an aid for complex tasks, such as taking inventory or running a billing system.

This chapter covers the following enhancements and functions:

- 23.1, “Functional enhancements” on page 296
- 23.2, “RAS enhancements” on page 299
- 23.3, “Dynamic RAS enhancements” on page 300
- 23.4, “Support for 64-Bit callers with 64-bit addressed records” on page 302

© Copyright IBM Corp. 2014. All rights reserved.

23.1 Functional enhancements

z/OS 2.1 DFSORT includes the following functional enhancements:

- Alphanumeric tests
- PARSE enhancements
- Symbol enhancements
- Add string at end of VLR (variable-length records)

23.1.1 Invocation and use

The subsections that follow describe the four functional enhancements.

Alphanumeric tests

Before DFSORT 2.1, it was not possible to compare the contents of fields if they contained only characters in a specific set (for example, A - Z, a - z, or 0 - 9). The solution in z/OS 2.1 is to implement alphanumeric comparison tests, such as, the `UC`, `LC`, `MC`, `UN`, `LN`, and `MN` keywords. This is similar to the previously available `NUM` keyword.

DFSORT 2.1 allows you to test a field for various combinations of alphanumeric characters or non-alphanumeric characters by using binary (BI) format. Previously, it was necessary to code multiple compare conditions. This support makes it possible to specify various sets of characters by using a single compare condition. The following list shows the meaning of each comparator:

`UC` Uppercase characters (A-Z)
`LC` Lowercase characters (a-z)
`MC` Mixed case characters (A-Z, a-z)
`UN` Uppercase and numeric characters (A-Z, 0-9)
`LN` Lowercase and numeric characters (a-z, 0-9)
`MN` Mixed case and numeric characters (A-Z, a-z, 0-9)

Example 23-1 shows ways of using the keywords.

Example 23-1 Alphanumeric test

```
Alphanumeric Tests Usage:
You can use alphanumeric test keywords(UC, LC, MC, UN, LN and MN) in the
following comparison operands: COND, INCLUDE, OMIT, BEGIN, END, WHEN and
TRLID.
Examples:
INCLUDE COND=(11,1O,BI,EQ,MC)
OMIT COND=(5O,5,BI,EQ,LC)

Parse Field Tests:
PARSE function can now be used with alphanumeric test keywords(UC, LC, MC, UN,LN
and MN) to start or end when a character from any of various alphanumeric
character sets.
Example:
INREC PARSE=(%O1=(ENDBEFR=UC,FIXLEN=5)),BUILD=(%O1)
```

PARSE enhancements

It is common to have logical records with a very large number of delimited fields. IBM clients often have records with consecutive fields that they want to parse, based on a specific set of characters. With DFSORT 2.1, you can use up to 1000 parsed fields (%0-%999) with the PARSE function. The previous limit was 100 parsed fields (%0-%99). These added parameters provide this capability:

- STARTAFT=an, STARTAT=an, ENDBEFR=an, and ENDAT=an can now be used with the PARSE function to start or end when a character from any of the various alphanumeric character sets is found.
- REPEAT=v is a PARSE option that can be used to repeat a particular parse field definition multiple times.

The REPEAT=v allows you to easily ignore or process consecutive delimited fields of the same form. The alphanumeric tests allows you to specify various sets of characters by using a single PARSE keyword. Example 23-2 shows syntax examples.

Example 23-2 Parse enhancements

```
More parsed fields.
You can now use up to 1000 parsed fields (%0-%999) with the PARSE function; the
previous limit was 100 parsed fields (%0-%99).
Example:
OUTREC PARSE=(%121=(ENDBEFR=C',',FIXLEN=12),
%322=(ENDBEFR=C',',FIXLEN=8),
%999=(FIXLEN=5)),
BUILD=(%121,X,%322,X,%999)

Repeating Parse Fields:
REPEAT=v can be used with %n, %nn or %nnn to specify v identically defined
consecutive parsed fields for which data is to be extracted. The parsed fields
will start with the %n, %nn or %nnn field you select and be incremented by one for
each repeated parsed field.
Example:
INREC PARSE=(%=(ENDBEFR=C',',REPEAT=3),
%11=(ENDBEFR=C',',FIXLEN=10,REPEAT=4)),
BUILD=(%11,X,%12,X,%13,X,%14)
```

Symbol enhancements

Before z/OS 2.1, several DFSORT operands, especially those of the form KEYWORD=n, did not support symbols.

Now, KEYWORD=sym supports operands of the form KEYWORD=*n*, where n is a number. Therefore, symbols can now be used with more DFSORT functions, such as ID=sym, SEQ=sym, ABSPOS=sym, and FIXLEN=sym.

The following operands can have symbols:

```
ABSPOS
ACCEPT
ADDPOS
AVGRLEN
DO
ENDPOS
ENDREC
FIXLEN
```

```
ID
IFOUTLEN
INCR
LENGTH
LINES
MAXLEN
RECORDS
REPEAT
SAMPLE
SEQ
SKIPREC
SPLIT1R
SPLITBY
START
STARTPOS
STARTREC
STOPAFT
SUBPOS
```

See the symbol enhancement example in Example 23-3.

Example 23-3 Symbol enhancement

```
OUTFIL REPEAT=Mult,
IFOUTLEN=out_length,
IFTHEN=(WHEN=GROUP,RECORDS=Num_records,
PUSH=(id_col:ID=id_length))
```

Strings at the ends of variable-length records

Before DFSORT Version 2.1, it was not possible to add a particular string to the end of a variable length logical record. To accomplish that, you had to write your own E35 exit logic.

Now you can add a string at end of variable-length records (VLR). For example, you can add specific characters at the end of each VB record, such as `X'0D0A'` (CRLF).

`VLTRAIL=string` is a OUTFIL option that you can use to insert a character string (`C'`*`string`*`'`) or hexadecimal string (`X'`*`yy...yy`*`'`) at the end of each variable-length OUTFIL output record. You can add a string of 1 - 50 characters at the end of each VB record. See Example 23-4.

Example 23-4 String added to OUTFIL

```
OUTFIL VLTRIM=C' ',VLTRAIL=X'0D0A
```

23.1.2 Migration and coexistence

The following are DFSORT and ICETOOL reserved words that are no longer allowed as symbols: LC, LN, MC, MN, UC, and UN.

If you used any of these words as a symbol previously, you must change them. For example, if you used `MC`, you can change it to `mc`.

23.2 RAS enhancements

The objective of these enhancements is to improve the reliability, availability, and serviceability (RAS) of DFSORT 2.1, mainly when invoked by its heavy user, IBM DB2 Utilities. There are three objectives:

- Provide virtual storage constraint relief below the 16 M line by using the following functions for dynamically allocated work data sets:
  - Extended TIOT (XTIOT)
  - Uncaptured UCBs
  - Above the16 M line, DSAB
- Reduce the lack of space conditions for work area data sets by allowing the possibility of extending them.
- Improve scalability for very large sorts by increasing the maximum size of data sets and memory object work files.

23.2.1 Invocation and use

There are three key RAS enhancements:

- Provide virtual storage constraint relief below the 16 M line.

  These are the three techniques for saving virtual storage below the 16 M line:

  - Extended TIOT (XTIOT)
  - Uncaptured UCB
  - Above the16 M line, DSAB
- Reduce the lack of space conditions for work area data sets.

  DFSORT sets virtual storage, buffers, control blocks, and so on for work area data sets at open time. Then, the secondary space allocation is not used. To circumvent this, you may define additional secondary space with a zero primary space, which will be used only if needed. See the Example 23-5. SORTWK03 is used only if no more space can be allocated to SORTWK01 and SORTWK02.

  Example 23-5 Zero primary space allocation

```
//SORTWK01 DD UNIT=SYSDA,SPACE=(CYL,(10,5))
//SORTWK02 DD UNIT=SYSDA,SPACE=(CYL,(10,5))
//SORTWK03 DD UNIT=SYSDA,SPACE=(CYL,(0,5))
```

- Improve scalability for exceptionally large sorts.

  Before DFSORT 2.1, there was an internal design limitation of 1,048,576 for the number of tracks for work area data sets. With Version 2.1, it is increased to 16,777,216 tracks. This allows the installation to take better advantage of the IBM DS8000® extended address volumes (EAV) feature, and it also allows DFSORT to use larger memory object work areas. Because DFSORT uses the full track block facility, this limit figures mentioned above also apply to the maximum number of physical blocks.

  As a consequence of the extended limit for tracks, DFSORT 2.1 can use up to 255 work area data sets. Therefore, its theoretical work space limit is 255 * 16,777,216 tracks, for a total of about 220 TB.

To further use large central storage configurations, the maximum amount of memory object storage that can be used as intermediate work area has been increased from 64 GB to 1 TB. A memory object is a piece of virtual storage located in data spaces (we may also have data in hiperspaces). When appropriate, they are used instead of or with the data sets work area to improve the performance of sort applications.

23.3 Dynamic RAS enhancements

DFSORT aggressively uses central (and virtual) storage, such as these examples:

- It is not uncommon to have over-commitment of central storage and paging resources causing page faults in other applications, causing overall performance degradation. In some cases, the problem is related to auxiliary storage shortages because large sorts allocated too much memory object (data space) or hiperspace storage.
- Unbalanced use of resources used by concurrent sort applications. Whoever gets in first is likely to consume most of what's available.

DFSORT 2.1 implements improve DFSORT installation defaults to control the use of central storage in these four ways:

- TUNE installation default to favor storage or disk work area space optimization
- defaults for EXPOLD and EXPRES parameters
- Allocation of central storage in smaller increments
- Storage-related installation defaults that can be specified as a percentage of *current* available memory resources

23.3.1 Invocation and use

The subsections that follow describe the four enhancements to control the use of central storage.

TUNE keyword: ICEPRMxx

At DFSORT V2R1, the `TUNE` keyword specifies whether DFSORT should favor optimization of central storage or data set work area space. Because these two resources are both critical to sort processing, DFSORT must synchronize their use. When more central storage can be used, it might be possible to reduce the data set work area space requirement. When central storage resources are constrained, additional data set work area space might be required. `TUNE` has the following options:

`STOR` (default)	Favours data set work area space. DFSORT allocates available central (and virtual) storage as needed in increments sized to balance use when multiple sorts are executing on the same system. As a result, the DFSORT data set work area allocation is increased to account for increased data set work area requirements in case central storage resources become constrained during the sort.
`DISK`	Favours central storage space. DFSORT allocates all available central (and virtual) storage required at initialization. As a result, DFSORT data set work area is decreased, based on the expected central storage use, to minimize the DASD space requirements.
`DDYN`	Favours central storage space. DFSORT allocates all of the available central storage required at initialization but only when dynamic allocation for the data set work area is being used. As a result,

DFSORT dynamic work data set allocations are decreased, based on the expected central storage use, to minimize the disk space requirements. When the data sets work area has been preallocated in JCL or by an invoking program, DFSORT allocates available central storage as for the STOR option.

OLD (default) — This was the only option before DFSORT 2.1. DFSORT allocates central storage as needed by using increments of a fixed size. DFSORT's dynamic data set work area allocation is decreased, based on the expected central storage use.

The default (STOR) causes DFSORT to be more conservative in reducing data set work area space allocations. This is to compensate for potential changes in available central storage that could cause an unexpected increase in the data set work area space requirements. If available data set work area space is constrained, consider using TUNE=DISK, which causes DFSORT sort to obtain all of the storage that it needs on a first-come, first-served basis but allows DFSORT to be more aggressive in reducing disk work allocations, based on expected central storage use.

defaults for EXPOLD and EXPRES parameters

EXPRES specifies the minimum amount of available central storage to be reserved for use by non-Hipersorting, and non-memory object sorting (data spaces). At DFSORT V2R1 the default is being changed to 10%. Prior to DFSORT V2R1 is 0. With EXPRES=10%, we always make sure there' is at least a small amount of storage left available for non sort applications if a sudden need arises.

EXPOLD specifies the maximum total amount of least reference used (LRU) pages in central storage to be used by DFSORT at any one time by all hipersorting, memory object sorting (data spaces). Theses pages includes only the ones been *unreferenced* for a sufficiently long period of time, Then, DFSORT considers them eligible to be paged out to auxiliary storage to make room for hiperspace, memory object or data space data. There are two different ways of defining EXPOLD:

- EXPOLD=n specifies a fixed value for EXPOLD
- EXPOLD=p% specifies a value for EXPOLD that varies based on either the configured central storage or the amount of old storage in use by other applications at run time.

At DFSORT V2R1 the default is being changed to 50%. The default prior to DFSORT V2R1 is MAX. The former default would cause pages from other workloads to be paged out and drive up auxiliary storage use.

Allocation of central storage in smaller increments.

There two methods for allocating storage and data set work area space:

- In increments

  The advantage is to allow for checking available resources and avoiding system over-commitment,

- All you can get at once

  This method provides these advantages:

  - It requires less overhead.
  - If you use increments, you cannot be sure that there will be available storage when you need it.
  - You may share this storage and data set work area space among concurrent sorts.

If you use TUNE=STOR, DFSORT 2.1 allocates available central (and virtual) storage, as needed, in increments that are sized to balance use when multiple sorts are running on the same system.

Storage-related installation defaults

Storage-related installation defaults (as EXPMAX, EXPRES, and EXPOLD) can be specified as a percentage of *current* available memory resources rather than as a percentage of configured memory.

23.3.2 Migration and coexistence

To force DFSORT to operate as it did previously, in z/OS Versions 1.12 or 1.13, set the following installation defaults:

- TUNE=OLD
- EXPOLD=MAX
- EXPRES=0

23.4 Support for 64-Bit callers with 64-bit addressed records

Before DFSORT 2.1, it was not possible for AMODE64 program to invoke DFSORT.

23.4.1 Invocation and use

With DFSORT 2.1, eligible user programs and exits can now be written to accomplish the following tasks:

- Call DFSORT from a AMODE64 program. The invoking program uses the 64-bit parameter list and must use ICEMAN64 or SORT64 as the entry point name for the LINK, ATTACH, or XCTL macro.
- Pass 64-bit addressed parameters to DFSORT, which accepts these three types of parameter lists:
  - 24-bit parameter list
  - Extended (31-bit) parameter list
  - 64-bit parameter list (recommended)

  The layout of the 64-bit parameter list is modified.
- Define AMODE64 for DFSORT exit routines, such as E15, E35, and E32. The interface to these exits is slightly modified.
- Pass 64-bit addressed records to DFSORT by using E15, E32, and E35 exits.

The clear benefit is to save addresses below the 2 G bar.

23.4.2 Interactions and dependencies

The DFSORT exit routines E18, E39, E61 are not allowed in AMODE 64.

23.4.3 Migration and coexistence

Error messages:

- ICE290A INVALID 64-BIT INVOCATION PARAMETER LIST - REASON CODE IS rsn

  DFSORT was invoked with a 64-bit invocation parameter list. However, an error was found in this parameter list. Reason code values (rsn) are indicated. DFSORT terminates.

- ICE291I AMODE FLAG FOR EXIT WITHOUT CORRESPONDING EXITADDRESS IGNORED

  An AMODE flag was set for an exit for which a corresponding exit address was not present in the parameter list. For example, the AMODE 64 flag was set for the E15 exit, but the E15 exit address was zeros in the 64-bit invocation parameter list. System action: Processing continues. Each AMODE flag for an exit without a corresponding exit address is ignored.

- ICE034A MODS STATEMENT OPERAND ERROR

  A reason code was added for when MODS N64 parameter is used incorrectly.

  N64 was specified for the fourth parameter, but DFSORT was not invoked by using a 64-bit invocation parameter list. System action: The program terminates. Programmer response: N64 is not specified as the fourth parameter if DFSORT was not invoked using a 64-bit invocation parameter list.

23.4.4 Documentation and reference

The following documentation is available in the IBM z/OS 2.1.0 Knowledge Center:

`http://ibm.co/1oM3OdZ`

- *z/OS DFSORT Application Programming Guide*, SC23-6878
- *z/OS DFSORT Installation and Customization*, SC23-6881
- *z/OS DFSORT Messages, Codes and Diagnosis Guide*, SC23-6879
- *z/OS DFSORT Tuning Guide*, SC23-6882

These resources are also helpful:

- DFSORT web page

  `http://www.ibm.com/storage/DFSORT`

- DFSORT Hotline: DFSORT@us.ibm.com

24

z/OS Binder feature

This chapter describes the Binder enhancements, functions, and features that were introduced with IBM z/OS Version 2 Release 1 (2.1):

- 24.1, “Boundary alignment” on page 306
- 24.2, “Symbol resolution tracing” on page 309
- 24.3, “API DLL data sets resident support” on page 311
- 24.4, “LONGPARM option” on page 313
- 24.5, “Documentation and reference” on page 314

© Copyright IBM Corp. 2014. All rights reserved.

24.1 Boundary alignment

In early versions of the operating system, load modules that were built from object modules were composed of sections that were always double-word aligned. The ability to page-align a section at bind time was also provided. Later, the ability to have quad-word aligned sections in objects was added. When GOFF objects were introduced in the early 1990s, the architecture supported the ability to define every alignment, from byte to 2 GB, but it was never fully used.

The binder can now respect alignment information to allow more granularity in aligning the sections in modules. By aligning sections more granularity, you gain usability and, possibly, performance benefits without needlessly wasting space.

This support comes in three parts:

- ALIGNT control statement for binding load modules and program objects
- Alignment attributes coming from ESD records
- ALIGNT regular binder API VERSION=8

24.1.1 Invocation and use of ALIGNT control statement

ALIGNT is a binder control statement, which is a generalization of the PAGE control statement. The syntax is shown in Example 24-1.

Example 24-1 Syntax of ALIGNT Statement

```
ALIGNT boundary,sectionname[(classname1 [,classname2]...)]
```

Like PAGE and other binder control statements, ALIGNT causes a non-permanent change. You must use the ALIGNT option every time that you bind or rebind your programs.

The ALIGNT statement is different from PAGE in these ways:

- You can specify the alignment boundary.
- You can specify only a single section name on each statement.
- You have the option to specify one or more class names for that section.
- There is no interaction with the ALIGN2 binder option.

The boundary can be 0 - 4096 and must be a power of 2, where 0 means “reset,” as if ALIGNT was never specified for that section. The ALIGNT option that is used for a section name with no classes affects all but merge classes. When you want to merge classes, they must be explicitly specified. Only merge classes of parts are allowed; pseudo-registers are not. ALIGNT for the same section name but with classes is subtractive from the earlier specification with no classes, which means that multiple statements can be used.

Example 24-2 on page 307 and Example 24-3 on page 307 show you a small example with two object modules, OBJ1 and OBJ2, bound into a load module in a PDS but not using any Alignment or PAGE option. Each happens to be X'18' bytes long.

Example 24-2 Binder Module MAP unaligned

```
---------------
CLASS B_TEXT       LENGTH = 30 ATTRIBUTES = CAT, LOAD, RMODE= 24
                   OFFSET = 0 IN SEGMENT 001   ALIGN = DBLWORD
---------------
SECTION CLASS                              ------- SOURCE --------
 OFFSET OFFSET NAME        TYPE   LENGTH DDNAME   SEQ   MEMBER
             0 OBJ1       CSECT       18 OBJS      01   **NULL**
            18 OBJ2       CSECT       18 OBJS      01   **NULL**
```

Example 24-3 AMBLIST MODLIST unaligned

```
RECORD# 1 TYPE 20 - CESD     ESDID 1                   ESD SIZE 32
           CESD#    SYMBOL    TYPE    ADDRESS   R/R/A    ID/LENGTH(DEC)  (HEX)
              1    OBJ1      00(SD)    000000    00             24         18
              2    OBJ2      00(SD)    000018    00             24         18
```

The next example, shown in Example 24-4 and Example 24-4, was produced with the ALIGNT 32,OBJ2 control statement. Notice that the offset of OBJ2 is now aligned to X'20'.

Example 24-4 Binder Module MAP aligned

```
---------------
CLASS B_TEXT     LENGTH = 38  ATTRIBUTES = CAT, LOAD, RMODE= 24
                 OFFSET =  0 IN SEGMENT 001    ALIGN = 32 BYTE
---------------
SECTION    CLASS                              ------- SOURCE --------
OFFSET    OFFSET  NAME         TYPE   LENGTH DDNAME  SEQ   MEMBER
               0  OBJ1        CSECT       18 OBJS     01   **NULL**
              20  OBJ2        CSECT       18 OBJS     01   **NULL**
```

Example 24-5 AMBLIST MODLIST aligned

```
***PAGE ALIGNMENT REQUIRED
...
RECORD# 1     TYPE 20 - CESD    ESDID 1                   ESD SIZE 32
              CESD#     SYMBOL   TYPE    ADDRESS  R/R/A ID/LENGTH(DEC) (HEX)
                1      OBJ1     00(SD)    000000   00          24        18
                2      OBJ2     00(SD)    0000 20  00          24        18
```

24.1.2 Invocation and use of alignment from ESD records

Although the binder in earlier versions tolerated alignments (specified by a power of 2) for element (class) alignments, it supported only double-word, quad-word, and page values. Before the boundary alignment support, the binder changed unsupported class alignments on input from GOFF objects to the next most-restrictive alignment. So, for example, a 256 byte-aligned request would be changed to a page-aligned request.

With the boundary alignment support, for a GOFF object, a user can now tell the binder to not change any valid alignment values on input. To enable this, you must create a COMPAT(ZOSV2R1) or COMPAT(CURRENT) program object. Now all alignments up to 4096 (PAGE) can be entered, specified as a power of 2.

Restriction: Alignments that are less than double-word are still changed to double-word. Otherwise, assumptions that are made by assemblers and compilers might be invalid.

The High Level Assembler (HLASM) feature offers three ways to set alignments:

- The natural alignment of the type. For example, this aligns a pseudo register on a double-word boundary:

  ```
  alignDW DXD OD
  ```

- The SECTALGN option to align sections and the default for all classes. Unless the GOFF is used, only double-word (default) and quad-word sections can be created. Example:

  ```
  PROCESS SECTALGN(32)
  ```

- The CATTR statement ALIGN keyword to align a class. If it is a merge class, it also aligns all the parts within it. Example:

  ```
  MYPARTS CATTR PART(partQW),ALIGN(4),RMODE(31)
  ```

To have the binder honor the ESD alignment of the object, specify `COMPAT(ZOSV2R1)` or `COMPAT(CURR)` as a binder option. For example, if OBJ1 and OBJ2 (both 24 bytes long or X'18') is assembled and bound with the following options:

- High Level Assembler options that are used are for octa-word alignment:

  ```
  GOFF,SECTALGN(32)
  ```

- COMPAT(CURR) specified as a binder option
- The program is saved as a program object (PDS/E or UNIX file system)

Then the binder preserves (honors) the input ESD alignment option, as shown in the Binder MAP output in Figure 24-1.

```
                        *** M O D U L E  M A P ***

---------------
CLASS  B_TEXT           LENGTH =       38  ATTRIBUTES = CAT,   LOAD, RMODE= 24
                        OFFSET =        0 IN SEGMENT 001     ALIGN = 32 BYTE
---------------

 SECTION    CLASS                                          ------- SOURCE --------
  OFFSET   OFFSET  NAME                TYPE    LENGTH  DDNAME   SEQ  MEMBER

               0  OBJ1                CSECT        18  OBJ       01  OBJ1

              20  OBJ2                CSECT        18  OBJ       01  OBJ2
```

Figure 24-1 Binder MAP output

Binder API update for ALIGNT

The binder API function ALIGNT version 8 extends the parameter list to include an alignment value and a list of classes.

24.1.3 Interaction and dependencies

If a program is bound by using `COMPAT(ZOSV2R1)` or `COMPAT(CURR)` to preserve alignment information from ESD records and then saved to a load module (PDS), the load module will be properly aligned. However, the preserved alignment information cannot be represented in a load module. Message IEW2436I is issued, as this example:

```
IEW2536I 5257 ESD ALIGNMENT FOR SYMBOL OBJ1 CHANGED FROM  32 TO  16.
```

Rebinding the load module with the now changed alignment might not produce the results that you want. Use the ALIGNT control statement.

There are no loader dependencies because this alignment supports only more granular alignment within the module. The loader still supports only double-word and page alignment of programs (and segments).

24.1.4 Migration and coexistence

If a program object is built by using `COMPAT(ZOSV2R1)` to preserve alignment, it can be loaded and run on any z/OS system, as early as z/OS 1.1. However, it cannot be inspected or reprocessed on any system before z/OS 2.1.

24.2 Symbol resolution tracing

The most common problem that is encountered during binding is unresolved symbols. An unresolved symbol is a reference for which the binder could not find a definition. Sometimes, finding the root cause is difficult for complex modules that contain many input parts. This might require using autocall and being rebound numerous times. The underlying reason for an unresolved symbol can be difficult to determine. Starting with z/OS 2.1, a binder **SYMTRACE** option is added to help determine an unresolved symbol error.

24.2.1 Invocation and use

The **SYMTRACE** parameter can be used any place that a binder parameter can be specified, for example:

- `JCL PARM=`
- `SETOPT` control statement
- UNIX ID command **-b** option
- UNIX **c89** command, but only *uppercase*

The option value is the name of a symbol. It can be an entry point, an external reference, or a private name. Figure 24-2 on page 310 shows an example of use with JCL.

```
//BIND1    EXEC PGM=IEWL,PARM='CASE=MIXED')
//SYSLIN DD *
  SETOPT PARM(SYMTRACE('abc')) * for next INCLUDE(s)
  INCLUDE OBJ11
  SETOPT PARM(SYMTRACE('def')) * for next INCLUDE(s)
  INCLUDE OBJ12
 NAME MYMOD1
  SETOPT PARM(SYMTRACE('ghi')) * for next INCLUDE(s)
  INCLUDE OBJ21
  SETOPT PARM(SYMTRACE(''))    * turn off for rest
  INCLUDE OBJ22
 NAME MYMOD2
```

Figure 24-2 SYMTRACE in JCL example

The **SYMTRACE** option produces only informational messages about the symbol that is being traced. Furthermore, only one symbol name can be specified with each **SYMTRACE** option. The **SYMTRACE** option follows the usual binder option conventions:

- If **SYMTRACE** is specified more than once, the last specification is used.
- If the **SYMTRACE** symbol name is in mixed uppercase and lowercase, you must either:
  - Have previously specified the option `CASE=MIXED`
  - Enclose the symbol name in single quotation marks
- UNIX commands require either `-V` or `MSGLEVEL=0` as the default is to not display INFORMATIONAL messages

There are 12 messages that can be produced by **SYMTRACE**. Each is prefixed by **SYMTRACE**, followed by the message ID for easy identification. There are four categories:

- IMPORT (DLL)
- Reference found
- Definition found and origination information available
- AUTOCALLs

Figure 24-3 displays messages that are issued when attempting to IMPORT the symbol.

```
IEW2336I SYMTRACE: SYMBOL symbol OF TYPE symbol-type IS EXPECTED TO BE IMPORTED FROM DLL
dllname.

IEW2337I SYMTRACE: SYMBOL symbol OF TYPE symbol-type CAN NOT BE IMPORTED FROM DLL
dllname BECAUSE THE VALUE OF OPTION DYNAM IS NO.

IEW2423I SYMTRACE: SYMBOL symbol OF TYPE symbol-type WILL BE IMPORTED FROM DLL dllname.

IEW2424I SYMTRACE: SYMBOL symbol OF TYPE symbol-type CAN NOT BE IMPORTED BECAUSE THE
BINDING SCOPE OF ONE OR MORE REFERENCES TO THIS SYMBOL IS NOT IMPORT-EXPORT.
```

Figure 24-3 symtrace IMPORT messages

Figure 24-4 on page 311 shows messages that are issued when a reference to the symbol is found.

```
IEW2417I SYMTRACE: SYMBOL symbol IS REFERENCED IN SECTION sectname IN MEMBER memname IN
THE DATA SET IDENTIFIED BY DDNAME ddname WITH CONCATENATION NUMBER number FOR DATA SET
NAME dsname. [NAME SPACE = name-space]

IEW2418I SYMTRACE: SYMBOL symbol IS REFERENCED IN SECTION sectname IN MEMBER
archive-file-member-name IN THE UNIX FILE pathname. [NAME SPACE = name-space]
```

Figure 24-4 symtrace reference messages

Figure 24-5 shows messages that display when a definition of the symbol is found.

```
IEW2419I SYMTRACE: SYMBOL symbol IS DEFINED IN SECTION sectname IN MEMBER
memname IN THE DATA SET IDENTIFIED BY DDNAME ddname WITH CONCATENATION NUMBER
number FOR DATA SET NAME dsname. [NAME SPACE = name-space]

IEW2420I SYMTRACE: SYMBOL symbol IS DEFINED IN SECTION sectname IN MEMBER
archive-file-member-name IN THE UNIX FILE pathname. [NAME SPACE = name-space]
```

Figure 24-5 symtrace definition messages

Figure 24-6 shows messages that display when a definition is found and when binding from an object module or program object with `COMPAT(ZOSV1R5)` or a later z/OS version. In this case, origination information is maintained.

```
IEW2421I SYMTRACE: SYMBOL symbol DEFINITION ORIGINALLY CAME FROM MEMBER memname
IN THE DATA SET dsname. [NAME SPACE = name-space]

IEW2422I SYMTRACE: SYMBOL symbol DEFINITION ORIGINALLY CAME FROM MEMBER
archive-file-member-name IN THE UNIX FILE pathname. [NAME SPACE = name-space]
```

Figure 24-6 SYMTRACE definition messages with origination information

Figure 24-7 shows messages that result when attempting to AUTOCALL a symbol.

```
IEW2546I SYMBOL symbol IS BEING SEARCHED FOR IN THE DATA SET IDENTIFIED BY
DDNAME ddname.

IEW2547I SYMTRACE: SYMBOL symbol IS BEING SEARCHED FOR IN PATH pathname.
```

Figure 24-7 symbols AUTOCALL messages

24.3 API DLL data sets resident support

The Binder feature provides C/C++ APIs in the form of DLLs. They provide high-level language applications, such as IBM Language Environment, a more natural way of using the binder regular and fast data access APIs. Those DLLs were introduced in z/OS 1.9. Starting with z/OS 1.12, XPLINK DLLs were added. Unfortunately, in the current implementation, DLLs install in the UNIX file system only, which can require more runtime modifications, such as inclusion of a `LIBPATH` statement. Starting with z/OS 2.1, IBM includes equivalent DLLs into MVS data sets, with the matching side files also.

24.3.1 Invocation and use

z/OS 2.1 includes parts. However, the header files are still the same, and the application provider still needs to access the z/OS UNIX path.

The DLLs are included in the SYS1 library.SIEAMIGE, which is in the default Link List search path:

- SYS1.SIEAMIGE(IEWBNDD)
- SYS1.SIEAMIGE(IEWBNDDX) [XPLINK]

Side files are in the SYS1 library.SIEASID and are needed because DLL name match is exact.

- SYS1.SIEASID(IEWBNDD)
- SYS1.SIEASID(IEWBNDDX) [XPLINK]

Example 24-6 is a sample batch job that shows how to use the DLL support.

Example 24-6 Sample IEWBNDDX

```
// JCLLIB ORDER=(CBC.SCCNPRC)
// SET SRCLIB=SYS1.SAMPLIB
// SET MODLIB=&SYSUID..PDSELIB
//*
//CBAPCCC EXEC PROC=EDCXCBG,
// CPARM='OPTFILE(DD:COPTS)',
// BPARM='OPTIONS(BOPTS)',
// INFILE=&SRCLIB(IEWAPCCC)
//COMPILE.COPTS DD *
 NOSEARCH,SEARCH(/usr/include)
 RENT,LIST
/*
//*
//BIND.SYSLMOD DD DSN=&MODLIB(IEWAPCCC),DISP=SHR
//SIDEDECK DD DSN=SYS1.SIEASID,DISP=SHR
//BIND.SYSIN DD *
 INCLUDE SIDEDECK(IEWBNDDX)
/*
//BIND.BOPTS DD *
 DYNAM=DLL
/*
```

Use from z/OS UNIX is also supported, as Example 24-7 shows.

Example 24-7 UNIX invocation of IEWBNDD

```
export _C89_CSUFFIX_HOST=samplib
c89 -Wc,"dll,lang(extended)" -co iewapccc.o "//'sys1.samplib(iewapccc)'"

export _C89_EXTRA_ARGS=1
c89 -Wl,dll iewapccc.o "//'sys1.sieasid(iewbndd)'"

export LIBPATH=./
./a.out ./a.out # this works proving that LIBPATH setup is not required
```

24.3.2 Installation

To use this function, you must rebind your applications to use DLLs.

24.3.3 Migration and coexistence considerations

With this support, UNIX file system DLLs in /usr/include now have hard links to all uppercase names that match the MVS data set members:

- iewbndd.so points to IEWBNDD
- iewbnddx.so points to IEWBNDDX

When binding with the side files from SYS1, SIEASID, if the application is run with `POSIX(ON)`, the API DLL binder will be found in the z/OS UNIX file system.

Attention: This function can be rolled back into z/OS V1.13 with APAR OA39387 PTF UA65826, but it does not add the UNIX links.

24.4 LONGPARM option

The `LONGPARM` option indicates whether the program supports a parameter longer than 100 bytes. This applies mainly to programs that are invoked by using a JCL `EXEC` statement or a z/OS UNIX EXECMVS callable service. `LONGPARM` or `LONGPARM=YES` specifies that the program can accept a parameter string of more than 100 bytes. In this case, an appropriate directory entry bit is turned on. The system checks for this attribute only when the program is being invoked with a parameter string of more than 100 bytes and the program is APF-authorized. In this case, if the `LONGPARM` attribute is not set to `ON`, the system fails the invocation.

24.4.1 Invocation and use

The `PARMDD` change allows parameters of more than 100 characters. The purpose of `LONGPARM` is to provide a way to identify programs that support more than 100 characters for a parameter. It also provides a way to avoid any potential damage if a user passes more than 100 characters to an authorized program.

The `LONGPARM` binder option can be specified anywhere that a binder option can be specified. These are the options:

```
LONGPARM | LONGPARM=YES | LONGPARM=NO | NOLONGPARM
```

The binder listing in the Processing Options and the Save Module Attributes shows the `LONGPARM` value. AMBLIST output shows the `LONGPARM` value for the module.

If a parameter string of more than 100 characters is passed to an authorized program and the LONGPARM value is not set to `YES`, the program abends with an S306 and RC=44.

The `LONGPARM` option does not affect the invocation of a non-authorized program. The program starts execution regardless of the `LONGPARM` setting and length of the parameter string.

24.4.2 Migration and coexistence

Toleration for modules that are bound with `LONGPARM` is delivered for previous z/OS releases with APAR OA41887.

24.5 Documentation and reference

See the following IBM resources for more information and documentation:

- High Level Assembler (HLASM) and Toolkit Feature web page

  `http://www.ibm.com/software/awdtools/hlasm/library.html`
- Documentation in the z/OS MVS page in the z/OS 2.1.0 IBM Knowledge Center:

  `http://ibm.co/1suvDib`
  - *z/OS V2R1 MVS Diagnosis: Tools and Service Aids*, GA32-0905
  - *z/OS V2R1 MVS System Messages, Vol 8 (IEF-IGD)*, SA38-0675
  - *z/OS MVS Program Management: Advanced Facilities*, SA23-1392
  - *z/OS MVS Program Management: User's Guide and Reference*, SA23-1393

25

z/OS XL C/C++

This chapter describes the enhancements, functions, and features that were introduced for z/OS XL C/C++ in z/OS Version 2 Release 1 (2.1).

- 25.1, “Architecture use” on page 316
- 25.2, “Command line include files” on page 325
- 25.3, “Metal C: SYSSTATE macro enhancements” on page 325
- 25.4, “Metal C: User-nominated “main” function” on page 326
- 25.5, “Hardware use and tuning” on page 327
- 25.6, “Decimal floating point and zoned conversions” on page 330
- 25.7, “Expected path selection” on page 330
- 25.8, “Packed decimal built-in functions” on page 331
- 25.9, “[NO]THREADED option” on page 331
- 25.10, “Metal C: AMODE-switching support at IPA LINK” on page 332
- 25.11, “DEBUG(LEVEL) option to debug optimized code” on page 333
- 25.12, “Debugging” on page 333
- 25.13, “Documentation and reference” on page 335

© Copyright IBM Corp. 2014. All rights reserved.

25.1 Architecture use

In z/OS 2.1, there is a TARGET option to allow targeting previous releases of the OS. This allows a compiler on a later OS level to generate programs for an earlier z/OS level.

Note: The Architecture Level Set introduced in z/OS 2.1 causes the default architecture level for the compiler option ARCH to be 7. Targeting an earlier release changes the default ARCH level back to 5.

With the TARGET option, you can specify the runtime environment and release for your program's object module that z/OS XL C/C++ generates. This enables you to generate code that is downward-compatible with earlier levels of the operating system while preventing use of library functions not available on the targeted release. You can compile and link an application on a later-level system and run the application on an earlier-level system. The compiler generates a comment that indicates the value of TARGET in your object module to aid you in diagnosing problems in your program.

To use the TARGET option, select a runtime environment of either IBM Language Environment or IBM Information Management System (IMS™). Then, select the release that you want, for example, zOSV1R13 (1.13). If you do not select a runtime environment or release, the compiler uses the default of TARGET(LE, zOSV2R1). TARGET() generates object code to run under the runtime Language Environment. It is the same as TARGET(LE,CURRENT).

25.1.1 Invocation and use

The suboptions can be used by adding the TARGET option to your compile step. You can use the TARGET option to compile your programs with versions of the compiler and target previous releases of z/OS. This is helpful to prevent migration problems. Figure 25-1 shows the syntax of the compiler TARGET option.

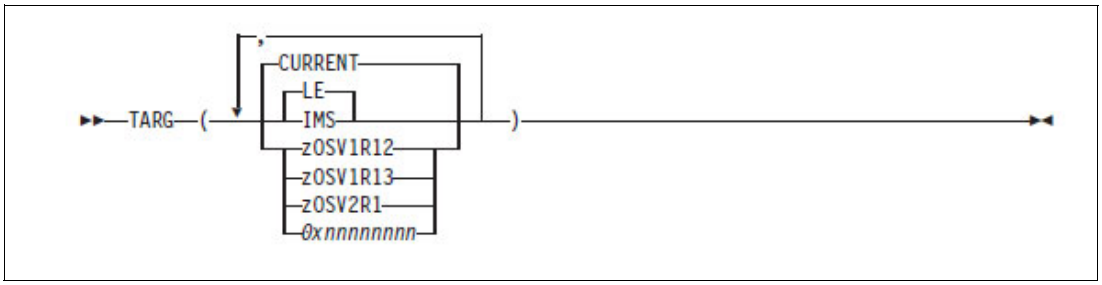


Figure 25-1 Syntax of the TARGET option

The first parameter of TARGET is explained in Table 25-1, which describes the runtime target environment.

Table 25-1 Suboptions for a runtime environment

Option	Description
LE	Generates object code to run under the Language Environment runtime environment. This is the default.

Option	Description
IMS	Generates object code to run under the Information Management System (IMS) subsystem. If you are compiling the main program, you must also specify the PLIST(OS) option. TARGET(IMS) is not supported with LP64.

The second parameter of the TARGET option describes the target release at run time, as Table 25-2 shows.

Table 25-2 Suboptions for the release at program run time

Option	Description
CURRENT	Generates object code that runs under the same version of z/OS that includes the compiler. Because the compiler is included with z/OS 2.1, TARGET(CURRENT) is the same as TARGET(zOSV2R1). This is the default. Be sure to see 25.1.2, “Interactions and dependencies” on page 318”. To make full use of the latest binder features, you need to explicitly specify the COMPAT binder option. The binder default value for this option is MIN, so the binder uses only the minimal set of features that are required for the program that is being processed.
zOSV1R12	Generates object code to run under z/OS Version 1 Release 12 (1.12) and subsequent releases.
zOSV1R13	Generates object code to run under z/OS 1.13 and subsequent releases.
zOSV2R1	Generates object code to run under z/OS 2.1 and subsequent releases.
0xnnnnnnnn	An eight-digit hexadecimal literal string that specifies an operating system level. This string is intended for library providers and vendors to test header files on future releases and is an advanced feature. Most applications should use the other release suboptions. The layout of this literal is the same as the __TARGET_LIB__ macro.

Example 25-1 demonstrates the use of the TARGET option.

Example 25-1 Sample C Program

```
#include <stdio.h>
int main(void) {
  printf(“Arch level is: %d\n”, __ARCH__);
  return 0;
}
```

The sample program is compiled with two different TARGET options as you can see in Figure 25-2.

```
with default TARGET:  xlc arch_check.c
with specific TARGET: xlc -qtarget=zosv1r13 arch_check.c
```

Figure 25-2 possible TARGET options

At compile time, you can see in Figure 25-3 the different outputs.

```
Output with default TARGET: Arch level is: 7
Output with specific TARGET:Arch level is: 5
```

Figure 25-3 Possible outputs

25.1.2 Interactions and dependencies

To make full use of the latest binder features, you need to explicitly specify the COMPAT binder option. The binder default value for this option is MIN, so the binder uses only the minimal set of features that are required to satisfy the program that is being processed.

25.1.3 C11 and C11++ feature enhancements

The ISO C standard was updated in 2011. In z/OS 2.1, a language level (EXTC1X) enables all of the features as they are available. This allows programmers to use the simple language level to take advantage of most of the language features and extensions. The EXTC1X language level was added to the existing EXTC99 language level.

Example 25-2 demonstrates this support.

Example 25-2 EXTC1X support

```
struct S {
  struct {
    int a;
  };
} s;
int main(void) {
    s.a = 55;
    return s.a;
}
```

Figure 25-4 shows the resulting compilation and use of the sample program.

```
> xlc -qlanglvl=extc1x anon_struct.c
> ./a.out
> echo $?
55
```

Figure 25-4 Output of the EXTC1X program

25.1.4 Complex type initialization

In current releases of IBM XL C, compiler complex type objects cannot be initialized with an infinity or NaN imaginary part in any general way. z/OS 2.1 includes macros that are suitable for static initialization and can create such values in an intuitive way for all complex types. The macros take two arguments, one for the real part, the other for the imaginary part. This allows easy and intuitive creation of complex values of the form `x + yi`. where `x` and `y` are any floating point values, including infinities and NaNs.

Invocation and use of complex type initialization

Figure 25-5 shows a sample C program that uses the previous C99 standard to calculate large values.

```
#include <stdio.h>
double _Complex big_value = (1.0/0.0) + (1.0/0.0) * __I;
int main(void) {
   float _Complex nan_value = 9.5 + (0.0/0.0) * __I;
   printf("Big value is %e + %e*i\n", __real__(big_value),
__imag__(big_value));
   printf("NaN value is %e + %e*i\n", __real__(nan_value),
__imag__(nan_value));
return 55;
}
```

Figure 25-5 C99 source code sample

Figure 25-6 shows the output of the C99 standard C program.

```
> xlc -qfloat=ieee -qflag=w complexValues.c
> ./a.out
Big value is NaNQ(1) + INF*i
NaN value is NaNQ(1) + NaNQ(1)*i
```

Figure 25-6 Sample C99 output

The next example, Figure 25-7, demonstrates the use of the C11 standard and complex numbers. The main differences are using of the modified `complex.h` and the compiler option, `-qlanglvl=extc1x`.

```
include <stdio.h>
#include <complex.h>
double _Complex big_value = CMPLX(1.0/0.0, 1.0/0.0);
int main(void) {
   float _Complex nan_value = CMPLXF(9.5, 0.0/0.0);
   printf("Big value is %e + %e*i\n", __real__(big_value),
__imag__(big_value));
   printf("NaN value is %e + %e*i\n", __real__(nan_value),
   __imag__(nan_value));
return 55;
}
```

Figure 25-7 C11 source program

The output of the C11 demonstration is shown in Figure 25-8.

```
> xlc -qlanglvl=extc1x -qfloat=ieee -qflag=w complexValues.c
> ./a.out
Big value is INF + INF*i
NaN value is 9.500000e+00 + NaNQ(1)*i
```

Figure 25-8 C11 sample output

25.1.5 C11: Generic-type generics

Typically, there is a single action that can be applied to several different types (for example, add two numbers together), but remembering each version of the action for each type is troublesome. Calling the wrong version can result in subtle errors. Starting with z/OS 2.1, you can create a single macro that represents the action that resolves to the actual function, based on the argument types. This allows for type-based selection for object manipulation, in general, which enables the programmer to use a simple, single name in code for a type of action across various types.

Invocation and use

The code that is shown in Figure 25-9 allows using `cbrt(<some object>)` to get the cube root for `<some object>` that is a “long double,” double, float type if the `cbrtl`, `cbrt`, or `cbrtf` functions are defined.

```
#define cbrt(X) _Generic((X), \
long double: cbrtl, \
default: cbrt, \
float: cbrtf \
)(X)
```

Figure 25-9 Sample for generic type

Note: The C99 type generic functions are still present and do not need to be redefined.

25.1.6 C11: Function never returns specifier

At the moment function declarations cannot give specialized information to other programmers or the compiler, missing some optimization opportunities. Specifically, a function that never returns. z/OS 2.1 adds the `_Noreturn` keyword to let the compiler and programmers know that the function declared does not return to the caller. This allows for a straightforward means of informing users of the function that it does not return, making runtime performance improvements.

Invocation and use

Figure 25-10 on page 321 shows the use of the `_Noreturn` keyword, which indicates no return of the function.

```
void _Noreturn end_program(char error_type) {
  if (error_type == 'E') {
   exit(16);
      } else {
        exit(0);
      }
}
int main(int argc, char** argv) {
  if (argc < 5) {
    end_program('E');
    // Any code here is invalid and will never be run.
  }
  return 55;
}
```

Figure 25-10 Sample program for using _Noreturn

Note: Do not write any code immediately after a call to such a function.

The output is shown in Figure 25-11. As you can see, the return code is set to 16 rather than 55.

```
> xlc -qlanglvl=extc1x aborter.c
> ./a.out
> echo $?
16
```

Figure 25-11 Sample output with _Noreturn

25.1.7 C11: Static assertions

Assertions are hard to create at compile time and do not give meaningful messages. With z/OS 2.1, you can create a static assertion mechanism to allow compile-time assertions. This allows the programmer to specify what message to emit if the assertion is violated. Compared to runtime assertions, potential errors can be found earlier in the development process.

Invocation and use

Figure 25-12 is an example of using station assertions.

```
#include <assert.h>
#include <stdlib.h>
static_assert(sizeof(long) == 4, "Not in 32-bit mode!");
int main(void) {
  long* buffer_of_10_longs = (long*)malloc(10 * 4); // Could be an external call
  for (int i = 0; i < 10; i++) {
       buffer_of_10_longs[i] = i;
  }
  return (int)buffer_of_10_longs[8] + 47;
}
```

Figure 25-12 Sample of static assertions

When we try to compile that source in 64-bit mode, as shown in Figure 25-13, a CCN3865 error occurs. But with 32-bit, as Figure 25-14 shows, everything works fine.

```
> xlc -qlanglvl=extc1x -q64 longBuffer.c
ERROR CCN3865 ./longBuffer.c:3 Not in 32-bit mode!
CCN0793(I) Compilation failed for file ./longBuffer.c. Object file not created.
```

Figure 25-13 Compile static assertions in 64-bit mode

```
> xlc -qlanglvl=extc1x -q32 longBuffer.c
```

Figure 25-14 Compile Static assertions in 32-bit mode

25.1.8 C++11: Explicit conversion operators

With z/OS 2.1, you can specify the *explicit* keyword on a user-defined conversion operator. Then, programmers can specify that user-defined conversion operators should be used only when explicit conversion is required. Explicit conversion operators generally enable writing robust classes that are less prone to unintended implicit conversion and ambiguity errors. Incorrect use of user-defined operators is rejected at compile time to prevent running faulty programs.

Invocation and use

Figure 25-15 shows an example of using explicit conversion operators.

```
#include<iostream>
template <class T> struct Ptr {
 Ptr() : rawptr_(0) { }
 Ptr(T* ptr) : rawptr_(ptr) { }
 operator bool() const { return rawptr_ != 0; }
 T * rawptr_;
};

int main() {
 int var1, var2;
 Ptr<int> ptr1( &var1 ), ptr2( &var2 );
 if( ptr1 ) // explicit conversion accepted
  return 66;
 std::cout << "ptr1 + ptr2 = " << ptr1 + ptr2 << std::endl; // warning
 if (ptr1 == ptr2) // warning
     std::cout << "does both pointers are really equal?" << std::endl;
}
```

Figure 25-15 Sample of using explicit conversion operators

The compiler output is shown in Figure 25-16 on page 323. The `if( ptr1 )` statement in line 13 was accepted because an explicit Boolean conversion is required. The `ptr1 + ptr2` statement in line 15 gets a warning because Boolean conversion is implicit. You need to define an `operator+` to correct this warning. Also, the statement on line 16 gets a warning and requires an `operator==` to be correct.

```
> xlC -qlanglvl=explicitconversionoperators -+ a.c
> "./a.c", line 15.40: CCN5218 (S) The call does not match any parameter list for
"operator+".
> "./a.c", line 15.40: CCN6283 (I) "builtin operator+(int, int)" is not a viable
candidate.
> "./a.c", line 16.11: CCN5218 (S) The call does not match any parameter list for
"operator==".
> "./a.c", line 16.11: CCN6283 (I) "builtin operator==(int, int)" is not a viable
candidate.
```

Figure 25-16 Compiler output from using explicit conversion operators

25.1.9 C++11: Scoped enums

Implicit integral promotion of enumeration type can lead to conversions that do not make sense. Enumerations that are defined in a global namespace leaks all enumerator declarations into a global namespace and can create name collisions. The enumeration types cannot be used reliably in structs. This makes it impossible to make a forward declaration.

The scoped enum with the syntax and extension to regular enums allows several improvements:

- Better type safety without manual workarounds
- Removal of inconsistencies with traditional C++ enumerations
- Faster compilation
- Fewer dependencies with the introduction of forward declarations of enumerations

Invocation and use

Figure 25-17 shows how to use a scoped enum with syntax.

```
enum class Colour { orange, green, purple };
enum struct Fruit { apple, pear, grape };
void foo(int) {}
void foo(Colour) {}
void foo(Fruit) {}
int main()
{
foo(green); // error, green is not introduced into global scope
foo (Colour::green); //ok
foo(Fruit::apple); //calls foo(Fruit)
int i = Colour::orange; // error, no conversion from Colour to int
}
```

Figure 25-17 Sample of using scoped enumerations

25.1.10 C++11: Generalized constant expression

The compile-time evaluation is limited to built-in types in prior releases of XLC Compiler. The current Implementation of standard template numeric_limits cannot be used in an array that is bound, a case expression, a non-type template argument, or a bit field length where 'const' expressions are required.

Starting with z/OS 2.1, a keyword, `constexpr`, has been added to the C++ language. Compile-time evaluation was extended to any abstract types and operators that can be used when constant expressions are required. This improves type safety and portability of code that requires compile-time evaluation.

However, generalized constant expression in V2.1 lacks the following functions:

- Constant expression evaluations that trigger template instantiations
- Graceful handling of infinite recursion within a constant expression
- Pointer arithmetic expression within constant expressions

Invocation and use

The support can be used by `-qlanglvl=[no]constexpr` or `-qlanglvl=extended0x`. Sample code is shown in Figure 25-18.

```
template <class C> struct my_numeric_limits
{
    constexpr static C max();
    constexpr static C min();
};

template <> constexpr int my_numeric_limits<int>::max() { return (1 << sizeof(int)*8-
1) - 1; }

template <> constexpr int my_numeric_limits<int>::min() { return 1 << sizeof(int)*8 -
1; }

template <> constexpr char my_numeric_limits<char>::max() { return (1 << 8-1) - 1; }

constexpr int f() { return my_numeric_limits<int>::max() / 2; }

enum E { min = my_numeric_limits<int>::min(), median = f(), max =
my_numeric_limits<int>::max() };
template <class C> struct S {
    C container_of_all_objects[my_numeric_limits<C>::max()];
};
S<char> s;
```

Figure 25-18 Sample code of using generalized constant expression

Migration and coexistence considerations

Any existing source code that is compiled with the `complex.h` header, uses the EXTC1X language level, and has identifiers that are named CMPLXF, CMPLX, or CMPLXL changes behavior.

Note: Because the language level was introduced in this release, existing builds are not affected. Only if the language level is used is this a potential migration issue.

The explicit keyword was supported before, but it was allowed only on constructors. When this keyword was used in another context, the following error shown in Example 25-3 on page 325 was generated. In V2.1, the additional information diagnostic was added when the keyword is used on user-defined conversion operators.

Example 25-3 Explicit keyword error notice

```
CCN5111 (S) The "explicit" specifier must be applied only to declarations of
constructors within a class declaration.
CCN8955 (I) SUGGESTION: Try using "LANGLVL(EXPLICITCONVERSIONOPERATORS)". It looks like
you are trying to use the associated C++0x feature.
```

The `constexpr` keyword is treated as an identifier when it is used without enabling generalized constant expression. The `-qwarn0x` compiler option issues a warning message about the potential conflict with C++11 features.

25.2 Command line include files

A common header file is often desirable to set up program defaults and targets, but changing the existing source code might not be something that you want or even be possible. With z/OS 2.1, a compiler option, `-qinclude=<file_name>`, is introduced. This is an equivalent to the `#include file_name` directive that is specified at the beginning of each source file. The option creates the opportunity to have different builds that use the same source files. This file can encapsulate any macros or definitions that are required by most or all the source files, which do not need to be modified individually.

25.2.1 Invocation and use

Figure 25-19 shows C source code examples. This simple code demonstrates the use of the option.

```
<t.h>
#define STRING "hello world"
</t.h>

<t.c>
int main () {
  printf ("%s\n", STRING);
  return 0;
}
</t.c>
```

Figure 25-19 Sample source code for include files

To compile that source code, you must use the compiler option, `-qinclude`, which is shown in Figure 25-20.

```
> xlc t.c -qinclude=stdio.h -qinclude=t.h
> ./a.out
> hello world
```

Figure 25-20 Use of the compiler option

25.3 Metal C: SYSSTATE macro enhancements

In the current release of the Metal C Implementation the SYSSTATE Assembler, a macro that is automatically generated by the compiler does not have the OSREL parameter nor the

ASCENV parameter specified. z/OS 2.1 provides a option, SYSSTATE, which allows the user to add the **OSREL** parameter with the desired value and to have the **ASCENV** parameter generated according to the ASC mode of the function. Metal C users now have a way to tune the SYSSTATE Assembler macro that is generated by the compiler.

25.3.1 Invocation and use

Figure 25-21 shows the syntax of the Metal C SYSSTATE option, where:

`OSREL=NONE | ZOSVnRm OSREL` — Provides the value for the **OSREL** parameter on the SYSSTATE macro that is generated by the compiler. The value that is provided must be in the form of ZOSVnRm, as described in the *z/OS MVS Programming: Assembler Services Reference IAR-XCT*, SA23-1370. The default is NONE, so no **OSREL** parameter appears on the SYSSTATE macro.

`ASCENV | NOASCENV ASCENV` — Indicates to the compiler to automatically generate more SYSSTATE macros with the **ASCENV** parameter to reflect the ASC mode of the function. The default is NOASCENV, so no **ASCENV** parameter appears on the SYSSTATE macro.

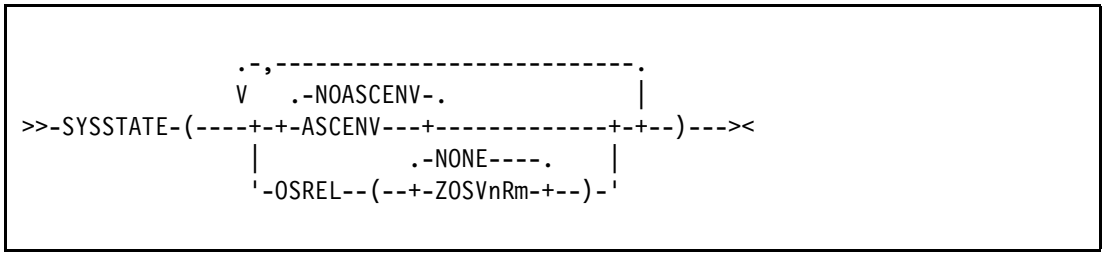


Figure 25-21 SYSSTATE syntax

Note: The Metal C system default is SYSSTATE(NOASCENV,OSREL(NONE)).

25.3.2 Migration and coexistence considerations

The SYSSTATE option is meaningful only when the C compiler is generating High Level Assembler (HLASM) source code in combination with the GENASM option.

25.4 Metal C: User-nominated “main” function

Currently, when a Metal C program is built with the RENT option, it needs a “main” function to anchor the writable static area (WSA) creation process. However, a Metal C program might not have a function called “main” as the entry point, so it does not have the opportunity to be built with the RENT option. With the release, you can use the #pragma map directive to associate the “main” function with an alternative external name. Therefore, a Metal C program can now have an alternative entry point name for the “main” function, yet maintain all of the characteristics of that function.

25.4.1 Invocation and use

If you want to call your Metal C “main” function “ANEWMAIN,” for example, you can simply add this directive in your source file where “main” is defined, as you see in Figure 25-22.

```
#pragma map(main, "ANEWMAIN")
void dosomething(char *);
int main(int argc, char *argv[]) {
   int i;
   for (i=1; i<argc; i++) {
       dosomething(argv[i]);
   }
   return 0;
}
```

Figure 25-22 Sample of using the Main

The entry point name in the generated code is ANEWMAIN. When you link your program, you need to use the -e option to tell the binder that the entry point name is ANEWMAIN. For example:

```
/bin/ld -o a.out a.o -e ANEWMAIN
```

25.5 Hardware use and tuning

The IBM zEnterprise EC12 (zEC12) server is the first general-purpose IBM server to incorporate transactional memory technology. In zEC12, this technology is adapted for software to better support concurrent operations that use a shared set of data, such as financial institutions that process transactions for the same set of accounts.

The zEC12’s Transactional Execution (TX) facility is an architectural framework for lockless interlocked execution of a block of code called a transaction. In this context, a *transaction* is a segment of code that appears to other CPUs to run atomically. Other processors in the system detect either all or none of the storage updates made by the transaction.

Table 25-3 lists the built-in functions introduced for the XL C/C++ compiler. Transactions are bound by TBEGIN and TEND instructions. A storage conflict is detected by the hardware if another CPU updates storage used that is by the transaction. The conflict triggers a transaction to abort and rolls back the hardware state (general purpose registers and storage) to what was observed at the TBEGIN instruction. Program execution is also rolled back to the instruction immediately following the TBEGIN, with transactional execution now disabled. A transaction failure condition code is set so that program flow can be diverted to a transaction failure handler. The transaction failure handler can choose to retry the transaction or perform traditional coarse-locking to ensure progress of the program.

Table 25-3 Transactional Execution (TX) facility built-in functions

Intrinsic prototype	Description	Pseudo assembly
`long __TM_simple_begin()`	Starts a light TBEGIN, returns the condition code set by TBEGIN	`TBEGIN 0,0xFF`
`long __TM_begin(void* const TM_buff)`	Starts a TBEGIN with a buffer where failure information will be stored, returns the condition code set by TBEGIN	`TBEGIN 0(r1),0xFF`

Intrinsic prototype	Description	Pseudo assembly
`long __TM_end()`	Ends a transaction	`TEND`
`void __TM_abort()`	Aborts a transaction with abort code 256	`TABORT 256`
`void __TM_named_abort(unsigned char const code)`	Aborts a transaction with abort code 256+code	`TABORT 256+code`
`void __TM_non_transactional_store ( void* const addr, long long const value)`	Stores the value in the address provided, non-transactionally	`NTSTG&n ; r0,0(,r1)`

`long __TM_nesting_depth(void* const TM_buff)`	Returns the current nesting depth or, if not in transaction mode, the most recent nesting depth if transaction aborted or 0 if committed	`LLH r0,buf(,r2,14)`
		`LTR r0,r0`
		`JH @L1`
		`ETND r0`
		`J @rest_of_code`
		`@L1 DS 0H`
		`LLH r0,buf(,r2,6)`
		`@rest_of_code`
		`DS ...`

Additional architectural functions in the zEC12 Transactional Execution facility include the built-in failure reasons shown in Table 25-4.

Table 25-4 Failure reasons built in

Intrinsic prototype	Description	Pseudo assembly

`long __TM_is_user_abort(void* const TM_buff)`	Returns 1 if the abort is due to a user-specified TABORT	`LLH r0,buf(,r2,14)`
		`NILF r0,0x1FF`
		`LTR r0,r0`
		`LA r0,0`
		`JE *+8`
		`LA r0,1`

`long __TM_is_named_user_abort(void* const TM_buff, unsigned char* code)`	Returns 1 if the abort is due to a user-specified TABORT, and the value returned in parameter code is the code passed to the TABORT instruction	`LLH r0,buf(,r2,14)`
		`NILF r0,0x1FF`
		`LTR r0,r0`
		`LA r0,0`
		`JE *+8`
		`LA r0,1`
		`NILF r0,0xFF`
		`STC r0,code(,r1,0)`

`long __TM_is_illegal(void* const TM_buff)`	Returns 1 if the transaction aborted because of an illegal instruction or results in a program interrupt if the abort code is 4 or 11	`LLH r0,buf(,r2,14)`
		`LA r1,0`
		`CHI r0,H'11'`
		`JE @is_illegal`
		`CHI r0,H'4'`
		`JNE @not_illegal`
		`@is_illegal DS 0H`
		`LA r1,1`
		`@not_illegal DS 0H`

Intrinsic prototype	Description	Pseudo assembly

`long __TM_is_footprint_exceeded( void* const TM_buff)`	Returns 1 if the transaction aborted because of touching the maximum number of cache lines, abort code 7 or 8	`LLH r0,buf(,r2,14)`
		`LA r1,0`
		`CHI r0,H'7'`
		`JE @is_large`
		`CHI r0,H'8'`
		`JNE @not_large`
		`@is_large DS 0H`
		`LA r1,1`
		`@not_large DS 0H`

`long __TM_is_nested_too_deep(voi d* const TM_buff)`	Returns 1 if the transaction aborted because of trying to exceed the maximum nesting depth, abort code 13	`LLH r0,buf(,r2,14)`
		`CHI r0,H'13'`
		`LA r15,1`
		`JE *+8`
		`LA r15,0`

`long __TM_is_conflict(void* const TM_buff)`	Returns 1 if the transaction aborted because of a conflict, abort code 9 or 10	`LLH r0,buff(,r2,14)`
		`LA r1,0`
		`CHI r0,H'9'`
		`JE @is_conf`
		`CHI r0,H'10'`
		`JNE @no_conf`
		`@is_conf DS 0H`
		`LA r0,1`
		`@no_conf DS 0H`

`long __TM_is_failure_persistent( long const result)`	Returns 1 if the transaction aborted because of a reason that is persistent, condition code set is 3	`L r0,rc(,r13,152)`
		`CHI r0,H'3'`
		`LA r0,1`
		`JE *+8`
		`LA r0,0`

`long __TM_failure_address(void* const TM_Buff)`	Returns the code address at which the most recent transaction aborted and points to the instruction following the failure	`L r0,buf(,r2,28)`

`long long __TM_failure_code(void* const TM_Buff)`	Returns the raw failure code, so programmer needs to read the hardware specification document to understand	`L r1,buf(,r2,8)`
		`L r0,buf(,r2,12)`
		`ST r1,rc(,r13,152)`
		`ST r0,rc(,r13,156)`

25.5.1 Invocation and use

To use the TX function of zEC12, you must use the functions in your code that are listed in Table 25-3 on page 327 and Table 25-4 on page 328.

25.5.2 Interactions and dependencies

You must compile your code with the `ARCH(10)` option. This function requires an zEC12 server.

25.6 Decimal floating point and zoned conversions

The zEC12 hardware introduces the Decimal-Floating-Point (DPF) Zoned-Conversion Facility, which provides instructions for conversions between DFP and Zoned types. Provide hardware built-in functions for accessing the DFP-Zoned conversion instructions that are introduced by IBM zEnterprise EC12. That leads to hardware-performed DFP-Zoned type conversions that can improve application performance.

25.6.1 Invocation and use

When compiled with the `DFP` and `ARCH(10)` options, the following hardware built-in functions shown in Figure 25-23 and Figure 25-24 are available.

```
_Decimal128 __cxzt(void* source, unsigned char length, const unsigned char
mask);
_Decimal64 __cdzt(void* source, unsigned char length, const unsigned char
mask);
```

Figure 25-23 Functions for Zoned-to-DFP conversions

```
int __czxt(_Decimal128 source, void* result, unsigned char length, const
unsigned char mask);
int __czdt(_Decimal64 source, void* result, unsigned char length, const
unsigned char mask);
```

Figure 25-24 Functions for DFP-to-Zoned conversions

25.7 Expected path selection

The compiler might not know what an expression normally evaluates to, but the programmer might know. If there is a way for the user to convey this information to the compiler, the compiler might be able to use this knowledge to direct optimization. A built-in function, `__builtin_expect`, is added in z/OS 2.1 as a way to provide additional information to the compiler that can be used for optimization. You might get improved runtime performance as a result.

25.7.1 Invocation and use

In the sample code in Figure 25-25, we expect that x is equal to 0, and we will not execute the statement for this branch frequently.

```
if(__builtin_expect(x, 0)) { error();
...
}
```

Figure 25-25 Sample of __builtin_expect call

25.8 Packed decimal built-in functions

In the previous implementation of the C++ compiler, there is no support for packed decimal intrinsic types. Also, decimal instructions were not normally generated by XL compilers. V2.1 of IBM XL C++ includes six built-in functions that are prototyped in `builtins.h`:

CP	Compare Decimal
AP	Add Decimal
SP	Subtract Decimal
MP	Multiply Decimal
DP	Divide Decimal
SRP	Shift and Round Decimal

Now, C++ and Metal C users can directly use packed-decimal instructions similar to assembler programs.

25.8.1 Invocation and use

To use the C++ function, you must include `builtins.h`, as shown in Figure 25-26. The sample program uses the Compare Packed function to demonstrate the use of the packed decimal functions.

```
#include <builtins.h>
int main() {
unsigned char dec1[4] = { 0x17, 0x25, 0x35, 0x6D};
unsigned char dec2[3] = { 0x72, 0x14, 0x2D};

int c = __cp(dec1, 3, dec2, 2); // the length field coded is the actual
length -1

if (c == 1) // indicates the first operand is less than the second
 return 55; //expected condition
else
 return 66;
}
```

Figure 25-26 Use of decimal built-in functions

25.9 [NO]THREADED option

The current Implementation of the compiler optimizer must assume that the user application is multithreaded, which inhibits it from making transformations that are not threadsafe, even though those could speed up the single-threaded applications. That is the reason for the implementation of a `NOTHREADED` option, so that the user can assert that the application is single-threaded. This might lead to performance benefits at both compile time and run time.

25.9.1 Invocation and use

To maintain thread safety, always specify the `THREADED` option when compiling or linking multithreaded applications. This option does not make code threadsafe, but it ensures that code that is already threadsafe remains so after compilation and linking. It also ensures that all optimizations are threadsafe. Specifying the `NOTHREADED` option enables the optimizers to

perform non-threadsafe transformations for single-threaded programs. If you specify the NOTHREADED option with the SMP option, a warning message is issued, and the NOTHREADED option is ignored.

For single-threaded applications, you would instruct the compiler like this:

```
xlc -qnothreaded single-threaded.c
```

For multi-threaded applications you can choose either of these options:

```
xlc -qthreaded multi-threaded.c
```

or

```
xlc multi-threaded.c
```

Note: The THREADED compiler is the default.

25.10 Metal C: AMODE-switching support at IPA LINK

You might need to switch addressing mode (AMODE) between programs. The default AMODE that is assigned by the XL C compiler is based on the LP64 compiler option or the ILP32 compiler option. AMODE 64 is assigned when LP64 is specified, and AMODE 31 is assigned when ILP32 is specified.

The Metal option enables the XL C compiler to generate code for calling an external function with an AMODE that is different from the default AMODE. This capability supports the creation of Metal C programs that require AMODE switching across functions. The resulting compiler-generated code follows the linkage conventions that are expected by the called function, particularly in the areas of save area format and the parameter list width. You can use the amode31 function attribute to mark an AMODE 31 function or the amode64 function attribute to mark an AMODE 64 function in your source files. The __ptr64 qualifier can be used when the Metal option is specified so that a 64-bit pointer can be handled by an AMODE 31 function without dereferencing it.

For more information about the amode31 function attribute, amode64 function attribute, and the __ptr64 qualifier, see *z/OS XL C/C++ Language Reference*, SC14-7308.

z/OS Metal C Programming Guide and Reference, SC14-7313 describes the impact of AMODE switching across functions on the save area chain in the user-supplied prolog or epilogue code and the restrictions that apply to AMODE switching across functions.

25.10.1 Invocation and use

Figure 25-27 shows how to compile two source files, one with AMODE 31 and one with AMODE64. Both are linked with Interprocedural Analysis (IPA) optimization, which is now allowed.

```
> xlc -qmetal -q32 -qipa -c 32.c
> xlc -qmetal -q64 -qipa -c 64.c
> xlc -qmetal -qipa -S 32.o 64.o
```

Figure 25-27 AMODE switch with IPA sample

25.11 DEBUG(LEVEL) option to debug optimized code

It is hard to debug optimized code because the debugger does not identify where to find the value of a variable. For example, the value can be in a register, not in memory. The code-generated ordering might not match the source code ordering.

The compiler creates different levels of snapshots of objects at selected source locations and makes the program state available to the debugging session at the selected source locations. When stopping at the snapshot points, the debugger should be able to retrieve the correct values of variables. The granularity of the snapshot points is controlled by the DEBUG(LEVEL) suboption.

25.11.1 Invocation and use

The DWARF suboption produces debug information in the DWARF Version 4 debugging information format, which is stored in the file that is specified by the FILE suboption. This is the only format supported when LP64 or Metal is specified.

Apply to DEBUG(FORMAT(DWARF)) + O2. The debug ability support for optimization levels that are higher than -O2 is limited. O2 or OPTIMIZE(2) indicates that global optimizations are to be performed.

Note: The size of your functions, the complexity of your code, the coding style, and support of the ISO standard can affect the global optimization of your program. You might need significant additional memory to compile at this optimization level.

Figure 25-28 shows how to use this support.

```
> xlc -Wc,"DEBUG(FORMAT(DWARF),LEVEL(8))" -O2 a.c
> xlc -Wc,"DEBUG(FORMAT(DWARF))" -O2 -g8 a.c
```

Figure 25-28 DEBUG(FORMAT(DWARF) option

25.12 Debugging

In V1.13, debug information was added for inline procedures that set entry break points for all inline instances of a procedure. No debug information was provided for the parameters and local variables of the inline instances. The debugger cannot show the value of these objects.

With z/OS 2.1, debug information is provided for parameters and local variables of each inline instance of a procedure. The debugger can show the values of the parameters and locals of an inline instance.

When setting a line break point with the **`st at`** command, dbx sets it on the first LNT entry that matches the specified line. This does not work for optimized code when inlining happens. z/OS 2.1 provides a relative statement number for LNT entries on the same line. When debugging, you can set line break points for all LNT entries that match the specified line and have relative statement number turned on (`relstmtno 1`).

25.12.1 Debugging inlined functions

You can debug Dwarf format code by using DEBUG(FORMAT(DWARF)) + O2 for g5, g8, or g9. Debug information is also generated for O3 and higher, but there is no guarantee about the correctness.

```
> xlc -Wc,"DEBUG(FORMAT(DWARF),LEVEL(8))" -Wc,XPLINK -O2 a.c
> xlc -Wc,"DEBUG(FORMAT(DWARF))" -Wc,XPLINK -O2 -g8 a.c
```

Figure 25-29 Example of using DWARF

```
<1>< 353>       DW_TAG_subprogram
                DW_AT_name                      foo
                DW_AT_inline                    DW_INL_declared_inlined
<2>< 370>       DW_TAG_formal_parameter
                DW_AT_name                      input
<2>< 384>       DW_TAG_variable
                DW_AT_name                      aaa
<2>< 488>       DW_TAG_inlined_subroutine
                DW_AT_abstract_origin           <353>
                DW_AT_low_pc                    0xb8
                DW_AT_high_pc                   0xf6
<3>< 505>       DW_TAG_formal_parameter
                DW_AT_abstract_origin           <370>
                DW_AT_location                  DW_OP_breg4+2148
<3>< 514>       DW_TAG_variable
                DW_AT_abstract_origin           <384>
                DW_AT_location                  DW_OP_breg4+2152
```

Figure 25-30 Generated debugging Information

25.12.2 Relative statement number

Figure 25-31 shows part of a C program to demonstrate the debugging information that is provided.

```
…
5 i = 11.0f; j = 55.0f;
…
```

Figure 25-31 Sample source code for debugging

The program must be compiled with the DEBUG option as shown in this example:

```
> xlc -Wc,"DEBUG(FORMAT(DWARF))" b.c
```

Figure 25-32 on page 335 shows the debugging output that is produced by that compiler call. You see the line numbers from the source in Figure 25-31 and the relative offset.

```
Line number information at line program offset 0

<source>             [  row,col][ rel] <pc> <statement or basic block>
itor/test/394873/b.c [    1, 12][   0] 0x98 // begin of prologue
itor/test/394873/b.c [    1, 12][   0] 0xe8 // end of prologue
itor/test/394873/b.c [    5,  5][   1] 0xec // statement [ HOOK ]
itor/test/394873/b.c [    5, 16][   2] 0xf8 // statement [ HOOK ]
```

Figure 25-32 Debugging output

The next enhancement was implemented to show the debug information for all of the variables on the line that is being debugged. This information similar to the debug tool's COBOL automatic monitor support. In the IBM Xpertise Library (XL) compiler, there is now an automonitor table in the Dwarf debug sidefile. Conceptually, it is a table of pairs (line number and list of variables). This provides information for debuggers to easily look up variable values, based on source code lines.

```
.debug_srcattr

                      |ecpileseolemeppedgdbc|
                      |xmgnsiestolimounieela|
            alt  rel  |enmcengghggnipsasncal| .debug_xref
[  row,col] line stmt |ct         vn hbarlnl| DIE  offset [xreflist DIE names]
------------------------+--------------------+--------------------------------
[     5, 5]    0    1 |X...................|          20 DW_TAG_IBM_xreflist_item: i
[    5, 16]    0    2 |X...................|          33 DW_TAG_IBM_xreflist_item:
```

Figure 25-33 Debug automonitor output

25.13 Documentation and reference

For more information, see the documentation and related publications in the IBM Knowledge Center for z/OS 2.0.1:

- *z/OS Common Debug Architecture Library Reference*, SC14-7311

  http://ibm.co/1yiQkAW
- *z/OS Common Debug Architecture User's Guide*, SC14-7310

  http://ibm.co/1oAYVYD
- *z/OS DWARF/ELF Extension Library Reference*, SC14-7312

  http://ibm.co/1nzH2wP
- *z/OS Internet Library*

  http://www.ibm.com/systems/z/os/zos/library/bkserv/index.html
- *z/OS Metal C Programming Guide and Reference*, SC14-7313

  http://ibm.co/Wdrlld
- *z/OS Standard C++ Library Reference*, SC14-7309

  http://ibm.co/1kWQLsf
- *z/OS XL C/C++ Compiler and Runtime Migration Guide for the Application Programmer*, GC14-7306

http://www.ibm.com/support/docview.wss?uid=pub1gc14730600

- *z/OS XL C/C++ Language Reference*, SC14-7308

  http://www.ibm.com/support/docview.wss?uid=swg27036892

- *z/OS XL C/C++ Messages*, GC14-7305

  http://www.ibm.com/support/docview.wss?uid=pub1gc14730500

- *z/OS XL C/C++ Programming Guide*, SC14-7315

  http://ibm.co/1jspfYthttp://ibm.co/1jspfYt

- *z/OS XL C/C++ Runtime Library Reference*, SC14-7314

  http://ibm.co/W4WW8f

- *z/OS XL C/C++ User's Guide*, SC14-7307

  http://ibm.co/U5i6BB

26

Language Environment

This chapter describes the enhancements, functions, and features that were introduced for the Language Environment in z/OS Version 2 Release 1 (2.1).

- 26.1, “Heap overlay tolerance” on page 338
- 26.2, “Remove Language Environment USERMODs options” on page 339
- 26.3, “Runtime system symbol access” on page 341
- 26.4, “AMODE 64 support for 1M and 2G large pages” on page 344
- 26.5, “Nested CEEPIPI environments” on page 346
- 26.6, “BSAM type=blocked support” on page 347
- 26.7, “N1040 functions” on page 348
- 26.8, “Standard I/O extensions” on page 350
- 26.9, “Unicode auto-conversion support” on page 352
- 26.10, “C11 validation Items” on page 352
- 26.11, “C11 alignment-related support” on page 354
- 26.12, “Documentation and reference” on page 355

© Copyright IBM Corp. 2014. All rights reserved.

26.1 Heap overlay tolerance

You might have a poorly written application that causes small overlays in user heap storage but must continue to run until a fix is available. During application testing, the ability to detect these small overlays is needed, even when there is no obvious application failure.

z/OS 2.1 provides a runtime option called HEAPZONES that can be activated to provide a heap check zone for each user heap element. The runtime option can also be used to notify the user of a storage overlay when an element is freed. That allows you to tolerate heap overlays in production environment. Furthermore, you can detect heap overlays during application testing. There are no explicit installation steps to perform to use this function.

The HEAPZONES runtime option controls user heap overlay toleration and checking. A *heap check zone* is an extra piece of storage that is appended to a heap element during a storage allocation request. The size of the check zone is controlled by the runtime option. The runtime option also controls checking and how much diagnostic information is provided when an overlay is found. If checking is requested, the check zone is examined for overlays when the heap element is freed.

Attention: The runtime option affects heap pools as well as user heap. The runtime option cannot be specified at the system or region level.

26.1.1 Invocation and use

Figure 26-1 shows the syntax of the HEAPZONES option.

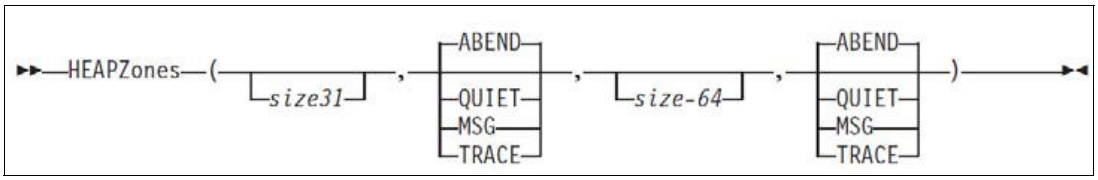


Figure 26-1 HEAPZONES syntax

The first suboption controls the size of the check zone for all user heap storage that is below the 2 G bar. Check zone size is rounded up to the nearest multiple of 8 bytes. The maximum size allowed for a check zone is 1024 bytes.

Attention: Specifying a value of 0 in size31 indicates that no check zone is active.

The second suboption controls the validation of the check zone for all user heap storage that is below the 2 G bar. All available options are described in Table 26-1 on page 339.

The third suboption controls the size of the check zone for all user heap storage which is above the 2G bar. Check zone size is rounded up to the nearest multiple of 8 bytes with a minimum size of 16 bytes. The maximum size allowed for a check zone is 1024 bytes.

Attention: Specifying a value of 0 in size64 indicates no check zone is active.

The fourth suboption controls the validation of the check zone for all user heap storage which is above the 2G bar, as shown previously in Table 26-1 on page 339.

Table 26-1 Validation options for 31 bit and 64 bit

Option	Description
QUIET	No validation is requested.
MSG	Validation when storage freed. Informational message if overlay detected.
TRACE	Specifies that in addition to informational messages, a CEEDUMP containing only a traceback is produced.
ABEND	Validation when storage freed. Informational message and ABEND U4042 with reason code 3 if overlay detected.

26.1.2 Interactions and dependencies

HEAPZONES is a lightweight mechanism that detects heap overlay damage only during the freeing of an element. It looks for damage in the heap check zone of the freed element only.

Attention: Depending on the size of the heap check zone and the number of allocation requests, you might notice a significant amount of extra storage being used by the application. Performance might be affected due to the overhead of examining each heap check zone.

There is another Language Environment runtime check called HEAPCHK to investigate the entire user heap for damage during heap-related calls at a frequency based on the specified settings in the option. Because HEAPCHK traverses the entire user heap, a slowdown in application performance occurs.

When deciding which runtime option is better to use with your application, HEAPZONES or HEAPCHK, consider the differences that are related to performance, storage use, and time of damage detection. Although both runtime options affect performance, an application that chooses HEAPCHK performs slower than an application that chooses HEAPZONES. If storage use is a concern, HEAPCHK does not consume extra amounts of storage in the manner that HEAPZONES does. Determining when heap damage has occurred might be simpler to accomplish if HEAPCHK is chosen because of the frequency and scope of its analysis.

26.2 Remove Language Environment USERMODs options

User modifications (USERMODS) to the Language Environment runtime options for setting installation defaults have been an ongoing source of confusion and errors for IBM clients. Because they were Assembler-based control sections (CSECTs), they were difficult to maintain and update. Updating them at each release of z/OS was also required. Other runtime option mechanisms have made it possible to remove this Language Environment to support in favor of simpler, more user friendly methods.

26.2.1 Interactions and dependencies

The IBM wording of “Installation Default” is renamed to “IBM-Supplied Default.” All of the CEEDOPT, CEECOPT and CELQDOPT samples that were provided in previous releases of z/OS are removed from the SCEESAMP Library, as are the CEEWDOPT, CEEWCOPT, and CEEWQDOP sample jobs. CEERDOPT, CEERCOPT, and CELQRDOP samples are provided for CEEROPT generation.

There was a Health Check called “CEE_USING_LE_PARMLIB” in earlier releases of z/OS to verify that the installation is using the CEEPRMxx member to configure Language Environment runtime options. This Health Check is no longer necessary, so it is not included in z/OS 2.1.

26.2.2 Migration and coexistence considerations

Starting with z/OS 2.1, there is no support for CEECOPT, CEEDOPT, and CEEQDOPT usermods. Therefore, before you migrate to V2.1, you must remove the usermods and migrate your system to the CEEPRMxx PARMLIB support that was introduced in z/OS Version 1.7 (V1R7).

Important: CEEROPT and CEEUOPT support are not affected and remain unchanged.

How to get the CEEPRMXX support

The CEEPRMxx PARMLIB member can have up to three sections, as shown in Table 26-2. In each section, you can code all available options to configure the destination environment.

Table 26-2 Supported sections in the CEEPRMxx PARMLIB member

Section	Explanation
CEECOPT	31-bit IBM CICS® option group
CEEDOPT	31-bit non-CICS option group
CELQDOPT	64-bit options group

To activate a CEEPRMxx PARMLIB member after an IPL of the system, use the **SET CEE=xx** command. In Figure 26-2, you can see the successful activation of the CEEPRM00 PARMLIB member.

```
SET CEE=00
CEE3742I THE SET CEE COMMAND HAS COMPLETED.
```

Figure 26-2 Example of the SET CEE=00 command

After the activation of a CEEPRMxx PARMLIB member, all values replaced come from the member.

To change a parameter in a specified group, use the **SETCEE** command. The command is limited to 126 characters. Figure 26-3 shows the syntax of the command.

```
SETCEE [CEEDOPT,opt,opt,...]
       [CEECOPT,opt,opt,...]
       [CEELQDOPT,opt,opt,...]
```

Figure 26-3 Syntax of the SETCEE command

As Figure 26-4 on page 341 shows, you can set the Language Environment POSIX to ON.

```
SETCEE CEEDOPT,POSIX(ON)
CEE3743I THE SETCEE COMMAND HAS COMPLETED.
```

Figure 26-4 Example of the SETCEE command

To display the current active member and its settings, use the **D CEE** command in this format:

```
D CEE [{,CEEDOPT} ] |,{CEECOPT} |,{CELQDOPT} [,L={a|name|name-a}]
```

If you enter the **D CEE** command only, you will see the active CEEPRMxx PARMLIB member suffixes. Table 26-3 lists descriptions of all valid options for the **D CEE** command.

Table 26-3 D CEE command options

Option	Description
CEEDOPT	Displays all options for the 31-bit CICS option group
CEECOPT	Displays all options for the 31-bit non-CICS option group
CELWDOPT	Displays all options for the 64-bit options group
L=	Specify an output console by name or number for the output area

26.3 Runtime system symbol access

In z/OS 2.1, the IBM JES2 and Scheduler components enhance the use of JCL symbols in JCL, providing an interface to retrieve JCL symbols at run time. A Language Environment callable service is included to provide a function similar to the scheduler service but with less complexity and more usability.

Also, interfaces added to the Language Environment allow an application to retrieve the value of an exported JCL symbol. An application can now retrieve the value of an exported JCL symbol from a high-level language.

26.3.1 Invocation and use

This release of z/OS includes the two C functions that are shown in Table 26-4, which you can use to get information about JCL symbols.

Table 26-4 Function overview

31-bit function	64-bit function
CEEGTJS	__le_ceegtjs

Figure 26-5 shows the calling conventions to use that functions.

```
CEEGTJS(function_code, symbol_name, symbol_value, value_length, fc);

void __le_ceegtjs(_INT4 * function_code,_VSTRING * symbol_name,_CHAR255 *
                  symbol_value,_INT4 * value_length,_FEEDBACK * fc);
```

Figure 26-5 Syntax of CEEGTJS and __le_ceegtjs

Table 26-5 lists parameters for both **CEEGTJS** and **__le_ceegtjs**. These are basically the same with the exception of the feedback code. **CEEGTJS** provides a 12-byte optional feedback code. The fc in **__le_ceegtjs** has a length of 16 bytes.

Table 26-5 Function parameters

Parameter	CEEGTJS	__le_ceegtjs
function_code	A full-word integer that contains the function code. Currently, only a value of 1 is allowed. Retrieve the value and its associated length of an exported JCL symbol.	A full-word integer that contains the function code. Currently, only a value of 1 is allowed. Retrieve the value and its associate length of an exported JCL symbol.
symbol_name	A halfword-length prefixed character string (VSTRING) that represents the name of an exported JCL symbol to be retrieved.	A halfword-length prefixed character string (VSTRING), that represents the name of an exported JCL symbol to be retrieved.
symbol_value	A 255-byte fixed-length string. On return from this service, the symbol_value contains the value of the exported JCL symbol. If the length of the exported JCL symbol is shorter than 255 characters, the returned string is padded with blanks.	A 255-byte fixed-length string. On return from this service, the symbol_value contains the value of the exported JCL symbol. If the length of the exported JCL symbol is shorter than 255 characters, the returned string is padded with blanks.
value_length	A full-word integer that contains the length of the value of the specified JCL symbol.	A full-word integer that contains the length of the value of the specified JCL symbol.
fc	A 12-byte feedback code that is optional.	A 16-byte feedback code that is optional.

Note: Lowercase characters in the symbol_name will be converted to uppercase.

Figure 26-6 on page 343 shows an example of using the 64-bit function, **__le_ceegtjs**. This simple C program gets the value of the JCL S1 symbol.

```
 /*********************************************************************
 * project      : ITSO redbook SG24-5034                              *
 *                                                                    *
 * function     : get JCL Variables                                   *
 *                                                                    *
 * history      : 21.05.2013 Lutz       first creation                *
*********************************************************************/
 #define  _LP64
 #include <stdio.h>
 #include <stdarg.h>
 #include <errno.h>
 #include <string.h>
 #include <time.h>
 #include <__le_api.h>
 #include <leawi.h>
 #include <ceeedcct.h>
/*********************************************************************
 * main program                                                       *
 *********************************************************************/
int main( int argc, char * argvÝ ¨  )
 {
   _INT4       function_code;
   _VSTRING    symbol_name ;
   _CHAR255    symbol_value;
   _INT4       value_length;
   _FEEDBACK   fc;
   char *symbol="S1";

   printf("JCLSym Version 1.0 IBM ITSO \n");

   function_code=1;                            /* Set fucntion code */

   symbol_name.length=strlen(symbol);          /* fill request Block*/
   memcpy(symbol_name.string, symbol,strlen(symbol));

   __le_ceegtjs( &function_code,               /* Execute Function  */
                 &symbol_name,
                  symbol_value,
                 &value_length,
                 &fc );

    if( _FBCHECK (fc, CEE000) !=0) {
    printf("Get JCL Symbol failed, Error code %d\n",fc.tok_msgno);
    exit(1); }

    symbol_valueÝvalue_length¨='\0';
    printf("JCL symbol %s is %s",symbol,symbol_value);

  return;
 }
```

Figure 26-6 Sample of using __le_ceegtjs

Figure 26-7 shows an example of JCL to compile the C program to get the load module.

```
//LUTZCP   JOB ,'COMPILE',NOTIFY=LUTZ,REGION=512M,
//         CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1),TIME=1440
/*JOBPARM S=SC74
//COMP     EXEC EDCQCB,
//        INFILE='LUTZ.SOURCE(JCLSYM)',
//        OUTFILE='LUTZ.LOADLIB(JCLSYM),DISP=SHR'
//BIND.SYSIN  DD *
  NAME JCLSYM(R)
/*
//
```

Figure 26-7 Sample of compile the JCL

Now, you are ready to run the program. In Figure 26-8, sample JCL demonstrates the use of JCL symbols. As you can see, we set a JCL symbol called S1 with a value of `JFK`.

```
//LUTZJSYM JOB ,'SYMBOLS',NOTIFY=LUTZ,REGION=512M,
//         CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1),TIME=1440
/*JOBPARM S=SC74
//RUN      EXEC PGM=JCLSYM
//STEPLIB DD  DSN=LUTZ.LOADLIB,DISP=SHR
//EXPORT1 EXPORT SYMLIST=(S1)
//SET1 SET S1=JFK
//
```

Figure 26-8 Example of JCLSYM

If the JCL symbol is set properly, you see output similar what Figure 26-9 shows.

```
JCLSym Version 1.0 IBM ITSO
JCL symbol S1 is JFK
```

Figure 26-9 Sample output of JCLSYM

Important: If you receive an error code 3753, the JCL symbol is not defined properly.

26.4 AMODE 64 support for 1M and 2G large pages

In z/OS 2.1, you can use the PAGEFRAMESIZE64 runtime option to request that storage be backed by either 4K or 1M pages for user heap, library heap, I/O heap, and stack for Language Environment AMODE 64 applications. The ability to back storage with 1 M pages can improve performance of certain AMODE 64 applications.

26.4.1 Invocation and use

The option can be placed in your CEEPRMxx member or set dynamically by using the **SETCEE** system command. Figure 26-10 on page 345 shows the syntax of this Language Environment runtime option.

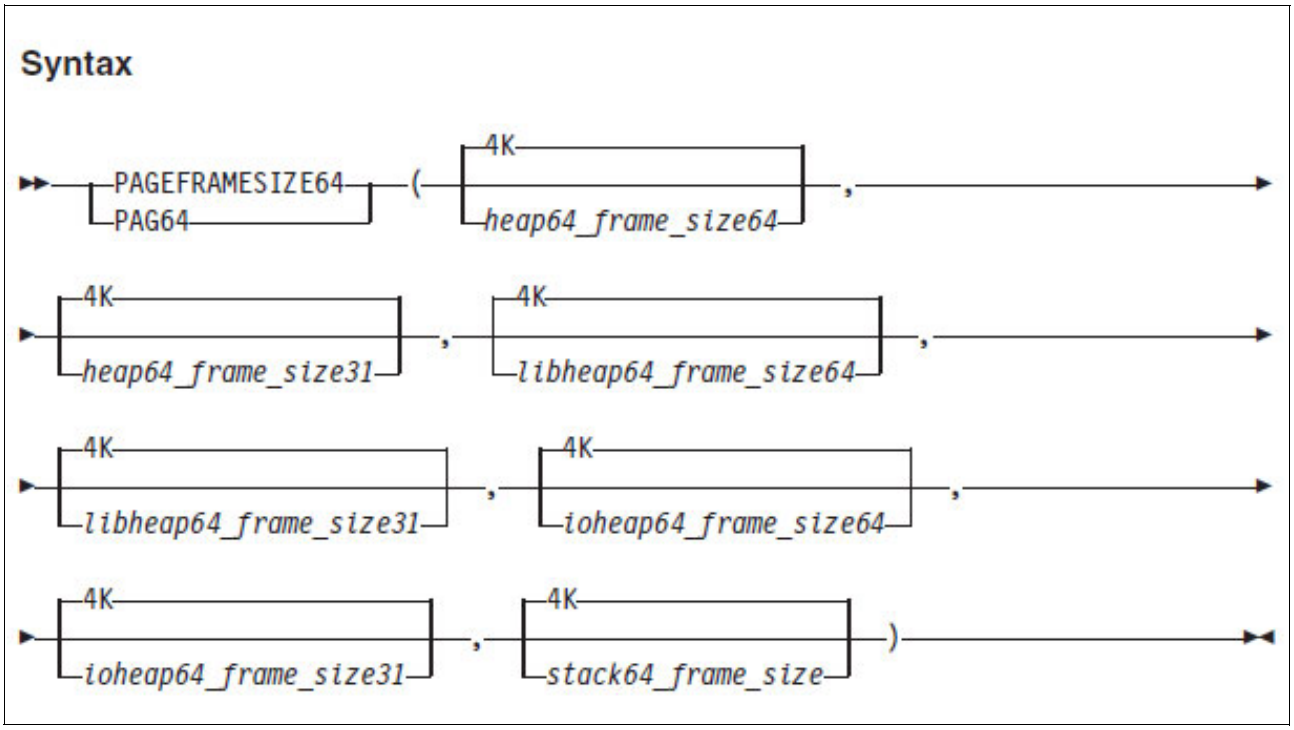


Figure 26-10 Syntax of PAGEFRAMESIZE64 option

The details about the parameters for the PAGEFRAMESIZE64 option are documented in Table 26-6.

Table 26-6 Parameters for PAGEFRAMESIZE64 option

Parameter	Description

`heap64_frame_size64`
`libheap64_frame_size64`
`ioheap64_frame_size64`
	Specifies the preferred page frame size in virtual storage for initial heap/library heap/I/O heap storage allocation and any subsequent heap/library heap/I/O heap increments above the 2 GB bar. These can be specified as one of the following values:
	► 4 K: Requests the default 4 KB pages
	► 1 M: Requests the 1 MB pages to be used, if available

`heap64_frame_size31`
`libheap64_frame_size31`
`ioheap64_frame_size31`
	Specifies the preferred page frame size in virtual storage for initial heap/library heap/I/O heap storage allocation and any subsequent heap/library heap/I/O heap increments above the 16 MB line and below the 2 GB bar. These can be specified as one of the following values:
	► 4 K: Requests the default 4 KB pages
	► 1 M: Requests the 1 MB pages to be used, if available

`stack64_frame_size`	Specifies the preferred page frame size in virtual storage for initial stack storage allocation above the 2 GB bar. This can be specified as one of the following values: ► 4 K: Requests the default 4 KB pages ► 1 M: Requests the 1 MB pages to be used, if available

You cannot specify this option with the CEEBXITA Assembler user exit interface. If 1 MB page frames are not available, 4 KB page frame size will be used. No message is issued to indicate this behavior. Page frame sizes larger than 4 KB are not allowed below the 16 MB line. If a **`PAGEFRAMESIZE64`** parameter specifies 1 MB but that storage type is allocated below the 16 MB line, the default 4 KB page frames will be used. No message is issued to indicate this

behavior, and the runtime option report will show what the user specified. If any **PAGEFRAMESIZE64** parameter specifies 1 M, all of the storage preallocated to the enclave will request 1 MB page frames. The previous two use notes apply as well. When running in a preinitialized environment with a CELQPIPI GETSTORE service routine, a flag is passed to indicate that storage was requested to be backed by 1 MB page frames.

26.4.2 Interactions and dependencies

The large pages support must be available on the hardware and must be activated through the **LFAREA** z/OS initialization parameter.

26.5 Nested CEEPIPI environments

Preinitialization applications running in a MAIN_DP environment are unable to call CEEPIPI to run applications in nested MAIN_DP environments. In earlier releases, the CEEPIPI call failed if this was attempted. With z/OS 2.1, certain CEEPIPI calls from an active MAIN_DP environment are allowed. An application running in a MAIN_DP environment can call another application in a nested MAIN_DP environment without having to return to the Assembler PreInit driver to call the second application.

26.5.1 Invocation and use

To run preinitialization applications in nested MAIN_DP environments, you need to follow these basic steps:

1. Create more than one MAIN_DP environment, using the existing CEEPIPI init_main_dp function (integer value = 19), and keep track of the output tokens.
2. Use the existing CEEPIPI call_main function (integer value = 2) with one of the tokens to run program A.
3. From program A, use the CEEPIPI call_main function with another token to invoke program B in a nested MAIN_DPenvironment.
4. Program B can invoke yet another program in a third nested MAIN_DP environment.

There is no specified limit to the number of levels of nesting for MAIN_DP environments, although nesting is limited by storage or other constraints. Each level of MAIN_DP environment nesting requires a separate pre initialization token. This nested MAIN_DP support is for AMODE 31 only (CEEPIPI only, not CELQPIPI). Nested enclaves are supported within nested MAIN_DP environments, and nested enclaves can call CEEPIPI to run a nested MAIN_DP environment.

26.5.2 Interactions and dependencies

- If the calling MAIN_DP environment has a user-provided @EXCEPRTN, the nested MAIN_DP must also have a user-provided @EXCEPRTN.
- If the user-provided Assembler preinitialization driver has established a SPIE or ESPIE routine, the nested MAIN_DP environments must have a user-provided @EXCEPRTN.
- All CEEPIPI calls that use a specified MAIN_DP preinitialization token must be on the same TCB.
- MAIN_DP environments are POSIX(OFF) only, whether nested or not.

26.6 BSAM type=blocked support

The C runtime library I/O interface was enhanced to allow for high-performance processing of sequential data sets to copy and transfer data in blocked format. Enhancements in z/OS 2.1 support reading, writing, and repositioning of sequential data sets by blocks, rather than by bytes or records. That can results in dramatic improvement in performance of an XL C/C++ runtime library when there is no need to manipulate data within the blocks.

26.6.1 Invocation and use

The support is invoked by calling the fopen() or freopen() functions on a sequential data set with keyword parameter `type=blocked` specified. Figure 26-11 shows an example of how to open a data set in blocked mode.

```
FILE *stream;
stream = fopen("//USR01.TMP","rb+, lrecl=80,blksize=240, recfm=VB,
type=blocked");
```

Figure 26-11 Sample fopen() in blocked mode

After the data set is opened, other I/O functions can be used to process the stream:

- `fread()` and `fwrite()`
- `rewind()`, `ftell()`, `fseek()`, `ftello()`, `fseeko()`, `fgetpos()`, and `fsetpos()`
- `fflush()`, `fldata()`, and `fclose()`

Restriction: For blocked I/O files, buffering is always meaningless. A block is written to external storage as soon as fwrite() completes. The fflush() function has no effect for files opened with `type=blocked`.

fread()

For files opened in blocked format, fread() is the only interface that supports reading. Each time that you call fread() for a blocked I/O file, fread() reads one block. If calling fread() with a request for less than a complete block, the requested bytes are copied to your buffer, and the file position is set to the start of the next block. If the request is for more bytes than are in the block, one block is read and the position is set to the start of the next block.

z/OS XL C/C++ does not strip any blank characters or interpret any data. The fread() function returns the number of items read successfully. If passing a size argument equal to 1 and a count argument equal to the maximum expected length of the block, fread() returns the length, in bytes, of the block read. If passing a size argument equal to the maximum expected length of the block, and a count argument equal to 1, fread() returns either 0 or 1, indicating whether block of length “size” bytes was read. If a block is read successfully but is less than “size” bytes long, `fread()` returns 0.

Attention: A failed read operation can lead to undefined behavior until you reposition successfully.

fwrite()

Attention: A failed read operation can lead to undefined behavior until you reposition successfully.

The fwrite() function is the only interface allowed for writing to a file opened for blocked I/O. Only one block is written at a time. If you write more than your block (BLKSIZE) can hold, your data will be truncated. Writing less than BLKSIZE bytes leads to short block upon blocks and, in case of updates only, the requested part of the block will be updated. At the completion of a fwrite() function, the file position is at the start of the next block, and the block is flushed to the system. By default, all fixed-format records must be full and any block that you write must be a multiple of a record, or the write request fails.

26.6.2 Interactions and dependencies

Only binary open mode on BSAM I/O is supported for blocked I/O. Byte I/O functions listed below are not supported for blocked I/O:

- fgetc() and fgets()
- fputc() and fputs()
- fprintf(), printf(), and sprintf()
- fscanf(), scanf(), and sscanf()
- fwide()
- getc(), getchar(), and gets()
- getc_unlocked(), getchar_unlocked(), putc_unlocked(), and putchar_unlocked()
- putc(), putchar(), puts(), and putwchar()
- ungetc() and ungetwc()
- vfprintf() and vprintf()considerations

26.7 N1040 functions

The latest C programming language standard, ISO/IEC 9899:2011(C11), adds two types, char16_t and char32_t, along with syntax for character and string literals of those types, and four functions: mbrtoc16(), mbrtoc32(), c16rtomb(), c32rtomb(). They enhance the C language for manipulating the Unicode characters.

With z/OS 2.1, you get the four functions specified in C11: mbrtoc16(), mbrtoc32(), c16rtomb(), and c32rtomb(). You can use these four functions to convert between multibyte characters and specified Unicode characters.

26.7.1 Invocation and use

The mbrtoc16 and 32 functions convert a multibyte character to char16_t or char32_t. The other two functions c16rtomb/c32rtomb converts char16_t or char32_t to a multibyte character. Figure 26-12 on page 349 shows the syntax of the functions.

```
#include <uchar.h>
size_t c16rtomb(char * restrict s,char16_t c16,mbstate_t * restrict ps);
size_t c32rtomb(char * restrict s,char32_t c32,mbstate_t * restrict ps);

size_t mbrtoc16(char16_t * restrict pc16,const char * restrict s,size_t n,
                mbstate_t * restrict ps);
size_t mbrtoc32(char32_t * restrict pc32,const char * restrict s,size_t n,
                mbstate_t * restrict ps);
```

Figure 26-12 Prototypes of the functions

The mbrtoc16 and mbrtoc32 functions are very similar. Example 26-1 shows an example of using mbrtoc16 to convert a multi-byte character to char16_t.

Example 26-1 Example of using mbrtoc16 and c16rtomb

```
/*********************************************************************
 * project      : ITSO Redbooks SG24-5034                             *
 *                                                                    *
 * function     : example of using mbrtoc16                           *
 *********************************************************************/
#include <uchar.h>
int main(void)
{
  char16_t c16;
  char mbs[] = "a" ; /* string containing the multibyte char */
  mbstate_t ss = 0 ; /* set shift state to the initial state */
  int length = 0 ;
  length = mbrtoc16(&c16, mbs, MB_CUR_MAX, &ss);
  if (length < 0) {
     perror("mbrtoc16() fails to convert");
     exit(-1);
     }
  printf(" mbs:\"%s\"\n", mbs);
  printf(" length: %d \n", length);
  printf(" c16: 0x%04x \n", c16);
}
/*********************************************************************
 * project      : ITSO redbook SG24-5034                              *
 *                                                                    *
 * function     : example of using c16rtomb                           *
 *********************************************************************/
#include <uchar.h>
int main(void)
{
  char16_t in = u'a';
  mbstate_t st = 0;
  char out[MB_CUR_MAX];
  int rc, i;
  rc = c16rtomb(out, in, &st);
  if (rc < 0) {
      perror("c16rtomb() fails to convert");
      exit(-1);
      }
  printf(" c16: 0x%04x \n", in);
```

```
  printf(" return code: %d \n", rc);
  printf(" mb character: ");
  for (i=0; i<rc; i++)
    printf(" 0x%02x", out[i]);
  printf("\n");
  return 0;
}
```

To use this function, you must compile with LANGLVL(EXTC1X). The function supports only the code sets (CCSIDs) that are provided by Unicode Service. Figures 26-13 and 26-14 show the output of the sample code that was introduced previously in Figure 26-12 on page 349.

```
mbs:"a"
length: 1
c16: 0x0061
```

Figure 26-13 Sample output of mbrtoc16 use

```
c16: 0x0061
return code: 1
mb character: 0x81
```

Figure 26-14 Sample output of c16rtomb use

26.8 Standard I/O extensions

In z/OS 2.1, applications can query an open FILE stream for information about the current status of the stream. Other types of modifiable requests can be performed against the open FILE stream also. Twelve APIs are included in a header, `<stdio_ext.h>`. Non-threadsafe versions of six of the APIs are provided. The header also defines macros for use with APIs.

26.8.1 Invocation and use

Table 26-7 shows all newly implemented functions. Only those marked as Threadsafe ensure a safe execution in a multiple-thread environment. Thread safety is a computer programming concept that is applicable in the context of multi-threaded programs. A piece of code is *threadsafe* if it manipulates shared data structures only in a manner that guarantees safe execution by multiple threads at the same time.

Table 26-7 i/O extensions

Function	Description	Threadsafe
`__fbufsize`	Takes an open FILE stream pointer, stream. Returns the size of the buffer in bytes.	Yes
`__flbf`	Takes an open FILE stream pointer, stream. Determines whether the specified stream is line-buffered.	Yes
`_flushlbf`	Flushes all open line-buffered streams. Behaves similar to the fflush() function.	Yes

Function	Description	Threadsafe
_flushlbf_unlocked()	Flushes all open line-buffered streams. Behaves similar to the _flushlbf function but without threadsafe.	No
__fpending	Takes an open FILE stream pointer, stream. Returns the number of bytes pending to be written in the output buffer. When the stream is opened for text processing, the function returns the number of characters that is pending (can be wide characters).	Yes
__fpending_unlocked	__fpending_unlocked() is functionally equivalent to __fpending() with the exception that it is not threadsafe.	No
__fpurge	Takes an open FILE stream pointer, stream. The function requests that any pending data that is in the stream be discarded.	Yes
__fpurge_unlocked	__fpurge_unlocked() is functionally equivalent to __fpurge() with the exception that it is not threadsafe.	No
__freadable	Takes an open FILE stream pointer, stream. Determines if the specified stream has been opened for reading.	Yes
__freadahead	Takes an open FILE stream pointer, stream. Returns the number of bytes remaining to read in the input buffer. When the stream is opened for text processing, the function returns the number of characters remaining to be read (can be wide characters).	Yes
__freadahead_unlocked	__freadahead_unlocked() is functionally equivalent to __freadahead() with the exception that it is not threadsafe.	No
__freading	Takes an open FILE stream pointer, stream. Determines if the last operation on the specified stream was a read operation or if the specified stream is open for read-only.	Yes
__freading_unlocked	__freading_unlocked() is functionally equivalent to __freading() with the exception that it is not threadsafe.	No
__fseterr	Takes an open FILE stream pointer, stream. The function sets the specified stream in error.	Yes
__fsetlocking	Takes an open FILE stream pointer, stream and a locking request, type. The function allows the type of locking on an opened stream to be controlled or queried by the application. ► FSETLOCKING_INTERNAL: Subsequent stdio functions perform implicit locking. This is the default behavior. ► FSETLOCKING_BYCALLER: Subsequent stdio functions assume that the caller is responsible for maintaining the integrity of the stream in the face of access by multiple threads. ► FSETLOCKING_QUERY: Returns the current locking type of the stream without changing it.	Yes
__fwritable	Takes an open FILE stream pointer, stream. Determines whether the specified stream has been opened for writing.	Yes

Function	Description	Threadsafe
__fwriting	Takes an open FILE stream pointer, stream. Determines whether the last operation on the specified stream was a write operation or if the specified stream is open for write-only or append-only.	Yes
__fwriting_unlocked	__fwriting_unlocked() is functionally equivalent to __fwriting() with the exception that it is not threadsafe.	No

The majority of the functions return 0 when in error and set errno accordingly. Boolean type querying functions return 1 when true and 0 otherwise.

Restriction: The function __fpurge is supported for UNIX files only.

26.9 Unicode auto-conversion support

Today's installations can no longer assume that files are read or written on the same system or at the same geographic location where they originated. Tagging and conversion of files between varying code pages is expected to become more prevalent in the future. z/OS 2.1 provides an interface to the z/OS UNIX Unicode auto-conversion environment. The interface allows applications the use of files that come from other platforms without change, and it can generate files that can be used directly by applications on other platforms.

26.9.1 Invocation and use

To enable the conversion environment, you can use one of the following methods:

1. Set **_BPXK_AUTOCVT** environment variable to **ALL**.
   - The setenv() and putenv() have no effect on multi-thread programs.
   - Non-IPT threads have the same AUTOCVT state as an IPT thread.
2. Use the C runtime library fcntl() function to set cvtcmd to **SETCVTALL**.
3. Using the C runtime library function __ae_autoconvert_state() with **CVTSTATE_ALL**.

To query the conversion environment of a thread, use one of the following methods:

1. Using C runtime library function fcntl() to set cvtcmd to **QUERYCVT**.
2. Using C runtime library function __ae_autoconvert_state() with **CVTSTATE_QUERY**.

To tag a file, you can use the C runtime library function fcntl() with **F_SETTAG**. To set a CCSID to a thread, set the _BPXK_PCCSID environment variable to the CCSID.

26.10 C11 validation Items

In the latest C language standard, C11, changes and enhancements were implemented. Starting with V2.1 of z/OS, some of these changes and enhancement are implemented in the Language Environment runtime to make Language Environment compliant with C11.

Single UNIX Specification Version 3 (SUSv3) incorporates POSIX.1, POSIX.2, and their subsequent amendments, as well as the core volumes of the Single UNIX Standard,

Version 2. The Language Environment's implementation of SUSv3 is based on IEEE Standard 1003.1-2004, which includes IEEE STD 1003.1-2001 and the two subsequent corrigenda issued after the initial release of the standard. From the C/C++ and Language Environment point of view, many standards have been implemented in the latest z/OS releases, and we continue with this implementation in z/OS 2.1.

Invocation and use

This section discusses behavioral changes that are provided for SUSV3 compliance in an error path. The affected interfaces are typically setting errno to values that were not used before and, in some cases, returning failure for conditions that had not been tested before SUSV3.

By default, the affected interfaces do not check for these conditions. When the value of **_EDC_SUSV3** is set to `1` or `2`, the SUSV3 behavior is enabled. Existing programs using these interfaces and running with _EDC_SUSV3=1 or 2 might fail in z/OS 2.1 with an errno, where in the past, they might have appeared to succeed. You must set **_EDC_SUSV3=1** or **2** to get the behavior.

z/OS XL C/C++ Runtime Library Reference, SA14-7314 documents the use of _EDC_SUSV3 in individual interface descriptions. The functions that are affected by the _EDC_SUSV3 environment variable are shown in Table 26-8.

The **_EDC_SUSV3** environment variable can be set with the setenv function:

```
setenv("_EDC_SUSV3","2",1)
```

The following functions are made compliant in V2.11 when the environment variable **_EDC_SUSV3=2** is set:

- `log(), logf(), logl()`
- `log10(), log10f(), log10l()`
- `log1p(), log1pf(), log1pl()`
- `log2(), log2f(), log2l()`
- `pow(), powl()`

Table 26-8 _EDC_SUSV3 values

_EDC_SUSV3	Description
0	Functions performed without SUSV3 behavior. This is the default. It is like unsetting that variable.
1	Enables SUSV3 behavior for setenv(), readdir(), getnameinfo(), and tcgetsid().
2	Enables SUSV3 behavior for log(), logf(), logl(), log10(), log10f(), log10l(), log1p(), log1pf(), log1pl(), log2(), log2f(), log2l(), pow(), and powl().

Important: When **_EDC_SUS3 = 2** is set, setenv(), readdir(), getnameinfo(), and tcgetsid() are also enabled for the SUSV3 behavior.

Default settings of **_EDC_SUSV3** preserve old behavior, so these environment variables do not raise a migration issue. However, users should be aware of the behavioral differences in the application environment when setting them.

26.10.1 Migration and coexistence considerations

Set environment variable **_EDC_SUSV3=2** to get SUSv3 error handling for setenv(), readdir(), getnameinfo(), and tcgetsid().

26.11 C11 alignment-related support

z/OS 2.1 introduces a header file, <stdalign.h>, which defines following macros according to the C11 standard:

- alignas
- alignof
- __alignas_is_defined
- __alignof_is_defined

You need to include the <stdalign.h> header in your program to get some of the features of C11.

26.11.1 Invocation and use

C11 provides syntax for specifying alignment. The expression **alignof(*type-name*)** designates the alignment of type-name; it is a constant expression, as is the familiar **sizeof(*type-name*)**. There is one exception in C: Applying **sizeof** to a variable length array produces a non-constant expression.

There is a similar syntax for declarations:

```
int alignas(double) b;
```

This specifies that b is an int but is aligned suitably for a double. Or for a more realistic example:

```
alignas(double) unsigned char buf[sizeof(double)];
```

This specifies that buf is an array of unsigned char, with a size and alignment that are suitable to hold a double.

Alignment can be specified as an integer. For example, alignas(constant-expression) specifies the constant-expression as an alignment. Therefore, alignas(type-name) means the same thing as alignas(alignof(type-name)).

For the z/OS platform, there is a type that has the largest alignment requirement. That type can be named by the typedef max_align_t, so a declaration that specifies alignas(max_align_t) requests an alignment that will be suitable for any type on that platform. If a program requests an alignment that is greater than alignof(max_align_t), the program is not portable because support for an over-aligned type is optional.

The C11 library provides aligned_alloc(size_t bound, size_t nbytes), which allocates nbytes of storage aligned on a bound boundary.

Important: To use stdalign.h,compile source code with language level EXTC1X.

26.11.2 Migration and coexistence considerations

The existing behavior that is for POLE errors is unchanged. To request the C11 POLE error behavior, set **_EDC_SUSV3=2**.

26.12 Documentation and reference

See the following publications in “IBM z/OS V2R1.0 documentation” in the IBM Knowledge Center for z/OS 2.1.0 for more information:

http://ibm.co/1oZaqdP

- *Language Environment Concepts Guide*, SA38-0687
- *Language Environment Customization*, SA38-0685
- *Language Environment Debugging Guide,* GA32-0908
- *Language Environment Programming Guide*, SA38-0682
- *Language Environment Programming Guide for 64-bit Virtual Addressing Mode*, SA38-0689
- *Language Environment Programming Reference*, SA38-0683
- *Language Environment Vendor Interfaces*, SA38-0688
- *z/OS Language Environment Runtime Messages*, SA38-0686
- *z/OS XL C/C++ Programming Guide*, SC14-7315
- *z/OS XL C/C++ Runtime Library Reference*, SC14-7314

27

Resource Recovery Services

This chapter describes the enhancements, functions, and features that were introduced for the Resource Recovery Services (RRS) component of z/OS Version 2 Release 1 (2.1). It includes these two sections:

- 27.1, “RSS Internal Cold Start feature” on page 358
- 27.2, “Documentation and reference” on page 360

© Copyright IBM Corp. 2014. All rights reserved.

27.1 RSS Internal Cold Start feature

The resource manager (RM) data log is vital to RRS, and RRS cannot function without it. When an RM registers with RRS on a particular system in the IBM Parallel Sysplex, an in-storage copy of the RM is created, along with an entry in the RM data log. The RM data log contains active and inactive RMs so that if something happened to RRS, the RMs involvement with RRS can be recreated from the log.

A nonfunctioning RM data log is corrected by manual deletion and redefinition of the log stream. That action requires RRS to be inactive. In a Parallel Sysplex, this means recycling RRS on all systems with a cold start of RRS, using the ATRCOLD procedure, and then a recycle of all active resource managers that were connected to RRS. During that time, no application that requires RRS will able to run properly.

A feature called RRS Internal Cold Start has been added. This feature can help avoid this Parallel Sysplex-wide outage if an RRS cold start is required. With an Internal Cold Start, if something happens to the RM data log, the log can be recreated from the in-storage copy of each RM on every system in the sysplex

27.1.1 Invocation and use

The RRS Internal Cold Start function is enabled by default and can be used by the Parallel Sysplex if all the following requirements are met:

- Systems in the Parallel Sysplex are running z/OS 2.1.
- RRS is not initializing or terminating.
- In-storage copies of the RMs are viable.
- RRS remains active on all of the systems in the logging group.

When RRS detects an RM data loss, you will see the ATR212I message that is shown in Figure 27-1.

```
ATR212I RRS DETECTED LOG DATA LOSS ON LOGSTREAM ATR.PLEXX.RM.DATA DUE TO
INACCESSIBLE LOG DATA. LOG DATA FROM 2012/09/17 17:30:45 TO 2012/09/17 17:46:11
ARE AFFECTED.
```

Figure 27-1 Sample ATR212I error message about a failing RRS log stream

Starting with z/OS 2.1, you will see an additional system message, ATR250E, as shown in Figure 27-2.

```
ATR250E RRS LOGSTREAM ERROR FOUND. CORRECT THE ERROR OR OPTIONALLY REPLY
COLDSTART TO BEGIN A RRS INTERNAL COLD START.
```

Figure 27-2 Sample ATR250E message about the RRS Internet Cold State

At that point, you have two options the recover the log stream:

- Use of the existing ATRCOLD procedure.
- Reply `COLDSTART` to the ATR250E message to initiate the automated internal RRS Cold Start.

When you reply with COLDSTART to the message or from Auto Reply, RRS proceeds with the following actions:

- Deletes all data from the RM data log stream
- Disconnects from the RM data log stream.
- Based on the previous two actions, the logger deletes all of the offload data sets. However, there might be a delay before the data sets are deleted.
- Reconnects to the RM data log stream.
- Relogs all of the resource managers.
- Allows transactions to proceed, unaffected by the log error.
- Completes with ATR253I (shown in Figure 27-4).

If a reply of COLDSTART is given to instruct RRS to attempt an Internal Cold Start, RRS remains active, but work will not be accepted until the cold start has finished. An attempt will be made to save in-storage transactions, which will be relogged as part of the Internal Cold Start procedure. This prevents a system-wide outage of RRS.

Without the COLDSTART reply, a cold start of RRS is required, using the ATRCOLD procedure, which requires RRS to be terminated properly on all systems in the RRS group. After it is terminated, request a cold start of RRS using the ATRCOLD procedure, and then restart RRS on each system in the RRS group. This can be done manually, but it results in all outstanding transactions being lost and not recoverable.

Note: After an Internal Cold Start, the log data is marked for deletion. The system logger does not physically delete the log data or log data set until an offload requires a data set to be allocated. This means that there will often be a delay before eligible log data sets are physically deleted, because offloading will not occur until the high threshold associated with the log stream is reached. If off-loads are relatively infrequent, there might be a considerable delay before log data sets that are eligible for deletion are actually deleted.

When using the Internal Cold Start function, the ATR251I system message is issued on the system that is selected as the cold start coordinating systems. Other systems issue message ATR252I, which informs you about the progress of the cold start, as you see in Figure 27-3.

```
ATR251I RRS INTERNAL COLD START IS IN PROGRESS
ATR252I RRS INTERNAL COLD START IS IN PROGRESS
```

Figure 27-3 RRS Cold start progress

After the cold start is finished successfully, you will see either the ATR253I or the ATR254I message that is shown in Figure 27-4. ATR253I and ATR254I are the same, but ATR253I is issued to the operator console and ATR254I is written to the hardcopy log.

```
ATR253I RRS INTERNAL COLD START HAS COMPLETED SUCCESSFULLY
```

Figure 27-4 RRS cold start completed

At this point, RRS Internal Cold Start processing has completed successfully. The severe log stream error that prompted the cold start has been corrected. There was no loss of data or resource managers.

If for some reason an Internal Cold Start is not allowed due to an error or prerequisite that is not being met, the ATR257E message shown in Figure 27-5 is issued. In this case, RRS continues in a degraded state and the error must be resolved by using the ATRCOLD procedure.

```
ATR257E INTERNAL COLD START PROCESSING FAILED TO START
```

Figure 27-5 Cold Start now allowed message

Any failures or errors encountered during the Internal Cold Start process result in the inability to recreate the log data from the in-storage RMs. RRS will be terminated on all systems in the logging group. The ATR255E message shown in Figure 27-6 is issued, and RRS will be abended with ABEND code 5C4 and reason 29. RRS must be restarted by using the ATRCOLD procedure.

```
ATR255E RRS INTERNAL COLD START FAILED
```

Figure 27-6 Cold start failure message, ATR255E

27.1.2 Interactions and dependencies

The use of the Internal Cold Start function requires that the entire Parallel Sysplex be running z/OS 2.1 or a later version. If your sysplex contains systems with earlier releases of z/OS, no further problem resolution is done by the system. In that case, you must follow your manual recovery process to fix the problem.

27.2 Documentation and reference

You can find the following documentation in the z/OS 2.0.1 IBM Knowledge Center:

https://ibm.biz/BdFRTU

- *z/OS MVS Programming: Resource Recovery*, SA23-1395

  http://ibm.co/1sIYhfF
- *z/OS MVS System Codes*, SA38-0665

  http://ibm.co/1txkwco
- *z/OS MVS System Messages, Vol 3 (ASB-BPX)*, SA38-0670

  http://ibm.co/1mTjSTd
- *z/OS Problem Management*, SC23-6844

  http://ibm.co/1qsIKyW

28

Base Control supervisor and timer supervisor changes

This chapter describes the following Base Control Program (BCP) supervisor and timer supervisor enhancements that were introduced with z/OS Version 2 Release 1 (2.1).

- 28.1, “Transaction execution” on page 362
- 28.2, “Offload time at task level” on page 371
- 28.3, “Larger and extendable facility list” on page 372
- 28.4, “Tracing disabled interrupt exits” on page 373
- 28.5, “Entry table entries expanded” on page 373
- 28.6, “Support for clock values in the second UNIX epoch” on page 373
- 28.7, “Reserve SVC entries to use yet maintain ABEND upon use” on page 374
- 28.8, “ACCURACY keyword in CLOCKxx” on page 375
- 28.9, “Documentation and reference” on page 377

© Copyright IBM Corp. 2014. All rights reserved.

28.1 Transaction execution

The transaction execution facility (TX) was introduced with the IBM zEnterprise EC12 (zEC12) system. z/OS 2.1 provides the infrastructure so that the z/OS software, middleware, and applications can use TX.

APAR OA38829 was included with z/OS Version 1.13 to coincide with the zEC12 GA. It provided minimal infrastructure for Java and for testing use in anticipation of the z/OS 2.1 support. z/OS 1.13 is considered a “test environment” for transaction execution and should not be used for production code.

Primarily, what was missing was diagnostic support that is needed to debug either development or production code problems when using TX.

28.1.1 Overview

The *transaction execution facility* is a hardware-based tool that is with IBM zEnterprise EC 12 (zEC12). It supports a *transaction*, which is a sequence of instructions that act atomically.

Terminology

The following terms are used in describing the transaction execution facility:

Transaction	A set of storage accesses and general register changes that appear to occur as a single concurrent transaction.
Abort	A transaction is said to *abort* if execution ends before the end of the transaction.
Commit	At the end of the transaction, the accesses to storage are committed.
Conflict	A conflict occurs for a transaction storage access if the same cache line is accessed at the same time and one or both accesses is a store.
Constrained transaction	Runs in a constrained transaction execution mode. Only a subset of instructions is available, and the transaction eventually completes.
Nonconstrained transaction	Runs in a non-constrained transaction execution mode. Not limited to a subset of instructions but might be aborted, so not completed.
Transaction execution mode	The state that is entered when a transaction begins.

These terms are explained in more detail during the following discussion.

instructions for transaction execution

The following instructions are used during transaction execution:

`TBEGIN`	Causes the CPU to remain in non-constrained or *begin* transaction execution mode.
`TBEGINC`	Causes the CPU to remain in or *begin constrained* transaction execution mode.
`TABORT`	*Transaction abort* action if it is not constrained.
`TEND`	Causes the *transaction end*, which causes storage access and general register changes to be committed.

PPA	*Perform processor assistance* to increase the likelihood of transaction success during abort processing.
NTSTG	*Non-transactional storage* is used during non-constrained transactions to perform stores that will be retained even if the transaction aborts.

Transaction execution

A transaction begins with either TBEGIN, for a non-constrained transaction, or TBEGINC, for a constrained transaction, and it ends with TEND. A non-constrained transaction either completes successfully or aborts. A constrained transaction eventually completes.

When a non-constrained transaction aborts:

- Storage is unchanged compared to the time of beginning, except for storage that is updated through the NTSTG instruction and the transaction diagnostic block (TDB).
- General purpose registers may be unchanged.
- Access registers are not unchanged.
- Floating point registers are not unchanged.

28.1.2 Invocation and use

The transaction execution (TX) facility is fully available when the CVTTX and CVTTXC bits in CVTFLAG4 are on. These bits are on by default for z/OS 2.1.

The transaction execution facility is available for testing only when CVTTXTE is on. This bit is on for z/OS Version 1.13 when OA38829 is installed.

Special rules are applied to instructions that are executed within transaction execution mode. These are described in detail in the *z/Architecture Principles of Operation*, SA22-7832 (PoOp) manual. The “Restricted Instruction” section of Chapter 5 lists the instructions that are not allowed within a transaction. In general, the instructions in Chapter 7 of the PoOp are available, but control instructions that are listed in Chapter 10 of the PoOp are not.

The instructions are treated as block-concurrent as observed by other CPUs and the channel subsystem. The TX facility provides the serialization that you might otherwise need to implement yourself.

A transaction execution conflict can occur in many ways. One is that during a storage access for one transaction another access is made to the same cache line, and one or both accesses is a store. Not all conflicts can be controlled by the application. When a conflict occurs, the transaction cannot complete successfully, but it might succeed if it is retried.

When no conflict exists, one processor might be able to complete in a simple way, without having to obtain software-managed serialization or using serializing instructions.

You can think of *stores* within a transaction as being “rolled back” upon transaction abort. Similarly, you can think of the *registers* as being saved when transaction begins and then (optionally) rolled back to their pre-transaction values.

Types of transactions

There are two types of transactions:

- Non-constrained
- Constrained

Non-constrained transactions

A non-constrained transaction has the following characteristics:

- Begins with TBEGIN
- Ends normally with TEND
- Can be aborted with TABORT
- May be aborted for many system-defined reasons
- TBEGIN may identify a transaction diagnostic block (TDB), which is mapped by the IHATDB macro
- Upon an abort action, the flow proceeds to the instruction after TBEGIN

The instruction after TBEGIN is usually a conditional relative branch to handle the condition codes that result from TBEGIN completion.

CC=0	Transaction initiation successful, so it should “fall through.”
CC=1	Abort due to an indeterminate condition. It should branch somewhere to deal with this situation.
CC=2	Abort due to a transient condition. It should branch somewhere to deal with this situation (dealing with this might mean to retry, but should eventually time out and go to the fall-back path). Although there is no number to answer the question “How many times should I retry?” six is the number suggested as the default.
CC=3	Abort due to a persistent condition. It should branch somewhere to deal with this situation. Eventually, it winds up in a fall-back path because, for some reason, the system believes that the transaction is unlikely ever to succeed. (This applies to the current circumstances only. For example, a list search on a hash table might not succeed for the current hash, but for most other hashes, it might succeed.)

Example 28-1 shows a simple non-constrained transaction.

Example 28-1 Non-constrained transaction

```
          LA 2,Source_Data_Word
          LA 3,Target_Data_Word
          TBEGIN theTDB,X'8000' the "80" indicates to restore
*                           GRs 0-1
*                           upon abort, each bit in those 2
*                           hex digits corresponds to a
*                           double register pair
          BRC 7,Transaction_aborted
          L 1,0(,2)
          ST 1,0(,3)
          TEND
<<when you get here, register 1 will have been changed by the "L", and the target
word will have been set>>
...
Transaction_aborted DS 0H

<<when you get here, all the registers will have the value they had before the
TBEGIN instruction (0-1 restored, 2-F not used), the target word will be
unchanged, and the TDB, identified on the TBEGIN instruction, will contain
information about the transaction abort>>
```

Transaction diagnostic block

When a transaction is aborted, status information is saved in a *transaction diagnostic block* (TDB). The TBEGIN-specified TDB is always stored at transaction abort, and the program-interruption TDB is stored when a transaction is aborted due to a program interruption.

The TDB is a 256-byte area mapped by the IHATDB macro, and it has the following format:

- Bytes 8 - F: Transaction abort code, which can include the following action codes:
  - External interrupt (2)
  - Program interrupt (4)
  - I/O interrupt (6)
  - Many codes for overflows and cache conflicts
- Bytes 10 - 1F: Conflict token
- Bytes 18 - 1F: Aborted transaction instruction address
- Byte 20: Exception access ID
- Byte 21: Data exception code
- Bytes 24 - 27: Program interruption ID
- Bytes 28 - 2F: Translation exception ID
- Bytes 40 - 47: Breaking event address
- Bytes 80 - FF: 64-bit GRs or 0 - F

The format of the TDB is shown in detail in Chapter 5 of the PoOp manual, *z/Architecture Principles of Operation*, SA22-7832.

Constrained transactions

A constrained transaction has the following characteristics:

- Begins with TBEGINC
- Ends normally with TEND
- May be aborted for many system-defined reasons
- Upon abort, flow proceeds to the TBEGINC

In the absence of repeated constraint violations, a constrained transaction is assured of eventual completion and so needs no fallback path.

Constrained transactions can execute no more than 32 instructions and have the following constraints:

- All instructions within the transaction must be within 256 contiguous bytes of storage.
- The only branches allowed are relative branches that branch forward. Therefore, there can be no loops.
- No SS or SSE format instructions may be used.
- Additional general instructions may not be used.
- The transaction's storage operands may not access more than four octowords.
- The transaction may not access storage operands in any 4 K byte blocks that contain the 256 bytes of storage that begin with the TBEGINC instruction.
- Operand references must be within a single doubleword, except for some of the "multiple" instructions for which the limitation is a single octoword.

Example 28-2 shows an example of a simple constrained transaction.

Example 28-2 A simple constrained transaction

```
          LA 2,Source_Data_Word
          LA 3,Target_Data_Word
          TBEGINC 0,X'8000'
          L 1,0(,2)
          ST 1,0(,3)
          TEND

<<when you get here, register 1 will have been changed by the "L", and the target
word will have been set>>
```

The transaction might have aborted one or more times, but it succeeded.

The TBEGINC asked that register the 0/1 pair be restored, but there was no need to do so because the transaction did not depend on the initial values of those registers. So the operand could have been `X'0000'`.

User controls over a transaction

You can control the following conditions during transaction execution:

- The general register pairs that are to be saved at the initiation of a transaction (TBEGIN or TBEGINC instruction) and restored if a transaction aborts.
- Whether access register modification is allowed within the transaction. If the transaction aborts, access register values are not restored.
- Whether floating point operations are allowed within the non-constrained transaction. If the transaction aborts, floating point register values are not restored.
- Program interrupt filtering for a non-constrained transaction. For example, the application might ask that certain classes of program interrupts be presented to the application as an abort, rather than processed by the system as a program interrupt.
- Whether data is returned to provide information through the Transaction Diagnostic Block (TDB) about the abort. This is mapped by IHATDB.

Program interruption information

When a program interrupt occurs, the `X'0200'` bit in the program interrupt code indicates that it occurred within transaction execution mode.

The Program Interrupt TDB (PITDB) is saved at location `x'1800'` upon a program interruption during a transaction. It contains information about the register and instruction addresses at the time of the program interruption. This is not saved by z/OS.

A program interrupt code, `x'0018'`, indicates that a transaction constraint exception has occurred within a constrained transaction.

Non-transactional store

The non-transactional store (NTSTG) instruction is issued for a data store that is not rolled back upon abort, so that it can be examined after the transaction ends.

You can use NTSTG to count the number of tries, for example.

High Level Assembler support

High Level Assembler (HLASM) support for the TX instructions is provided through PM49761. HLASM also provides a print exit named ASMAXTXP, which you can use in your assembly. It is available via PM66334. ASMAXTXP flags things as errors that violate a transaction execution restriction, to the extent that it can determine those violations. Primarily, it detects violations of constraints in constrained transactions.

Transaction and fallback considerations

When using non-constrained transactions, the transaction must be serialized against the fall-back path. For example, if one processor is within a transaction and another within the fall-back path, each needs to know enough to protect itself against the other.

In addition, a fall-back path typically needs to be serialized against other concurrent executions of that fall-back path.

You can think of the fall-back path as needing “real” serialization (hardware-provided or software-provided) and the transaction needing to be able to recognize that the fall-back path is running. One way of accomplishing this is for the fall-back path to set a footprint when it has obtained “real” serialization and for the transaction to query that footprint.

Example 28-3 shows a transaction with its fallback path

Example 28-3 Transaction and fallback path example

```
Transaction
          TBEGIN
          BRC 7,Transaction_Aborted
          If I_Have_ENQ is on then TABORT
          the group of updates
          TEND

Fallback_Path
          ENQ
          Set bit I_Have_ENQ
          the group of updates
          Reset bit I_Have_ENQ
          DEQ
```

Even this simple example is incomplete. To avoid cases where a spin would result (when the ENQ holder does not get a chance to reset the bit), the transaction path needs to limit the number of times that the transaction is started over. This can be done using a counter in a register that is not restored upon the abort or a non-transactional store. Therefore, at “Transaction_Aborted,” there might be a test to see whether the flow should proceed to the fall-back path or back to TBEGIN. We suggest limiting the number of CC=2 retry times to six.

The architecture also provides a Perform Processor Assist instruction that should be used before retrying the transaction.

Example 28-4 shows an example of a transaction where these concepts are applied.

Example 28-4 Transaction and abort or retry logic

```
          LHI 15,0 Zero count of transaction aborts
Transaction_Again DS 0H
          TBEGIN
          BRC 7,Transaction_Aborted
```

```
          If I_Have_ENQ is on then TABORT
          the group of updates
          TEND
          …
Transaction_Aborted DS 0H
          JC 5,Fallback_Path Not worth retrying for CC=1,CC=3
          AHI 15,1 One more transaction abort
          CIJNL 15,6,Fallback_Path Give up after 6 tries
          PPA 15,0,1 Perform Processor Assist option
*                    1, count in GR15
          J Transaction_Again
          ...
Fallback_Path DS 0H
```

Transaction execution facility rollout history

Limited support for the transaction execution facility was provided in z/OS 1.13 through APAR OA38829. It is now fully supported by z/OS 2.1.

z/OS 1.13 with OA38829 installed

This is referred to as “TX Lite,” because it provides only the following limited support:

- Bit CVTTXTE is set, which indicates a “TX test environment.”
- There are no diagnostics, but the TDB may be used and is mapped by IHATDB. The program interrupt writes TDB into PSA, as designed, at the x'1800' location, but this is not captured.
- Transactions work, and both non-constrained and constrained transactions are supported.

Note: This version is not recommended for use in a production environment.

z/OS 2.1

Full support for the transaction execution facility is provided with z/OS 2.1:

- Bits CVTTX and CVTTXC are set, which indicates “full support.”
- Program interrupt TDB is captured into SDWA.
- SDWA has both “time of program interrupt registers and PSW” and “transaction abort registers and PSW.”
- EPIE for ESPIE service also has both sets of registers and PSWs.

Transaction execution facility diagnostics

Normal z/OS recovery processing takes place for a program interrupt if the program interrupt is not filtered via the non-constrained transaction TBEGIN instruction.

Additional information is added to the system diagnostic work area (SDWA) that is related to transaction execution. This includes FRR SDWAs and SDWALOC31=YES ESTAE type SDWAs.

- Bit SDWAPTX1 within byte SDWAIC1H in field SDWAAEC1 and bit SDWAPTX2 within byte SDWAIC2H in field SDWAAEC2 indicate that the program interrupt occurred while within transaction execution mode.

  Bit SDWAPTX1 is valid only when bit SDWAPCHK is on.
- Existing fields SDWAG64, SDWAG64H, and SDWAGRSV contain the time of error register information. These are the registers current when the program interrupt occurred.

- Existing field SDWAPSW16 contains the time of error register information. This is the PSW current when the program interrupt occurred.
- When bits SDWAPCHK and SDWAPTX2 are on, field SDWATXG64 contains the registers that resulted from the transaction abort due to the program interrupt.
- When bits SDWAPCHK and SDWAPTX2 are on, field SDWATXPSW16 contains the PSW that resulted from the transaction abort due to the program interrupt.

Random aborts of transaction execution

When running in transaction execution mode in Version 2.1, random aborts can be requested to aid in testing transaction code including the fallback path.

Transaction execution facility controls

Transaction Diagnostic Control (TDC), bits 62 and 63 of Control Register 2 may be used to randomly abort transactions for diagnostic purposes.

- TDC = 0: Normal operation
- TDC = 1: Abort every transaction at a random instruction, but before execution of the outermost TEND instruction.
- TDC = 2: Abort random transactions at a random instruction.

Transaction Diagnostic Scope (TDS), bit 61 of Control Register 2 controls whether TDC applies for problem state or supervisor state.

- TDS = 0: The TDC applies for both problem and supervisor states.
- TDS = 1: The TDC applies only for a problem state.

The IEATXDC macro

The IEATXDC macro (Example 28-5) can be used to request random aborts of transactions for your task so that, when you run it repeatedly, you are likely to exercise both the non-abort and abort paths.

Example 28-5 IEATXDC macro

```
IEATXDC SCOPE={PROBLEM | ALL},
OPERATION={NO_ABORT, SET_EVERY, SET_RANDOM}
```

IEATXDC is implemented by modifying the TDS and TDC bits in Control Register 2 through the DUCT.

DIAGxx options for transaction execution

There are DIAGxx options available for having aborts cover every work unit in the system, including SRBs:

- IEATXABEVERY: Abort every transaction at a random instruction.
- IEATXABRANDOM: Abort random transactions at a random instruction.

Note: Use these DIAGxx options only in non-production environments.

PER event processing during transaction execution mode

A PER event is a program interrupt with code `X'0080'`, so there is a need to avoid an endless abort loop when a PER trap matches within a transaction. To avoid this, there are two PER event controls introduced with the transaction execution mode facility:

- PER event suppression
- PER Transaction End event

Consider a SLIP trap in effect with a PER range covering a constrained transaction. Every time through the transaction, the SLIP trap hits, the PER event interrupt aborts the transaction, and the abort process resumes at the TBEGINC.

To avoid this endless loop, the system recognizes “PER event within constrained transaction” and sets “PER Event Suppression” so that subsequent PER events within a transaction are suppressed. The system sets “PER Transaction End event” so that the Event Suppression is turned off at TEND.

SLIP keywords for transaction execution mode

Two keywords are available when specifying SLIP trap parameters for transaction execution mode:

- TXIGD
- TXPROG

TXIGD keyword for the SLIP command

The Transaction Execution Ignore Data (`TXIGD`) keyword for the SLIP command was introduced for transaction execution mode to avoid matches via the `DATA` keyword.

The problem with a SLIP trap that matches with the `DATA` keyword during transaction execution is that it can get rolled back, so by the time that SLIP sees the storage pointed to by the `DATA` keyword, it no longer matches. The best that can be done is to capture diagnostic data by using the following methods in the hope that this is the cause:

- Specifying the `TXIGD` keyword (Transaction Execution Ignore Data) on a SLIP trap allows a match if there is a match when the `DATA` keyword is ignored.
- Specifying the `NOTXIGD` keyword on a SLIP trap allows a match only when the `DATA` keyword matches.

The SLIP display of an active trap displays, when non-zero, the number of times that the SLIP trap was examined for a transaction execution event but did not match because of the `DATA` keyword. In a transaction execution case, this could be normal, because the stores were rolled back when the error or PER program interrupt occurred. This is referred to as the transaction execution *DATA filter mismatch count*. If the value is 255, the count of events could have exceeded 255.

The SLIP GTF record, at offset (decimal) 135, has a 1-byte count that is the transaction execution `DATA` filter mismatch count,

TXPROG keyword for the SLIP command

TXPROG is a type for the ERRTYP option. When specified, the SLIP trap matches only if the event was a program interrupt error event that occurred within transaction execution mode.

28.1.3 Interactions and dependencies

Transaction execution mode has the following software dependencies:

- Transaction execution mode is available if CVT bits CVTTX and CVTTXC are on.
- Transaction execution mode is not available for z/OS that is running under a virtual machine (VM).

There are no hardware dependencies.

These are the current users of transaction execution mode:

- Java, which uses transaction execution mode by using the z/OS 1.13 support
- z/OS, which uses transaction execution mode in JBB778H for Flash support

28.1.4 Installation

HLASM APARs PM49761 and PM66334 must be installed to have access to the transaction execution mode instructions and the print exit.

28.2 Offload time at task level

The TIMEUSED service is enhanced to accommodate requirements from IBM CICS and IBM DB2 software.

CICS had a requirement to measure CPU time on offload engines or to measure time while offload-eligible on a standard CP that was not available to unauthorized applications at a task level, or both. (Offload engines: IBM System z Integrated Information Processors, zIIPs, or System z Application Assist Processors, zAAPs.)

DB2 has a requirement for offload-eligible time on a standard CPU that was not available via the authorized interface.

To accommodate these two requirements, the TIMEUSED macro provides the following variants, which are available to all users:

- To satisfy the CICS requirement for offload times to be returned by TIMEUSED when running unauthorized, TIMEUSED can now return the following information:
  - Total time
  - Time on CP
  - Time on offload (zIIP or zAAP)
  - Time on CP when offload-eligible

  Using this macro requires the use of the extract CPU time (ECTG) instruction
- To satisfy the DB2 requirement for offload-eligible time on a standard CPU through the authorized interface, TIMEUSED can now return these values:
  - Time on offload
  - Time on CP

28.2.1 Invocation and use

For the CICS requirement for offload times to be returned by TIMEUSED when running unauthorized (problem state, user key), the following conditions apply:

- TIMEUSED ECT=YES, STORADR=x allows one or more of the following to be specified:
  - TIME_ON_CP=YES
  - OFFLOAD_TIME=YES
  - OFFLOAD_ON_CP=YES (this is offload-eligible time)

- STORADR names an 8-word area where the following data is returned:
  - Words 0 - 1: Total time
  - Words 2 - 3: Time on CP when `TIME_ON_CP=YES` is specified
  - Words 4 - 5: Offload time (unnormalized) when `OFFLOAD_TIME=YES` is specified
  - Words 6 - 7: Offload on CP when `OFFLOAD_ON_CP=YES` is specified
- TIMEUSED checks to see if the CVTECT1 bit in byte CVTOSLV8 of the CVT data area is on. If not, it issues RC=12 because the function is not available.

  In z/OS 2.1, it checks only whether SYSOSREL is set to at least z/OS 2.1, because we know that the function is available starting with this version. Because z/OS 2.1 requires a machine that supports ECTG, we know that we can use the function.

For the DB2 requirement for offload eligible time on a standard CP while authorized (supervisor state, key 0), these conditions apply:

- TIMEUSED `TIME_ON_CP=YES` without `ECT=YES` allows the use of `OFFLOAD_ON_CP=YES`.
- TIMEUSED checks the CVTOOCP bit and issues RC=12 if it is not on, because the function not available.

  In z/OS 2.1, it checks only whether SYSOSREL is set to at least z/OS 2.1, because we know that the function is available starting with this version.
- When `TIME_ON_CP=YES` with `OFFLOAD_ON_CP=YES`, the 64-bit GR2 output contains the offload on CP time in addition to what is normally returned with `TIME_ON_CP=YES`.

28.2.2 Migration and coexistence

Avoid using the variants unless the appropriate “feature bit” in the CVT is on. If you know you are running with z/OS 2.1 functions, you may assume that the bit is on.

These enhancements to the TIMEUSED service were made available in z/OS 1.13 through APAR OA38409.

28.3 Larger and extendable facility list

The Facilities List (FlceFacilitiesList) has room for only 128 facilities, but there might be more facilities than that.

z/OS 2.1 provides a larger facilities list that can be extended (ECVTFACL).

28.3.1 Invocation and use

The facilities list is obtained by the `STFLE` instruction. As a convenience, z/OS captures this during IPL to make it easily accessed by programs.

A area in z/OS 2.1, which the ECVTFACL field points to, is provided to access the entire facility list. The current area will continue to hold the first 128 facility bits. The address of ECVTFACL is `ECVT+X'354'` and is formatted under IPCS by using this command:

```
IP CBF ECVT
```

The area is mapped by the macro IHAFACL and consists of a header, followed by the list.

28.4 Tracing disabled interrupt exits

Disabled interrupt exits (DIEs) and their CPU times are not traced. This makes it difficult to diagnose performance problems or other errors that arise from misbehaving DIEs.

In z/OS 2.1, disabled interrupt exits are traced through SSRV 120.

28.4.1 Invocation and use

The exit from a timer DIE is traced within the system trace. SSRV SSID X'0120' is annotated as TimerDIE.

The following data is traced:

- PSW: Exit address
- Unique-1 and Unique-2: CPU time spent in the exit
- Unique-3 and Unique-4: STCKF value when DIE was entered

A DIE entry is not traced primarily because it provides no information that is not already in the system trace.

Example 28-6 shows an example of a DIE system trace entry.

Example 28-6 DIE system trace entry

```
0003-0001 00000000  SSRV   120 81573ED8  00000000 00027DDE CB5BDED1  TimerDIE
                                         2585096E
```

28.5 Entry table entries expanded

The limit of 128 entries in an entry table has been increased in z/OS 2.1 by allowing the architectural limit of 256 entries.

28.5.1 Invocation and use

The ETDEF macro now supports the architected limit of 256 entries in an entry table.

28.6 Support for clock values in the second UNIX epoch

The current TOD clock has 64 bits, where bit 51 has a value of one microsecond. It begins at 1900/01/01 00:00:00 and will wrap at 2042/09/17 23:53:47.370496.

Clock conversion routines need to handle “reasonable” dates in the future that extend beyond the first epoch, which ends in 2042.

In z/OS 2.1, STCKSYNC and CONVTOD have been enhanced to handle dates up to the end of the second epoch, year 2185. The enhancement caters for timestamps that are prefixed with `X'01'` as the first byte. This support extends the TOD clock to 2185/04/06/ 23:47:34.740992.

28.6.1 Invocation and use

The CONVTOD and STCKCONV services now support values up to the end of the second UNIX epoch, which ends on 2185/04/06/ 23:47:34.740992. This is an STCKE value with less than `X'02'` in the first byte.

Table 28-1 shows the format of the operand for Store Clock Extended (STCKE).

Table 28-1 128-bit TOD format

Zeroes	TOD clock	Programmable Field
0 8		112 127

The current hardware architecture does not support a non-zero value in the first byte. CONVTOD and STCKCONV now support a value of `X'01'` in the first byte.

A value greater than `X'01'` in the first byte will be rejected with return code `X'14'`.

Example 28-7 shows examples of CONVTOD and STCKCONV for a date and time in the second epoch.

Example 28-7 STCKCONV and CONVTOD examples

```
STCKCONV STCKEVAL=INAREA,CONVVAL=OUTAREA,TIMETYPE=DEC,                X
               DATETYPE=MMDDYYYY

INAREA  DC     X'0111A9B678656D582E00000000000000'   STCKE VALUE
OUTAREA DS     CL16          CONVERTED DATE/TIME VALUE

Result: OUTAREA = 09424319 15090000 07232052 00000000

CONVTOD CONVVAL=INAREA,ETODVAL=OUTAREA,TIMETYPE=DEC,              X
               DATETYPE=MMDDYYYY

INAREA   DC    X'0942431915090000072320520000000'   INPUT DATE/TIME
OUTAREA  DS    CL16     CONVERTED STCKE VALUE

Result: OUTAREA = 0111A9B6 78656D58 00000000 00000000
```

Note: The **`IPCS LTOD EXTENDED`** command already supports timestamps with `X'01'` in the first byte, for example:

```
IP LTOD 0111A9B678656D582E00000000000000 EXTENDED
07/23/2052 09:42:43.191509 STCKE  X'0111A9B6 78656D58 2E000000 00000000'
07/23/2052 09:42:18.191509 UTC    X'0111A9B6 608DE958 2E000000 00000000'
07/23/2052 05:42:18.191509 LOCAL  X'0111A980 BBA4E958 2E000000 00000000'
```

28.7 Reserve SVC entries to use yet maintain ABEND upon use

When an SVC entry is reserved, even if not to be used yet a routine must be provided and the system no longer issues the normal ABEND for an “undefined” SVC.

A function in z/OS 2.1 allows `IEASVCxx` to reserve an SVC while preserving the normal ABEND for an undefined SVC.

28.7.1 Invocation and use

The SVC interrupt handler issues a Fxx ABEND if SVCxx is issued when SVCxx is not defined. If you want to reserve the SVC for future use but not yet define the SVC routine, it is now possible while still preserving that characteristic.

A reserved entry is typically determined by programs as having the address set by z/OS for undefined SVCs: The address of IGCERROR within the nucleus.

It is now possible to define an entry in IEASVCxx by using the name IGCERROR. This indicates to z/OS that even though it is "defined," it still should get the ABEND Fxx upon use.

Note: The address for such a case will not be set to IGCERROR so that existing programs will think that the entry is not available to them.

28.8 ACCURACY keyword in CLOCKxx

A function has been added to z/OS 2.1 to monitor the accuracy of whether an STP network is synchronized with an external time source (ETS). The ACCURACY keyword in CLOCKxx specifies a threshold for the time deviation between the STP network and an external time source.

The external time source can be accessed in these ways:

- Dial-out service on the HMC
- NTP server connected via the IP network
- NTP server connected via a pulse-per-second CPC port

28.8.1 Installation

To enable the TOD Clock Accuracy Monitor, specify an ACCURACY value in the 1 - 60000 range in CLOCKxx. The system must be running in STP timing mode on an STP-only CTN with an external time source configured.

The ACCURACY keyword in CLOCKxx requires the syntax that is shown in Example 28-8. Specifying a value of 0 means that the ACCURACY function is inactive.

Example 28-8 ACCURACY keyword syntax

```
ACCURACY nnnnn
where nnnnn = 0 to 60000 milliseconds with a default of 0.
```

28.8.2 Invocation and use

The TOD Accuracy Monitor is enabled by a non-zero ACCURACY statement of the CLOCKxx PARMLIB member. As the default value for ACCURACY is zero, so it is disabled by default.

When CLOCKxx is processed at IPL time, and a value of 1 - 60000 is specified for a system running on an STP network that accesses an external time source, the ACCURACY function is activated. A timer queue element (TQE) is scheduled and requeued every 30 minutes to monitor the primary reference time (PRT) steering adjustment. This is the steering adjustment performed by STP to synchronize to the external time source.

If ACCURACY 0 is specified or defaulted to, or if an external time source is not configured, the the following message is issued at IPL:

```
IEA036I THE TOD CLOCK ACCURACY MONITOR IS NOT ACTIVE
```

If the ACCURACY value is 1 - 60000 and an external time source is configured, the following message is issued at IPL:

```
IEA034I THE TOD CLOCK ACCURACY MONITOR IS ACTIVE
```

If the TOD clock exceeds +/- the ACCURACY value, the following message is issued and then reissued every 30 minutes until the condition is corrected:

```
IEA032E TOD CLOCK ACCURACY LIMITS MAY HAVE BEEN EXCEEDED
```

These are the possible corrective actions:

- Allow STP to steer automatically to the external time source. STP will steer to the external time source if the STP time is within 60 seconds. The steering rate is 7 hours per second of adjustment.
- If the time difference is greater than 60 seconds or the time to steer out the difference will be too long, there are two possible actions:
  - Deconfigure the STP-only CTN and reinitialize with the correct time. This requires a Parallel Sysplex-wide outage for a multi-system sysplex running in STP timing mode.
  - Use a leap second adjustment to correct UTC time. This action is explained in detail in the *Server Time Protocol Recovery Guide*, SG24-7380, section 6.3: Synchronizing the CTN to an ETS when the time difference is greater than the 60-second threshold.

When the time difference is corrected the following message is issued:

```
IEA033I THE TOD CLOCK IS NOW WITHIN SPECIFIED ACCURACY BOUNDS
```

28.8.3 Migration and coexistence considerations

Toleration APAR OA37967 for z/OS Versions 1.11, 1.12, and 1.13 allows a CLOCKxx member with the ACCURACY keyword CLOCKxx to be processed at IPL, with a warning message if WARNUND is specified.

A CLOCKxx member with the following statements is used to show the behavior of a z/OS 1.1, 1.12, or 1.13 system when the ACCURACY keyword is specified:

```
OPERATOR NOPROMPT
TIMEZONE W.05.00.00
ETRMODE YES
ETRZONE YES
STPMODE YES
STPZONE YES
TIMEDELTA 10
ACCURACY 600
```

- Without OA37967 regardless of whether WARNUND is specified:

```
IEA599I CLOCK00 LINE    7: TIMEDELT STMT IGNORED. UNRECOGNIZED PARM.
IEA341A RESPECIFY CLOCK PARM OR PRESS ENTER
```

- With OA37967 but WARNUND not specified:

```
IEA599I CLOCK00 LINE    7: TIMEDELT STMT IGNORED. UNRECOGNIZED PARM.
IEA341A RESPECIFY CLOCK PARM OR PRESS ENTER
```

- With OA37967 and WARNUND specified:

```
ASA014I WARNING IN PARMLIB MEMBER=CLOCK00
'ACCURACY' IS UNDEFINED. PROCESSING CONTINUES DUE TO WARNUND.
DETECTING MODULE IS IEAVNP20. INPUT LINE:
ACCURACY 600
```

28.9 Documentation and reference

For more information, see the following publications:

- z/OS V2R1 Elements and Features web page (search by publication number):
  - *IBM z/Architecture Principles of Operation*, SA22-7832, for the description of the transaction execution mode facility and instructions
  - *MVS Assembler Services Reference Vol 2*, SA23-1373, for the transaction execution mode macro, IEATXDC
  - *MVS System Commands*, SA38-0666, for the transaction execution mode SLIP keywords
  - *z/Architecture Principles of Operation*, SA22-7832, particularly Chapters 5, 7, and 10
- *Server Time Protocol Recovery Guide*, SG24-7380

  http://www.redbooks.ibm.com/abstracts/sg247380.html?Open

29

Scheduler and Device Allocation

This chapter describes the enhancements, functions, and features that were introduced for the Job Scheduler and Device Allocation features with z/OS Version 2 Release 1 (2.1):

- 29.1, “Dynamic SYSDSN ENQ downgrade” on page 380
- 29.2, “Parallel batch recall” on page 382
- 29.3, “PARMDD keyword to pass a parameter of up to 32,000 characters” on page 384

© Copyright IBM Corp. 2014. All rights reserved.

29.1 Dynamic SYSDSN ENQ downgrade

IBM MVS Device Allocation feature has the responsibility of ensuring the integrity of data sets and devices. For data sets, it does this by using a GRS ENQ for each data set. When the JCL coder requests DISP=OLD, DISP=NEW, or DISP=MOD, Allocation requests an exclusive ENQ with the major name SYSDSN and a minor name of the data set. If the JCL coder uses the other disposition, DISP=SHR, Allocation obtains the same ENQ, but with shared control.

After it is obtained, the ENQ is not released until the last step in which it is used. So, for a job with five steps, if the first step codes a data set with DISP=and then uses the data set again in step 5 with DISP=SHR, the ENQ is held for all five steps. Further, the ENQ's level of control, initially exclusive in step 1, is never changed, even though step 5 needs only shared control. In this way, Allocation ensures that the data set integrity is not compromised in any way until the job that needed exclusive control gives up the data set or ends.

This is the behavior without the DSENQSHR function.

In batch mode, the SYSDSN ENQ is held exclusive throughout the job, as shown in Figure 29-1, even if later steps do not need exclusive control. This leads to other jobs that request the data set being hung until the original job ends with IEF099I/IEF863I.

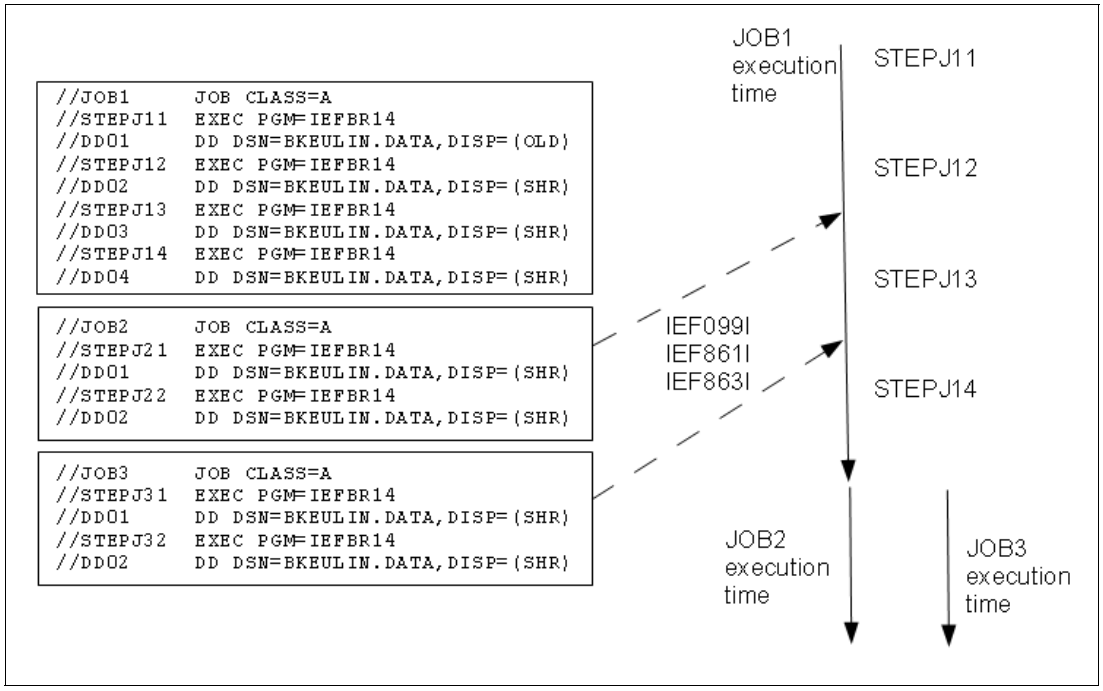


Figure 29-1 SYSDSN ENQ prior V2R1

In z/OS 2.1, you can downgrade SYSDSN ENQ when necessary by using the JCL keyword and a JES job class attribute, DSENQSHR. You can set these functions on a per job or per job class basis. This function enables better parallelism of batch job execution when multiple jobs need to share a data set resource. Figure 29-2 on page 381 shows the behavior of the DSENQSHR JCL keyword.

In the same example but using the DSENQSHR function, the SYSDSN ENQ is downgraded from exclusive control to shared control after step 1, because exclusive control is no longer needed for the data set. This allows other jobs that attempt to allocate the data set to have

access earlier than with previous limitations. This parallelism reduces the IEF099I, IEF861I, and IEF863I messages, which indicate that a particular job is waiting to use the data set.

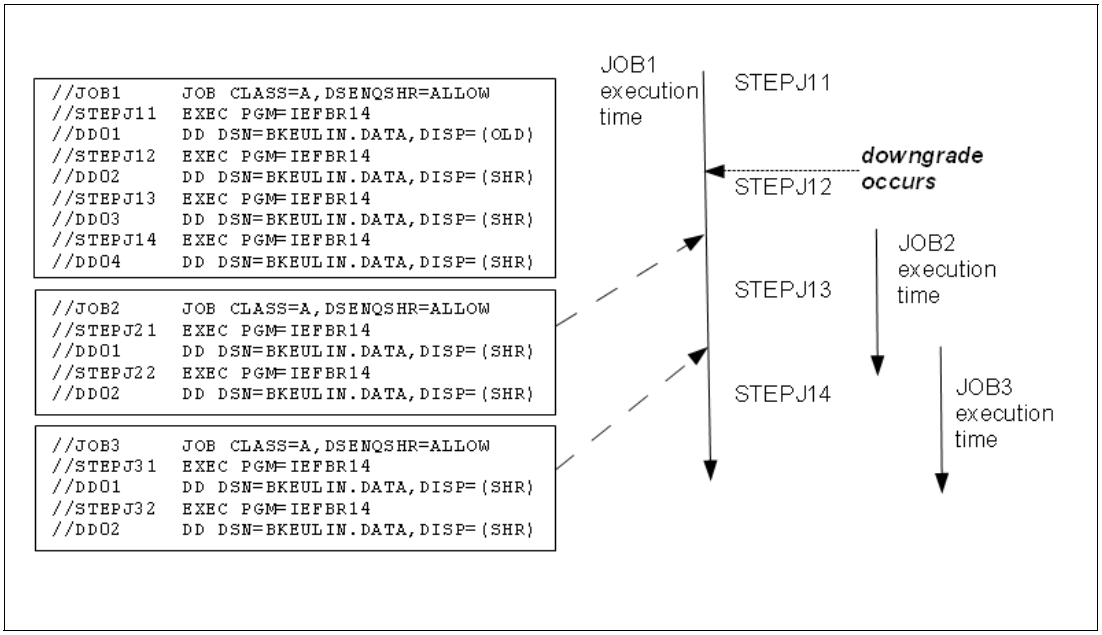


Figure 29-2 SYSDSN ENQ in z/OS 2.1 with JCL

29.1.1 Invocation and use

This support is enabled and customized by a JCL keyword, DSENQSHR, in the JOB statement. To allow this function on a per-job basis, use this code in your JCL:

```
//JOBNAME JOB DSENQSHR=<value>
```

Figure 29-3 shows the behavior of that function if you are coding JCL or JOBCLASS.

		JES JOBCLASS attribute		
		AUTO	*ALLOW*	DISALLOW
JCL	ALLOW	yes	yes	no
	USEJC	yes	*no*	no
	DISALLOW	no	no	no

Figure 29-3 Use matrix for DSENQSHR

DSENQSHR allows the function on a per-jobclass basis. To change the value, use this JES command:

```
$T JOBCLASS(<class>),DSENQSHR=<value>
```

Figure 29-4 on page 382 shows the use of the function. In DD1, we use data set BK.DS1 exclusively. After STEP1, there is no need for that, and the allocation in STEP2 changes to `DISP=SHR`, which means non-exclusive. With `DSENQSHR=ALLOW` in the Jobs JCL, an automatic downgrade of SYSZDSN ENQ is initiated.

```
//DSNSHRDN JOB MSGLEVEL=(1,1),DSENQSHR=ALLOW
//STEP1 EXEC PGM=IEFBR14
//DD1 DD DSN=BK.DS1,DISP=OLD   <-- Obtain ENQ EXCL for STEP1
//STEP2 EXEC PGM=IEFBR14
//DD2 DD DSN=BK.DS1,DISP=SHR   <-- Downgrade from EXCL to SHR after STEP1
//STEP3 EXEC PGM=IEFBR14
//DD3 DD DSN=BK.DS1,DISP=SHR
```

Figure 29-4 Downgrade example

In Figure 29-5, you can see a similar example for GDG use. In DDG1 STEP1, we allocate a GDG entry. Starting with STEP3, we allocate the data set as non-exclusive by using `DISP=SHR`. That causes an automatic GRS ENQ downgrade.

```
//DSNSHRDN JOB MSGLEVEL=1,DSENQSHR=ALLOW
//STEP1 EXEC PGM=IEFBR14
//DDG1 DD DSN=BK.GDG(+1),DISP=(NEW,CATLG), <-- Obtain ENQ
// SPACE=(TRK,1),VOL=SER=DASD01             EXCL for STEP1
//STEP2 EXEC PGM=IEFBR14
//DDG2 DD DSN=BK.GDG(+1),DISP=OLD
//STEP3 EXEC PGM=IEFBR14
//DDG3 DD DSN=BK.GDG(+1),DISP=SHR  <-- Downgrade after STEP2
```

Figure 29-5 Downgrade GDG JCL example

29.1.2 Migration and coexistence considerations

The JCL keyword requires JES2 at the z/OS 2.1 level.

Attention: With OA36878, the JCL keyword toleration is included with z/OS Versions 1.10 through 1.13. Any job that requests the function cannot be run on V1.13 and earlier level installations.

The function does not work in GRS Ring Mode and is not available for JES3.

29.2 Parallel batch recall

In batch processing the system does a Catalog Locate to gather data set info on every Allocation. This makes a DFSMShsm call to recall any data sets on a per data set basis. When many data sets are to be recalled, you need to wait for each one. This is because it uses serial allocations. No other batch jobs can use that initiator, so it results in delays in other jobs. z/OS 2.1 introduces a ALLOCXX keyword called BATCH_RCLMIGDS to do recalls in parallel, rather than serially. This allows a better parallelism of batch job execution when multiple data sets are migrated.

29.2.1 Invocation and use

There are two methods of using this function.

You can either customize your ALLOCxx member by adding the BATCH_RCLMIGDS(PARALLEL) statement, as shows.

Example 29-1 Adding the BATCH_RCLMIGDS(PARALLEL) statement

```
SETALLOC SYSTEM,BATCH_RCLMIGDS=<value>
```

Or you can use the **SETALLOC** system command to activate the function, as you see in Example 29-2.

Example 29-2 Example of SETALLOC

```
SETALLOC SYSTEM,BATCH_RCLMIGDS=PARALLEL
IEFA010I SETALLOC COMMAND SUCCESSFUL 486
BATCH_RCLMIGDS SET TO PARALLEL.
```

To see the current setting of BATCH_RCLMIGDS, you can use the DISPLAY ALLOC system command, as shown in Figure 29-6.

```
D ALLOC,OPTIONS
IEFA003I 15.11.23 ALLOC OPTIONS 509
SPACE           PRIMARY:          4
                SECONDARY:        24
                DIRECTORY:        0
                MEASURE:          AVEBLK
                BLKLNGTH:         8192
                ROUND:            NOROUND
                PRIM_ORG:         NONCONTIG
                RLSE:             RLSE
UNIT            NAME:             SYSALLDA
                UNITAFF:          SYSTEM DEFAULT
                REDIRECTED_TAPE:  TAPE
TIOT            SIZE:             32 (MAX DDS: 1635)
SDSN_WAIT       WAITALLOC:        NO
VOLUME_ENQ      POLICY:           WTOR
VOLUME_MNT      POLICY:           WTOR
SPEC_WAIT       POLICY:           WTOR
ALLC_OFFLN      POLICY:           WTOR
CATLG_ERR       FAILJOB:          NO
                ERRORMSG:         NO
2DGT_EXPDT      POLICY:           ALLOW
VERIFY_VOL      POLICY:           YES
SYSTEM          IEFBR14_DELMIGDS: LEGACY
                TAPELIB_PREF:     EQUAL
                REMIND_INTV:      90
                VERIFY_UNCAT:     FAIL
                TEMPDSFORMAT:     INCLUDELABEL
                MEMDSENQMGMT:     DISABLE
                BATCH_RCLMIGDS:   PARALLEL
                OPTCDB_SPLIT:     EXPLICIT
```

Figure 29-6 D ALLOC output

29.3 PARMDD keyword to pass a parameter of up to 32,000 characters

In previous releases of z/OS, the parameter string that can be specified for the existing PARM= keyword was limited to no more than 100 characters. Many programs need parameter strings that exceed that limit. Control block considerations prevent any significant compatible expansion beyond the 100 character limit. The PARMDD= keyword allows the name of a DD to be specified. The data set that is associated with the DD name can contain a parameter string of up to 32,760 bytes.

29.3.1 Invocation and use

The PARMDD= keyword is intended to replace use of the PARM= keyword for batch job step programs that require parameter strings in excess of 100 bytes.

Attention: The use of the PARMDD= keyword is mutually-exclusive with use of the PARM= keyword.

Figure 29-7 show a simple example of using the PARMDD keyword. The PARMDD statement points to the regular DD statement that contains the parameters for the calling program. In this example, input data will be read from MYPARMS.TEXT data set. The MYPARMS.TEXT data set is a Physical Sequential (PS) data set and can have fixed, fixed-block, variable, or variable-block record formats.

```
//LONGPARM JOB MSGLEVEL=1
//STEP0001 EXEC PGM=MYPROGRM,PARMDD=PARMINDD
//PARMINDD DD DSN=MYPARMS.TEXT,DISP=SHR
```

Figure 29-7 Sample PARMDD job using a PS data set

It is also possible to use partitioned data set members that have those record formats, as Figure 29-8 shows.

```
//LONGPARM JOB MSGLEVEL=1
//STEP0001 EXEC PGM=MYPROGRM,PARMDD=PARMINDD
//PARMINDD DD DSN=MYPARMS.LIBRARY(PARMS1),DISP=SHR
```

Figure 29-8 Sample PARMDD using a partitioned organization data set

The parameter string that will be passed in Figure 29-8 to the MYPROGRM program is contained in the MYPARMS.LIBRARY data set PARMS1 member.

The data set referenced by PARMDD= may also be a JES2 instream data set: DD * or DD DATA. (See Figure 29-9 on page 385.) The instream data set may contain symbols that are resolved by JES2.

```
//LONGPARM JOB MSGLEVEL=1
//STEP0001 EXEC PGM=MYPROGRM,PARMDD=PARMINDD
//PARMINDD DD *
Input parameters for MYPROGRM running on &SYSNAME
/*
```

Figure 29-9 Sample PARMDD using instream data

The parameter string that will be passed to the MYPROGRM program is this string:

```
Input parameters for MYPROGRM running on SC74
```

The data set referenced by PARMDD= may also be a UNIX System Service file DD PATH= (see Figure 29-10).

```
//LONGPARM JOB MSGLEVEL=1
//STEP0001 EXEC PGM=MYPROGRM,PARMDD=PARMINDD
//PARMINDD DD PATH='/tmp/myparms.txt'
```

Figure 29-10 Sample PARMDD using a z/OS UNIX file

The parameter string that will be passed to the MYPROGRM program is in the z/OS UNIX `/tmp/myparms.txt` file.

Input records with a fixed record format and records from JES2 instream data sets are examined for the presence of sequence numbers 8 contiguous numeric characters in the last 8 bytes of a record commonly produced by the TSO Interactive System Productivity Facility (ISPF) editor. If sequence numbers are found, the record length is adjusted so that the sequence numbers are not seen in subsequent processing.

Each input record (of any supported type) is examined for trailing blanks. If trailing blanks are found, the record length is adjusted so that the trailing blanks are not seen by subsequent processing. The parameter string passed to the program is formed by the simple left to right concatenation of the input records after sequence numbers (if any) and trailing blanks (if any) have been eliminated. Double ampersands in the parameter string are converted to single ampersands.

29.3.2 Interactions and dependencies

Authorized programs AC(1) that have been determined to be able to receive parameter strings in excess of 100 characters must be re-linked by the Binder with the LONGPARM attribute. Failure to do so causes the program to be terminated if the parameter string exceeds 100 characters.

29.3.3 Migration and coexistence

Programs that are expected to receive parameter strings through the PARMDD mechanism must be prepared to receive parameter strings up to 32760 bytes in length.

Attention: Many earlier programs assume that the parameter string can never exceed 100 bytes because it could never do so in the past.

There are no changes to the previously documented interface. On entry to the program, register 1 points to a one-element, fullword parameter list consisting of a 31-bit pointer. The 31-bit pointer points to a halfword length field that is located immediately in front of the parameter string area. The halfword length field contains the length of the following parameter string. The parameter string length can now be up to 32760 bytes. The PARM= keyword continues to behave as it did.

Important: There are no toleration or coexistence APARs or PTFs.

No migration actions are required. As longer parameter strings come into use, it will be necessary to use the Binder to re-link authorized programs that can handle long parameter strings with the LONGPARM attribute. Jobs that use PARMDD facilities must be run on z/OS 2.1 or later systems. When use of the PARMDD keyword is detected, the job is marked as having to run on a z/OS 2.1 or later level system and will be automatically scheduled to such a system by JES2.

30

Recovery Termination Manager

This chapter describes the following Recovery Termination Manager (RTM) enhancements that were introduced with IBM z/OS Version 2 Release 1 (2.1):

- 30.1, “CALLRTM TYPE=SRBTERM service” on page 388
- 30.2, “SdwaReleaseCode” on page 391
- 30.3, “64-bit FRR retry address in the system trace” on page 391
- 30.4, “Cross-memory failing instruction for the ESTAEX macro” on page 392
- 30.5, “SDWAMABD added to SDWANMFS” on page 392
- 30.6, “ASID added to the VRA for ABEND30D” on page 393
- 30.7, “Documentation and reference” on page 394

© Copyright IBM Corp. 2014. All rights reserved.

30.1 CALLRTM TYPE=SRBTERM service

TCBs typically lose control and are re-dispatched many times. CALLRTM TYPE=ABTERM is used to terminate a TCB the next time that it loses control. Traditionally, SRBs did not lose control and always ran to completion, so CALLRTM could not be used to terminate them.

With the advent of pre-emptible SRBs (including Pre-emptible, Client, and Enclave SRBs), we now have SRBs that lose control and are re-dispatched as TCBs are. These SRBs might also run longer than intended.

PURGEDQ can be used to terminate SRBs that have not run yet or that are stopped or suspended, but it waits for running SRBs. PURGEDQ cannot readily be used to target a specific SRB.

In z/OS 2.1, a form of CALLRTM, CALLRTM TYPE=SRBTERM, may be used to terminate a specific pre-emptible SRB. The pre-emptible SRB is uniquely identified by the token returned by IEAMSCHD using a keyword, SRBIDTOKEN.

Like TCBs, pre-emptible SRBs may temporarily protect themselves from CALLRTM by holding a lock or using STATUS SET,MC,PROCESS

CALLRTM TYPE=SRBTERM is processed asynchronously. This means that the target pre-emptible SRB can terminate after control is returned to the caller.

If the target pre-emptible SRB has not actually started running yet, it will be caught by the RTM's PURGEDQ, and its resource manager termination routine (RMTR) will receive control.

If the target pre-emptible SRB is stopped or suspended, it will be caught by the RTM's PURGEDQ and will be abended. It will have the completion and reason codes and RETRY= option in the SRBTERM request, rather than the usual PURGEDQ ABEND47B.

If the target pre-emptible SRB is running, it will eventually be caught and abended with the SRBTERM completion and reason codes and RETRY= option in one of the following places:

- The external first-level interrupt handler (FLIH) during normal interrupt processing
- SETLOCK release for a local lock (only if there are waiters)
- STATUS when resetting Process Must Complete

30.1.1 Invocation and use

CALLRTM TYPE=SRBTERM requires the SRBIDTOKEN and COMPCOD parameters. The SRBIDTOKEN value is returned by IEAMSCHD in the 16-byte output area pointed to by the SRBIDTOKEN parameter.

These optional parameters are supported:

- **REASON=,**
- **RETRY= (default 'YES')**
- **SYSTEM= (default 'YES')**

This form of CALLRTM requires a 144-byte work area address in GPR 13. Like CALLRTM TYPE=ABTERM, the work area is not used as a standard save area by RTM. GPR 13 is the only required input register.

SRBIDTOKEN

The SRBIDTOKEN uniquely identifies a pre-emptible SRB. It is created by IEAVSCHD.

This is the format of the SRBIDTOKEN:

```
SrbIdToken char(16) bdy(word) type
  SrbIdBytes0to3 char(4)     /* Currently always zeros    @P1C*/
  SrbIdWebAddress ptr(31)    /* Preemptable SRB's WEB address
  SrbIdSeq# bit(64)          /* Preemptable SRB's sequence number
```

IEAVSCHD copies the sequence number into the SRB's SSRX.

For validation, the WEB pointed to by the token is examined to determine whether it (still) represents a pre-emptible SRB. If so, the WEB points to an SSRB, which points to the SSRX. The sequence number in the token is then compared with the one in the SSRX

Environment requirements for the caller

- Minimum authorization: Key 0, Supervisor state
- Dispatchable unit mode: Task or SRB
- Cross memory mode: Any PASN, any SASN, any HASN
- Amode: 31 (amode 24 is not supported)
- ASC mode: Primary or Secondary
- Interrupt Status: Enabled or Disabled
- Locks: May be held, but are not required
- Control parameters:
  - For callers in Primary ASC mode, they must be in the Primary address space.
  - For callers in Secondary ASC mode, they must be in the Secondary address space.

Output register information

These are 64-bit. The high halves of 2 - 14 are preserved:

- 0 - 1 used as work registers
- 2 - 5 unchanged
- 6 used as a work register if you specify `REASON=`
- 7-13 unchanged
- 14 used as a work register
- 15 return or reason code

Additional information

The high-order bit of the reason code is defined as having a special meaning for SRBTERMs. When the issuer of CALLRTM turns this bit on, it is a signal that the issuer believes that an SVCDUMP will not be required for this abend.

Recovery routines can examine the high-order bit of the reason code when SDWASRBT is on as an aid in the decision about whether to capture an SVCDUMP. However, RTM does not do anything special for this bit. It is just a convention that has been put into place for this form of CALLRTM.

As with other forms of CALLRTM, a record is written to the system trace log to describe the CALLRTM TYPE=SRBTERM invocation.

Return codes

CALLRTM TYPE=SRBTERM returns a return and reason code in GPR 15 in the form xxxxxxyy, where xxxxxx is the reason code and yy is the return code.

The following return and reason codes are in hexadecimal format:

- 00000000
  Meaning: The SRBTERM request was processed successfully, and the SRB will be terminated at the next opportunity.
  Action: None.
- 00000104
  Meaning: The SRBIDTOKEN is no longer valid. This return code implies that the target SRB has already terminated.
  Action: None.
- 00000204
  Meaning: An SRBTERM request with RETRY=YES was issued against an SRB for which a previous SRBTERM request with RETRY=NO is still being processed. The older RETRY=NO SRBTERM will be honored rather than the RETRY=YES SRBTERM.
  Action: None.
- 00000108
  Meaning: The SRBIDTOKEN contains data that is not valid.
  Action: Ensure that the SRBIDTOKEN parameter points to a valid token that was returned by the IEAMSCHD service.
- 00000110
  Meaning: System error. The target SRB will terminate if it is running but might not terminate if it is suspended or stopped.
  Action: If the SRB does not terminate, reissue the SRBTERM request a reasonable number of times. If the SRB still does not terminate, report this error to IBM support.
- 00000210
  Meaning: System error. The target SRB will not be terminated.
  Action: Report this error to IBM support.

The RCVY SRBT system trace record

Example 30-1 is an example of an RCVY SRBT system trace record.

Example 30-1 RCVY SRBT system trace record

```
01 0023 005E77B0 *RCVY SRBT 89800CDE 86456000 00000014 00000000 000
                            00000000 042A6800 00000000 00000021 000
```

Where:

- 89800CDE is the address of the program that invoked CALLRTM.
- 86456000 is the requested completion code (system abend 456 in this example) with flags in the high-order byte.
- 00000014 is the requested reason code.
- 00000000 is the return code from RTM for this invocation.
- 00000000 042A6800 00000000 00000021is the SrbIdToken that was specified. For debugging purposes, notice that the second word contains the address of the target SRB's WEB control block.
- The right side of this record was truncated for clarity. It contains the standard things in an RCVY record: locks held, cross-memory information, and a time stamp.

30.1.2 Migration and coexistence

Support is available for z/OS 1.13 through APAR OA39392. Until the enablement PTF is installed, this support is not externally visible other than in the control block fields.

A flag, SDWASRBT, has been added to SDWAERRB in the SDWA that is mapped by IHASDWA, as shown in Example 30-2.

Example 30-2 SDWASRBT added

```
SDWASRBT EQU   X'20'   On, indicates that this abend was issued via CALLRTM
T                     YPE=SRBTERM
```

Note: SDWASRBT was not included as a factor in SDWANMFS (Not My Fault Summary), because SDWANMFS applies only to TCBs. When an SRB percolates (that is, no recovery routine can recover from the error) to a task after being terminated by an SRBTERM, RTM indicator SDWASRBM will be on, which was already included in SDWANMFS

The documentation updates are included in the enablement APAR OA39392 (see 30.7, “Documentation and reference” on page 394).

30.2 SdwaReleaseCode

The Pause release code might not be available to a program that has been abended while paused. A function is added to z/OS 2.1 to provide the Pause release code in SdwaReleaseCode.

30.2.1 Invocation and use

After a Release has been issued for a paused task, there is a window of time where an asynchronous abend (CALLRTM TYPE=ABTERM) can interrupt the task in a way that the release code is “lost” because it is not visible to the task’s recovery. After the abend, the task is not paused.

RTM now provides this release code in field, SdwaReleaseCode, so that it is visible to recovery. A bit, SdwaReleaseCodeValid, indicates when SdwaReleaseCode contains a value of interest.

SdwaReleaseCode, SdwaFain, and SdwaNMFS are all documented in the SDWA mapping macro, IHASDWA.

30.2.2 Migration and coexistence

SdwaReleaseCode is available in z/OS 1.13 via APAR OA40528.

30.3 64-bit FRR retry address in the system trace

Users of the limited 64-bit support for executable code need to see an entire FRR retry PSW. Because FRR processing does not support 64-bit addresses, the FRR retry address is placed in CVTBSM0F.

As of z/OS 1.13, users of the limited support for executable code above the 2 G bar have been able to arrange for FRR retry to that code by placing the actual retry address into 64-bit GPR 15 and requesting a retry to the address of CVTBSM0F.

The problem is that the FRR retry address is always the address CVTBSM0F, and it is valuable to see the actual retry address in the system trace.

A function is added to z/OS 2.1 to recognize a request to retry to CVTBSM0F for FRR retry processing. In that situation, the contents of 64-bit GPR 15 are traced as the retry address.

For normal FRR retry, the system trace entry contains a 64-bit address that was generated from the actual 32-bit retry address (the first character of this address is always zero).

30.3.1 Invocation and use

Example 30-3 shows part of an FRR retry system trace entry for 64-bit executable code.

Example 30-3 FRR retry system trace entry for 64-bit executable code

```
01 0024 005E7788 *RCVY  RTRY 00000048 0000006B  140C9000 00000009 00000000
                             07040001 80000000                    00000001
```

The system trace entry for FRR retry is documented in the RCVY trace entry section of the system trace chapter in *z/OS MVS Diagnosis: Tools and Service Aids*, GA32-0905.

30.4 Cross-memory failing instruction for the ESTAEX macro

It has been a long-standing concern that the failing instruction stream (SDWAFAIN) might be incorrect for an abend that has occurred in other than the Home address space for ESTAEX, ARR, and IEARR recovery routines.

In z/OS 2.1, RTM obtains the failing instruction stream from the correct address space for cases where the failing instruction stream was not previously obtained before an ESTAE-type recovery routine is going to receive control,

30.5 SDWAMABD added to SDWANMFS

The SDWANMFS bit was added in z/OS 1.12 to provide a summary bit for a Not My Fault list of SDWA indicators that show that a program has been abended for reasons beyond its direct control. Some recovery routines use this bit when determining whether to document that they were entered (for example, to take a dump or ask for a LOGREC record.)

When a subtask is detached by RTM because its mother task has failed, SDWANMFS might not be on. This could be because the subtask was presented with the mother task's abend information (where SDWANMF was not on) rather than the usual Detach ABEND13E (which would have caused SDWANMFS to be set on).

In z/OS 2.1, SDWAMABD is added to the list of indicators that are included in SDWANMFS to indicate that RTM has detached this subtask.

30.5.1 Invocation and use

Example 30-4 shows the explanation of the SDWANMFS of the SDWAERRB byte in IHASDWA, which now includes SDWAMABD.

Example 30-4 Not My Fault Summary as explanation of the SDWANMFS of the SDWAERRB

```
SDWANMFS EQU   X'40'   Not My Fault Summary -- indicates that this
*                     abend was received asynchronously (from an
*                     external source).  SDWANMFS may be examined
*                     as an alternative to checking individual abend
*                     codes when deciding whether to capture failure
*                     documentation or retry since when it is on the
*                     abend generally will not have been the fault
*                     of the program that received it.  SDWANMFS is
*                     available to Estae-type recovery and EUT FRRs
*                     running under TCBs.  It is set when any of the
*                     following abend indicators have been set:
*                     SDWAABTM - ABTERM indicator (note that Cancel
*                                and Detach are always ABTERMs)
*                     SDWAMABD - This TCB has been detached by RTM
*                                after its mother task abended
*                     SDWASRBM - An SRB has abended and percolated
*                                to this TCB
*                     SDWAIRB  - An IRB has interrupted this TCB
*                                and abended
*                     SDWAMCHK - A Machine Check has occurred
*                     SDWARKEY - A Restart was received
*                     SDWACTS  - An abend was Converted To Step
```

30.6 ASID added to the VRA for ABEND30D

master scheduler address space (ASID 1) attaches a subtask to call End of Memory resource managers for a terminating address space. If the subtask remains dormant for 4 or minutes, RTM assumes that the subtask has hung. Therefore, it takes an SVCDUMP for ABEND30D before issuing the abend to drive the EOM resource manager's recovery.

Before taking the SVCDUMP, RTM fills in the VRA of an SDWA with information about the failure so that it is available in the dump.

In z/OS 2.1, the ASID of the terminating address space is added to that VRA to aid in problem determination.

30.6.1 Invocation and use

Example 30-5 shows the VRA data for an abend S30D, which includes the terminating ASID information.

Example 30-5 VRA data for abend S30D

```
53003533 C9C5C1E5 E3D4D4E6 408881A2 06 *....IEAVTMMW has*
408485A3 8583A385 84408195 40C5D6D4    * detected an EOM*
409985A2 96A49983 85409481 95818785    * resource manage*
99408881 95875A35 1AC68189 93899587    *r hang!..Failing*
40D985A2 96A49983 8540D481 95818785    * Resource Manage*
997A40C9 10E9E3E3 D9D4C7D9 D7404040    *r: I.ZTTRMGRP   *
40404040 40500200 C9350DE3 C3C240C1    *     &..I..TCB A*
C4C4D9C5 E2E27A40 3808F0F0 F5C4C6C3    *DDRESS: ..005DFC*
F1F03512 E3859994 899581A3 89958740    *10..Terminating *
C1E2C9C4 7A403804 F0F0F2F8 00000000    *ASID: ..0028....*
```

30.7 Documentation and reference

The following documentation updates are included in the enablement APAR OA39392:

- Update to the IEAMSCHD service described by the *z/OS MVS Authorized Assembler Services Reference EDT-IXG*, SA23-1373 to include the SrbIdToken parameter
- Update to the CALLRTM service described by the *z/OS MVS Authorized Assembler Services Reference ALE-DYN*, SA23-1372 to include the TYPE=SRBTERM parameter
- A section about terminating pre-emptible SRBs in the “Using a service request block (SRB)” chapter of the *z/OS MVS Authorized Assembler Services Guide*, SA23-1371
- Information about using CALLRTM TYPE=SRBTERM in the “Invoking RTM” chapter of the *z/OS MVS Authorized Assembler Services guide*, SA23-1371
- A description of the RCVY SRBT system trace record in the *z/OS MVS Diagnosis: Tools and Service Aids*, GA32-0905

31

Service aids

This chapter describes the following service aids and enhancements that were introduced in IBM z/OS Version 2 Release 1 (2.1):

- 31.1, “IPCS append dump directory” on page 396
- 31.2, “IPCS SYSTRACE enhancement” on page 398
- 31.3, “System trace branch and mode health check” on page 400
- 31.4, “SLIP health check” on page 403
- 31.5, “SLIP zero address detection support” on page 405
- 31.6, “Documentation and reference” on page 409

© Copyright IBM Corp. 2014. All rights reserved.

31.1 IPCS append dump directory

As systems become capable of supporting more storage, the use of large real and large virtual storage has greatly increased the size of z/OS dumps. Increased dump size also means increase time to initialize the dump. The initialization process of a large dump can take several hours. This warrants enhancements to at least reduce the number of times that the initialization must execute.

A function in z/OS 2.1 provides a way to avoid repeated dump initialization. The data description records from the dump directory, which are built during COPYDUMP, are appended to the dump directory.

The dump and dump directory are sent to IBM Support, and the dump directory is then extracted and used by IBM Support personnel without having to initialize the dump. The function is available for SADMP, SDUMP, SYSMDUMP, and TDUMP.

This speeds up the process to access the dump and avoids having to manage sending the dump and a separate directory data set.

31.1.1 Invocation and use

INITAPPEND is a option for the output data set or DD name. If INITAPPEND is specified, COPYDUMP creates the dump directory entries, performs dump initialization for the output dump data set, and adds the dump directory entries to the end of the dump data set.

Figure 31-1 shows the syntax of the INITAPPEND keyword for **COPYDUMP** command.

```
IPCS COPYDUMP subcommand change

{ OUTDSNAME(outds-spec )|OUTDATASET(outds-spec )|ODS( outds-spec) }
{ OUTFILE(outdd-spec )|OFILE(outdd-spec)|OUTDDNAME(outdd-spec) }
[ INDSNAME(dsname_list)|INDATASET(dsname_list)|IDS(dsname¦list) ]
[ INFILE(ddname_list|IPCSINDD)| IFILE(ddname_list|IPCSIND)
INDDNAME(ddname_list|IPCSINDD) ]
[ DEFAULT ]
[ SPACE(nnnn[,mmmm]|1500,1500) ]
[ CLOSE | LEAVE ]
[ ASIDLIST(ddddd[,ddddd]) ]
[ JOBLIST(j1[,j2][,j3]..[,jn]) ]
NOSKIP | SKIP[(nnn|1|EOF)] ]
[ NOCLEAR | CLEAR ]
[ EASYCOPY ]
where
outds-spec := dsname [ INITIALIZE | INITAPPEND] | NULLFILE
[ COMPLEMENT(dsname | NULLFILE) ]
[ FULL(dsname [INITIALIZE]) | NULLFILE]
outdd-spec := ddname [ INITIALIZE | INITAPPEND] [ COMPLEMENT(ddname) ]
[ FULL(ddname [INITIALIZE]) ]
```

Figure 31-1 COPYDUMP syntax including the INITAPPEND keyword

The following rules apply for INITAPPEND:

- IINITAPPEND is not used with COMPLEMENT data sets.
- If both INITIALIZE and INITAPPEND are specified, INITAPPEND is used.
- If INITAPPEND is specified for an appended dump, the dump description will be refreshed by using the current dump directory (DDIR).
- You can specify either OUTDSNAME or OUTDDNAME when the dump data set is being exported by using COPYDUMP.
- If you use COMPLEMENT with INITIALIZE or INITAPPEND, the following message is issued during the command parsing:

  ```
  EXTRANEOUS INFORMATION WAS IGNORED: INITIALIZE (or INITAPPEND).
  ```

- If you use INITAPPEND for NULLFILE, the following message is issued:

  ```
  BLS18259I NULLFILE cannot be specified with INITAPPEND.
  ```

When the dump data set is imported via DSNAME (using IPCS 0. or SETDEF DSNAME), the imported dump description contains the dump data set name. When the dump data set is imported via DDNAME (using SETDEF DDNAME), the imported dump description contains the DD name.

If the original dump data set is renamed, and the appended dump description name contains a different name from the current dump data set name. Then, when the DDIR is imported, the dump description name in the DDIR is changed to contain the current dump data set name.

IPCS adjusts the dump data set name when the appended dump directory is imported into a DDIR.

Note: The appended dump directory does not support HFS data sets.

Example 31-1 shows a JCL example of using COPYDUMP with INITAPPEND

Example 31-1 JCL using COPYDUMP with INITAPPEND

```
//ACD01A01 JOB  EEA,GO,MSGCLASS=A,MSGLEVEL=(1,1)
//GTFPRNT EXEC PGM=IKJEFT01,DYNAMNBR=20,REGION=3000K
//SYSPROC DD DSN=SYS1.SBLSCLI0,DISP=SHR
//IPCSTOC DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSEXEC DD DSN=RSTLSVA.REXX.EXECS,DISP=SHR,DCB=BLKSIZE=32760
//     DD DSN=ISF.SISFEXEC,DISP=SHR DCB=BLKSIZE=27920
//SYSUT1  DD UNIT=SYSDA,SPACE=(CYL,(50,50))
//SYSTSIN DD *
 %BLSCDDIR DSNAME('SYS4.DDIR') VOLUME(VSAM01) NOENQ
 IPCS NOPARM
 SETDEF LIST NOCONFIRM
 COPYDUMP INDS('MVSSVA.LI2142.TDUMP1.R14') -
  OUTDS('MVSSVA.LI2142.TDUMP1.R14.AD' INITAPPEND)
 END
/*
```

Example 31-2 shows a JCL example of importing the copied dump.

Example 31-2 JCL to import a copied dump

```
//AIM01I01 JOB EEA,GO,MSGCLASS=A,MSGLEVEL=(1,1)
//GTFPRNT EXEC PGM=IKJEFT01,DYNAMNBR=20,REGION=3000K
//SYSPROC DD DSN=SYS1.SBLSCLI0,DISP=SHR
//IPCSTOC DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//SYSSNAP DD SYSOUT=*
//SYSEXEC DD DSN=RSTLSVA.REXX.EXECS,DISP=SHR,DCB=BLKSIZE=32760
// DD DSN=ISF.SISFEXEC,DISP=SHR DCB=BLKSIZE=27920
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(50,50))
//SYSTSIN DD *
%BLSCDDIR DSNAME('SYS4.DDIR') VOLUME(VSAM01) NOENQ
IPCS NOPARM
SETDEF DSN('MVSSVA.LI2142.TDUMP1.R14.AD') LIST NOCONFIRM
L 0
END
/*
```

The IPCS panel for COPYDUMP is updated to add the INITAPPEND option. Figure 31-2 shows the COPYDUMP panel accessed via option 3.2 - Utility, COPYDUMP. This is the second panel that is displayed after the dump data set name is entered on the initial panel.

```
----------------------------- Copy an MVS Dump Data Set -----------------------
Command  ===>

Enter Output  parameters to copy an MVS dump data set.
Use ENTER to copy data, END to terminate.

OUTPUT DATA SET: NULLFILE    INITIALIZE    INITAPPEND   (Enter "/" to select)
 NAME                                                   (DSNAME or DDNAME)

FULL DATA SET:   NULLFILE    INITIALIZE                 (Enter "/" to select)
 NAME                                                   (DSNAME or DDNAME)

COMPLEMENT DATA SET:         NULLFILE
 NAME                                                   (DSNAME or DDNAME)



 NAME TYPE       SPACE (No. of Records)    EXECUTION        DEFAULT
 1 1. DSNAME      PRIMARY   1500           2 1. BATCH
   2. DDNAME      SECONDARY 1500             2. FOREGROUND
```

Figure 31-2 IPCS COPYDUMP panel

31.2 IPCS SYSTRACE enhancement

With HiperDispatch enabled, work can be managed across fewer logical processors to maximize the benefits of the processor cache structures. This reduces the amount of CPU

time that is required to execute work. In HiperDispatch mode, logical processors are categorized, based on their share of a physical processor, as Vertical High (VH), Vertical Medium (VM), or Vertical Low (VL). VL processors are parked by the supervisor on WLM request when they are not needed to handle the partition's workload. Parked CPUs are in a long wait state and do not dispatch work. Therefore, system trace can contain trace entries that are from hours or sometimes days before the time of the dump. It is time-consuming to prepare a table that shows the start and end time for each CPU in the configuration, because there is no way to do so today in IPCS.

IPCS SYSTRACE is enhanced in z/OS 2.1 to provide a table that shows the start and end time and other related information for each CPU.

31.2.1 Invocation and use

A keyword, STATUS, has been added to the SYSTRACE subcommand. When STATUS is specified a table of processor-related information, as seen at the time of the dump i is displayed in the system trace. The table includes:

- CPU number
- CPU type
- Start and end time n the system trace
- Parked status and processor share when in HiperDispatch mode

In addition the system trace status is displayed

The following rules apply for the STAUS keyword for the SYSTRACE subcommand:

- Can be specified only with TIME and TTCH.
- If specified with keywords other than TIME and TTCH, following message is issued:

  ```
  STATUS OPTION IS NOT COMPATIBLE WITH ANY SYSTRACE OPTIONS EXCEPT TIME AND TTCH
  ```

- Time information is displayed in hexadecimal, local, or GMT format, depending on the value specified for the TIME parameter.

Figure 31-3 on page 400 shows the output from command IP SYSTRACE STATUS TI(LO) for a system in mode HiperDispatch with two online CPUs.

```
TRACE services are available
TRACE is active
ST=(On,0001M,00002M) AS=On BR=Off EX=On MO=Off

Currently active snapshots:   00000001
SNAPTRC high water mark:      00000001
SNAPTRC high water mark TOD:  05/20/2013 16:16:47.335098

The earliest timestamp in SYSTRACE is from CPU 0001: 05/21/2013 10:25:00.09711
The latest   timestamp in SYSTRACE is from CPU 0001: 05/21/2013 10:28:19.67580
TRACE data reporting from all CPUs starts at         05/21/2013 10:28:14.26626
TRACE data reporting from all CPUs ends at           05/21/2013 10:28:19.67580

------+------+------+------+----------------------------+--------------------
 CPU  | Type | Pol  | Park | SYSTRACE First Local Time  | SYSTRACE Last Local
------+------+------+------+----------------------------+--------------------
 0000 | CP   | Med  | No   |  05/21/2013 10:28:14.266265 | 05/21/2013 10:28:19
 0001 | CP   | Low  | No   |  05/21/2013 10:25:00.097118 | 05/21/2013 10:28:19
```

Figure 31-3 Output from SYSTRACE STATUS TI(LO)

This systrace status line:

```
ST=(On,0001M,00002M) AS=On BR=Off EX=On MO=Off
```

Indicates that:

- System trace is On
- System trace buffer size for each processor is 1M byte
- Total system trace buffer size for all processors is 2M bytes
- AS=on/off for the implicit address space tracing
- BR=on/off for branch tracing
- EX=on/off for explicit tracing
- MO=on/off for Mode tracing

The CPU table displays information about the status of the online CPUs where

- Type: Indicates CP for normal CP, ZI for zIIP or ZA for zAAP
- Pol: Indicates the amount of physical processor share for this processor, High, Med or Low, when the system is in HiperDispatch mode
- Park: Indicates if this processor was parked at the time of the dump, Yes or No, when the system is in HiperDispatch mode.
- SYSTRACE first time and SYSTRACE last time: Indicates the first and last time this processor was seen in SYSTRACE.

  This information is displayed in Hex, local, or GMT format, depending on the value of the TIME parameter with the default of Hex.

31.3 System trace branch and mode health check

System tracing allows the option of tracing branch instructions, such as BALR, BASR, BASSM, and BAKR, along with other system events. Issue TRACE ST,BR=ON/OFF to turn on or turn off the branch tracing function of system.

Mode tracing consists of recording occurrences of entering and leaving 64-bit mode. Issue TRACE ST,MODE=ON/OFF to turn on or turn off the mode tracing function of system.

Branch and mode tracing use CPU cycles. This tends to significantly increase the number of trace entries being generated, which increases the size for the trace buffers. Therefore, leaving branch and mode tracing on can affect system performance. It is best to leave the tracing on for only a short time to solve a specific problem.

In z/OS 2.1, the following two checks in the IBM Health Checker for z/OS alert for branch tracing or mode tracing that has been active for so long that someone might have forgotten that it is still on:

```
CHECK(SYSTRACE_BRANCH) OWNER(IBMSYSTRACE)
CHECK(SYSTRACE_MODE) OWNER(IBMSYSTRACE)
```

This helps to avoid system performance impact due to active branch or mode tracing.

31.3.1 Invocation and use

These two health checks provide the following functions:

- SYSTRACE_BRANCH checks whether system trace is using branch tracing (TRACE BR=ON, for certain branch instructions) and has been active for longer than a specified duration.
- SYSTRACE_MODE checks whether system trace is using mode tracing (TRACE MODE=ON, for AMODE 64 on/off) and has been active for longer than the specified time.
- The TRACE ST command processing saves the time when BR=ON/MODE=ON started. These times are used to calculate the duration of active time.

Figure 31-4 shows the default keywords for the SYSTRACE_BRANCH health check.

```
UPDATE,
 CHECK(SYSTRACE_BRANCH) OWNER(IBMSYSTRACE)
  ACTIVE
  DEBUG(OFF)
  INTERVAL(4:00)
  SEVERITY(LOW)
  PARM('TIME(DAYS,07)')
  DATE('20130509')
  REASON('An active, but not needed, SYSTRC BRANCH=ON
          option can cause degraded system performance.')
```

Figure 31-4 SYSTRACE_BRANCH health check

Figure 31-5 on page 402 shows the default keywords for the SYSTRACE_BRANCH health check.

```
UPDATE,
 CHECK(SYSTRACE_MODE)
  OWNER(IBMSYSTRACE)
  ACTIVE
  DEBUG(OFF)
  INTERVAL(4:00)
  SEVERITY(LOW)
  PARM('TIME(DAYS,07)')
  DATE('20130509')
  REASON('An active, but not needed, SYSTRC
          MODE=ON option can cause degraded
          system performance.')
```

Figure 31-5 SYSTRACE_MODE health check

The default values can be overridden by either a POLICY statement in the HZSPRMxx PARMLIB member or a **MODIFY** command.

For both the STYSTRACE_BRANC and SYSTRACE_MODE health checks, the following conditions apply:

- TIME(DAYS|HOURS,nn) allows the time threshold to be adjusted ('TIME(DAYS,07)').
- There are only two values for TIME: TIME(value1,value2):
  - value1 = DAYS|HOURS
  - value2 = m, where m=1 - 9999
- The default is TIME(DAYS,7).
- If BRANCH or MODE option has been ON for longer than the defined time, an exception message is issued.
- The health checks run every 4 hours by default: INTERVAL(4:00).
- The health check SEVERITY is LOW.

IPCS SYSTRACE STATUS output

The IPCS SYSTRACE STATUS output is changed to display the latest activation time of BR and MODE options if they were ever activated.

Example 31-3 shows examples of the SYSTRACE STATUS output that displays the current status of SYSTRACE_BRANCH and SYSTRACE_MODE.

Example 31-3 SYSTRACE STATUS output

```
TRACE services are available
TRACE is active
ST=(On,0001M,00001M) AS=On BR=On EX=On MO=On
The latest BR option activation time: 05/13/2013 16:47:56.308400
The latest MODE option activation time: 05/13/2013 16:47:56.308400

TRACE services are available
TRACE is active
ST=(On,0001M,00001M) AS=On BR=Off EX=On MO=Off

TRACE services are available
TRACE is active
```

```
ST=(On,0001M,00001M) AS=On BR=On EX=On MO=Off
The latest BR option activation time: 04/23/2013 01:35:38.180362
The latest MODE option activation time: 04/23/2013 01:36:14.538849

TRACE services are available
TRACE is active
ST=(On,0001M,00001M) AS=On BR=Off EX=On MO=Off
The latest MODE option activation time: 04/23/2013 04:07:20.493909
```

Messages

Figure 31-6 shows the messages associated with the SYSTRACE_BRANCH health check.

```
IEAH801E System trace Branch option has been active for a longer time than the
checkowner_or_installation-defined duration. It was activated on datemdy4 at
timehmsc.

IEAH802I System trace Branch option is on, but it has not been active as long
as the checkowner_or_installation-defined duration. It was activated on
datemdy4 at timehmsc.
datemdy4 – The local date when the first Branch option was set on
timehmsc – The local time when the first Branch option was set on

IEAH803I System trace Branch option is off
```

Figure 31-6 Messages for SYSTRACE_BRANCH health check

Figure 31-7 shows the messages associated with the SYSTRACE_MODE health check.

```
IEAH804E System trace Mode option has been active for a longer time than the
checkowner_or_installation-defined duration. It was activated on datemdy4 at
timehmsc.
IEAH805I System trace Mode option is on, but it has not been active as long as
the checkowner_or_installation-defined duration. It was activated on datemdy4
at timehmsc.
datemdy4 – The local date when the first Mode option was set on
timehmsc – The local time when the first Mode option was set on
IEAH806I System trace Mode option is off
```

Figure 31-7 Messages for SYSTRACE_MODE health check

31.4 SLIP health check

SLIP PER traps can have adverse effects on system performance. If a SLIP PER trap has been active for too long, it might have been forgotten.

In z/OS 2.1, a health check for the IBM Health Checker for z/OS issues alerts for a SLIP PER trap that might have been forgotten about. This avoids degraded system performance that is due to an active SLIP PER trap that is not needed.

31.4.1 Invocation and use

The CHECK(IBMSLIP,SLIP_PER) checks whether a SLIP PER trap has been continuously active for longer than the threshold specified.

Figure 31-8 shows the default keywords for the SLIP_PER health check.

```
CHECK(SLIP_PER)
 OWNER(IBMSLIP)
 INTERVAL(4:00)
 SEVERITY(LOW)
 PARM('TIME(DAYS,30)')
 DATE('20130509')
 REASON('Your reason for making the update.')
```

Figure 31-8 SLIP_PER health check

The default values can be overridden by either a POLICY statement in the HZSPRMxx PARMLIB member or a **MODIFY** command.

The SLIP_PER health check accepts the following parameters:

- `TIME(DAYS,`*n*`)` or `TIME(HOURS,`*n*`)`, with a default of 30 days.

  This identifies the length of time that a PER trap must be active before the exception is raised, where *n* can range from 1 - 9999.
- `INTERVAL(`*hh:mm*`)`, with a default of 4 hours.
- The health check SEVERITY is LOW.

Messages

Figure 31-9 shows the messages that is associated with the SLIP_PER health check.

```
IEAH101E SLIP PER trap id has been active for longer than the
checkowner_or_installation-defined duration. It was activated on datemdy4 at
timehmsc.

IEAH102I SLIP PER trap id is active, but it has not been active as long as the
checkowner_or_installation-defined duration. It was activated on datemdy4 at
timehmsc.

IEAH103I No SLIP PER trap is active
```

Figure 31-9 SLIP_PER Health Checker messages

DISPLAY SLIP command changes

z/OS 2.1 introduces changes to the DISPLAY SLIP command to display relevant PER traps. This aids in checking PER trap status after the SLIP_PER health check has issued an exception message.

The **DISPLAY SLIP** command has two options:

- **DISPLAY SLIP,PER**
  This displays a summary with the name only of the PER traps indicate the status:
  - DISABLED
  - ENABLED but IGNORE
  - ENABLED and non-IGNORE
- **DISPLAY SLIP,ENIPT**
  This displays the name of the enabled non-ignore trap, if there is one.

31.5 SLIP zero address detection support

Zero address detection (ZAD) is a PER function that was introduced as a feature with the IBM zEnterprise 196 (z196) machine.

A ZAD event is a PER program interrupt. It is caused by execution of an instruction that accesses (stores or fetches) storage by using an operand address that was formed from a general register that contains a zero when the PSW PER bit is on and ZAD is enabled.

z/OS 2.1 provides a function for SLIP that allows PER SLIP traps to detect ZAD events. The detection of ZAD events is enabled by a SLIP trap of type ZAD. A ZAD trap can help detect errors that involve unintentionally referencing the PSA. All instructions are monitored for ZAD events. You can limit ZAD events only by limiting the address spaces or jobs for which PER is active.

In addition, you can limit the ZAD events that match a ZAD type PER SLIP trap by using other SLIP filtering keywords, such as ADDRESS, LPAMOD, NUCMOD, PVTMOD, and so on.

31.5.1 Avoiding non-error ZAD events

A ZAD event might or might not be an error. When an instruction is executed and the conditions are met, a ZAD event occurs. It is up to you to decide whether this is an error or not, as is the case for any SLIP trap match.

It can also be useful to get rid of the non-error “noise” events. Then, when you are looking for events in your own modules, you expect to find none, so you know that any event needs to be dealt with.

These are examples of non-error ZAD events, and how you can avoid them is discussed in the subsections that follow:

- Accessing page zero of a data space
- Processing the PSA of the current CPU
- Assembler code that uses the index register as the base address

Accessing page zero of a data space

You are using a data space that starts at page 0. When the first page is used, the starting origin is a pointer with a value of 0, which is used as a base register when accessing the storage so that an ALET can be specified in the corresponding access register.

Many components choose to avoid using page 0. In fact, they cause page 0 to be hidden so that any use of page 0 prompts a program check. This approach can work for just about any program.

You can use IARVSERV ChangeAccess,Target_View=HIDDEN to hide page 0. If you use this IARVSERV invocation, be sure that you use it only when the origin returned by DSPSERV was 0.

¦z/OS 2.1 introduces the HIDEZERO=YES option on DSPSERV CREATE. If you know that you are running z/OS 2.1, it is simpler to use this than to use the “extra” IARVSERV call.

Processing the PSA of the current CPU

When a program is looping through the PSAs of each processor, it typically picks up the PSA address to use from PCCAPSAV. But when processing “this CPU,” an address of 0 is used.

There are various approaches to resolving this problem. For example, you can place a value of *n* in a register (such as 8). Then, you can have the assembler USING statement be `PSA+`*n*, and the other processors would use `PCCAPSAV+`*n* to share that USING instruction.

Assembler code that uses the index register as the base address

In Assembler, it is easy to have the base and index registers backwards, so that the base register is really the “array offset,” which might be 0, and the index register is the address of the array.

The machine’s address calculation will work fine, but this produces a ZAD event when the array offset is 0.

31.5.2 Invocation and use

We suggest that you look for ZAD events within your own product’s code when you are testing. You can help others if you find errors in their code, but code owners might not appreciate getting flooded with information about non-error events. Therefore, if you see a ZAD event outside of your scope and you do not have good reason to believe it is an error, it is probably best not to do anything about it.

You should not report a ZAD event to IBM unless you have reason to believe that it is an error or if you are providing a ZAD report to help diagnosis a problem that you are working on with IBM Support.

To use ZAD, you must be running on a z196 system. SLIP will reject the ZAD operand if it does not find ZAD available. This can happen even on a z196 is running under the IBM zVM operating system in some cases. It is better to run this on a test system to avoid unnecessarily affecting your production workload.

If the ZAD facility is not available, the following message is issued:

```
IEE737I THE facility FACILITY IS NOT AVAILABLE
```

Running the IEAVTSZR program

The IEAVTSZR program is included in the SYS1.LINKLIB with z/OS 2.1. It produces reports to help identify programs that cause ZAD events.

The procedure shown in Example 31-4 on page 407 can be used to produce the ZAD event reports.

Example 31-4 IEAVTSZR procedure

```
//*****************************************************************
//** SIZE MAY BE NK, NNK, NNNK, NM, NNM, NNNM. IT IS THE AMOUNT TO *
//** BE USED FOR INDIVIDUAL INSTRUCTION DATA. IT IS ADDED TO THE *
//** AMOUNT NEEDED TO CAPTURE DATA FOR EVERY ADDRESS SPACE. *
//** *
//** OP=FREE TO INDICATE THAT YOU'RE DONE, OR ANYTHING *
//** ELSE TO INDICATE *
//** - ALLOCATE IF NOT YET ALLOCATED *
//** - PRODUCE A REPORT BASED ON THE CURRENT DATA *
//** - AFTER PRODUCING A REPORT, CLEAR THE DATA *
//*****************************************************************
//IEAVTSZR PROC SIZE=1M,OP=DATA,STATS=YES,SYSOUT=*
//IEAVTSZR EXEC PGM=IEAVTSZR,TIME=1440,REGION=0M,
// PARM='OP=&OP,SIZE=&SIZE,STATS=&STATS'
//SYSPRINT DD SYSOUT=&SYSOUT
```

The following steps show how to produce the ZAD event report:

1. Add the IEAVTSZR procedure to the SYS1.PROCLIB or the equivalent.
2. START IEAVTSZR.

   The first run initializes an area to record the information to be gathered.
3. SETPROG LPA,ADD, DSN=SYS1.LINKLIB,MOD=IEAVTSZE,FIXED.

   Alternatively, you could use the FLPA system parameter with an IEAFIXxx PARMLIB member to have done this at IPL.
4. SLIP SET,ZAD,A=AEXIT,AEXIT=IEAVTSZE,ID=ZAD1,PL=50,OK,E
   AEXIT is SLIP functionality that is not supported for other use. You must use this syntax:

   `A=AEXIT,AEXIT=IEAVTSZE`
5. Run your programs.
6. START IEAVTSZR.

   This time, it writes to SYSPRINT a report about all of the ZAD events and resets things to allow more iterations of running and START IEAVTSZR, as appropriate.
7. START IEAVTSZR,OP=FREE.
8. When testing has completed (no need if you are IPLing), review the IEAVSZR reports for ZAD events within the programs that you own.

IEAVSZR report

The IEAVSZR report identifies the following information:

- Where the event occurred (both by address and by module name if available)
- How many times the ZAD event occurred
- What instruction was issued

You can add ASIDSA=SA to the SLIP trap to avoid getting hits for data space stores. The ZAD event will still occur for a data space store, but SLIP will filter it out. The report flags data space stores with a “D” so that the data space stores are easily identified when reported.

ZAD on SLIP command

You can add filtering to the SLIP trap to avoid matching for unrelated events via LPAMOD, NUCMOD, ASID, ADDRESS, and so on. The ZAD events still occur, but SLIP filters them.

You can specify ZAD similarly where you use SA. All of the other SLIP keywords work the same as if you had specified SA but had not specified the range of the storage alteration, with the exception of the address of the instruction that is causing the event. RANGE is not valid for a ZAD trap, but you can use ADDRESS to filter based on the address of the instruction that is causing the event.

ZAD report information

The information in the IEAVSZR report for a ZAD event includes the following information:

- Job name
- ASID in hex
- Address of the instruction
- Number of occurrences
- Module name (as can be determined) but not CSECT within module
- Offset within module (you can use AMBLIST to locate which CSECT within the module)
- Whether the event was in a data space or not
- Instruction text (hex), for example: BF3F2000BF3F
- Decoded instruction text, for example: ICM R3,15,0(R2)

Sample ZAD report

Example 31-5 shows an example of the IEAVSZR ZAD report.

Example 31-5 IEAVSZR ZAD report

```
SLIP Action Exit IEAVTSZR Event Report 08/28/2012 09:10:44.70

IEAVTSZR SLIP Trap ID=ZAD1 Duration: 0000005F_6095B7E5 Seconds: 100
CP AddrSp #: 00000000_00000004 DataSp #: 00000000_00000000
zAAP AddrSp #: 00000000_00000000 DataSp #: 00000000_00000000
zIIP AddrSp #: 00000000_00000000 DataSp #: 00000000_00000000
PerSecPerCPU: 00000000
PerSecPerCPU_CP: 00000000
PerSecPerCPU_zAAP: 00000000
PerSecPerCPU_zIIP: 00000000
Max PRCNTLIM: 0

TC (ASID 001C): CP AddrSp #: 00000000_00000004 DataSp #: 00000000_00000000
zAAP AddrSp #: 00000000_00000000 DataSp #: 00000000_00000000
zIIP AddrSp #: 00000000_00000000 DataSp #: 00000000_00000000

Pvt Search - Ser/Ok: 2 Ser/Fail: 0 UnSer/Ok: 0 UnSer/Fail: 0
Used Entries: 2 ( 0%) Longest Chain: 1

JobName ASID Address Count ModName Offset Dsp InstrucText Decoded Instruction Text
TC      001C 00007054 2    TEST    00000054    BF3F2000BF3F ICM R3,15,0(R2)
TC      001C 00007058 2    TEST    00000058    BF3F2004181D ICM R3,15,4(R2)
```

31.6 Documentation and reference

For more information, see these IBM publications:

- *IBM Health Checker for z/OS User's Guide*, SA22-7994
  http://www.ibm.com/support/docview.wss?uid=pub1sa22799407
- PDF files available with the z/OS Information Center for z/OS 2.1:
  http://pic.dhe.ibm.com/infocenter/zos/v2r1/index.jsp
  - *z/OS MVS Dump Output Messages*, SA23-1378
  - *z/OS MVS IPCS Commands*, SA23-1382
  - *z/OS MVS IPCS User's Guide*, SA23-1384
  - *z/OS MVS System Commands*, SA38-0666
  - *z/OS MVS System Messages Volume 6*, SA38-0673
- *z/OS MVS Data Areas, Volume 6*, GA76-0440
  http://publibz.boulder.ibm.com/epubs/pdf/iea3d601.pdf

32

Hardware Instrumentation Services

This chapter describes the Hardware Instrumentation Services (HIS) enhancements, functions, and features that were introduced with for IBM z/OS Version 2 Release 1 (2.1):

- 32.1, “HIS enhancements” on page 412
- 32.2, “HISSERV interface” on page 412
- 32.3, “SMF 113 Changes” on page 423
- 32.4, “Documentation and Reference” on page 424

© Copyright IBM Corp. 2014. All rights reserved.

32.1 HIS enhancements

The Hardware Instrumentation Services (HIS) function provides a framework for collecting and recording hardware event data for IBM System z10 and later servers. This function collects hardware event data for processors in SMF record type 113, subtype 2, and z/OS UNIX output files.

z/OS 2.1 makes using these services easier. This chapter describes the following changes and gives a high-level review of the function:

- Hardware performance analysis
- CPU Measurement Facility
- Hardware Instrumentation Services address space z/OS 2.1 enhancements:
  - The HISSERV Interface
  - SMF 113 changes

Note: You can use HIS only for IBM System z10 or later machines. You must start the HIS address space on each system where you want to collect data. Then, you must configure and activate HIS data collection (hardware event counters and sampling facilities) for each system by issuing the **`F hisproc`** command.

32.2 HISSERV interface

The HISSERV service can be used by authorized products to retrieve hardware instrumentation event and sampling data directly from storage. Although individual products can query the service for any subset of event types, the sampling parameters are global for every product that intends to collect sampling data. The global sampling parameters or service parameters are set by the most recent **`MODIFY HIS,BEGIN`** or **`MODIFY HIS,SERVICE`** command. A *profiler* chooses how often to collect event type data and how much time is spent processing sampling data, but the frequency is global for all profilers that are collecting sampling data. Any of these might cause a performance degradation to the system and should be carefully chosen.

z/OS 2.1 provides a programming interface called HISSERV to allow safe and real-time access to the CPU/MF data. You can now use the Enhanced Monitor Facility on IBM zEnterprise 196 (z196) and later servers to capture z/OS counters. In previous releases of z/OS, an HIS user had to use the **`MODIFY HIS,BEGIN`** command to allow the collection of instrumentation data and then record it into z/OS UNIX files and SMF. The HIS profiler now uses the HISSERV service to collect instrumentation data.

Starting with V2.1, you can use the HISSERV service to create your own profilers for customized collection and processing of instrumentation data and the following purposes:

- Collect and process sampling data
- Collect and process event counter data
- React to certain system state change events

Note: You can have more than one profiler active at the same time.

To collect instrumentation data, you must register with the HISSERV service to declare the caller's intentions. This registration identifies the user as a profiler of the system. A unique token identifies the profiler on subsequent HISSERV calls. A previously defined profiler of the system receives the following benefits:

- Notifications with sampling data through a callback mechanism
- Ability to query events such as counters
- Notifications from a callback mechanism of important profiling events, for example underlying system state changes

A profiler of the system must explicitly declare the intention to stop profiling the system. In Table 32-1 provides an overview of the HIS service functions that you can use. The macro definition files are in SYS1.MACLIB in your system.

Table 32-1 HIS macro definitions

Service name	Description
HISSERV	Executable macro to invoke the service
HISYSERV	Mapping macro for answer areas, input areas, return codes
HISYEXIT	Mapping macro for generic exit routine parameter area
HISYSMPX	Mapping macro for sampling exit routine parameter area
HISYCTRS	Equates for counter definitions

32.2.1 Invocation and use

Before issuing the HISSERV macro, the caller does not have to place any information into any general-purpose register (GPR) unless using it in a register notation for a particular parameter or using it as a base register.

Before issuing the HISSERV macro, the caller does not have to place any information into any access register (AR) unless using it in a register notation for a particular parameter or using it as a base register.

Important: To use that interface, the caller must be in Supervisor state or PKM 0-7.

After calling HISSERV and returning to the calling program, registers are set as shown in Table 32-2.

Table 32-2 Output register use

Register	Description
R0	Reason code if R15 is not zero
R1	Used as work register by the system
R2-R13	Unchanged
R14	Used as work register by the system
R15	Return code

Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service and restore them after the system returns control.

The HISSERV service provides an interface to begin profiling and retrieve instrumentation data from the system. There are currently two types of instrumentation data:

Events Events are recorded at the CPU level. As events occur, they are captured and recorded to be queried at any interval determined by the software. Events are grouped into event types, which can be enabled and disabled independently.

Sampling At predetermined intervals, a sample that represents the current state of a CPU is stored in a Sampling Data Buffer (SDB). As SDBs are filled, software is notified to allow the software to process the full SDBs. The SDBs are then cleared to be reused by the hardware. There are different sampling types that can be enabled and disabled independently. However, a profiler can indicate only the intention to receive sampling data. The sampling frequency and which sampling types are enabled are determined by the service parameters specified in an **`F hisproc,SERVICE`** command or an **`F hisproc,BEGIN`** command.

Specifically, with HISSERV you can do the following tasks:

- Query for event information, such as determining which events and event types are available. (REQUEST=QUERY,TYPE=EVENT)
- Query for sampling information such as the sampling interval and which sampling types are available. (REQUEST=QUERY,TYPE=SAMPLE)
- Query for statistics about whoever is currently profiling the system. (REQUEST=QUERY,TYPE=PROFILERS)
- Begin profiling the system, indicating to the system the intention of collecting one or more event types or sampling data. Requests that require a PROFILETKN must first use this to identify the profiler as wanting to profile the system. (REQUEST=PROFILE,ACTION=START)
- Stop profiling the system. The PROFILETKN is no longer usable. When the last profiler stops profiling the system, any unnecessary resources are released. (REQUEST=PROFILE,ACTION=STOP)
- Query for event data provided by the service. Requires a PROFILETKN. (REQUEST=QUERY,TYPE=EVENTDATA).

Important: The HISSERV service is enabled only when the HIS address space has been initialized. The dynamic exit HIS.SERVSTAT can be used to be notified when the service has been enabled or disabled.

HISSERV REQUEST=PROFILE,ACTION=START

The HISSERV service request ACTION=START declares a user's intention to start profiling the system. You must pass the following information to the service:

- EVENT: Which event types you intend to query
- SAMPLE=YES|NO: Whether to be called back with sampling data
- NAME: An externally identifiable name
- EXITRTN: A module to get control for callbacks

After the successful registration, you will get the following value:

- OUTPROFILETKN: The token to use for subsequent requests

The **`HISSERV REQUEST=PROFILE,ACTION=START`** is also used to register an exit routine that will process the sampling data. This exit routine is called when sampling data is available or when

an underlying system state change occurs. You must use **HISSERV REQUEST=PROFILE,ACTION=STOP** when it is time to stop the exit.

Important: Exit routines that involve processing of sampling data must be written with performance in mind. Runtime analysis of the sampling data should not be done under the sampling exit.

We recommend copying the sampling data provided to the exit routine and doing additional processing and analysis under a different work unit.

HISSERV REQUEST=PROFILE,ACTION=STOP

In addition to the HISSERV ACTION=START request, you must unregister that service after finishing your work. The **HISSERV REQUEST=PROFILE,ACTION=STOP** can be used to declare your intention to stop profiling the system. The only parameter that you must pass is the PROFILTKN that you received from the HISSERV START request.

HISSERV REQUEST=QUERY

The length of the storage passed in through ANSAREA on an HISSERV REQUEST=QUERY request must have an ANSLEN of at least `HisAns_kLength`. Depending on the request, if not enough storage is provided for a successful request, the service returns this warning code:

```
Hisserv_kRsnWarn_AnsAreaSmall
```

Retrying the query with an ANSLEN of HisAns_LengthRequired will eventually be successful.

To ensure future compatibility, you must recognize that the amount of storage obtained for a specific request might not be enough. Also, you must provide code in case the reason of Hisserv_kRsnWarn_AnsAreaSmall is returned, even if the request's ANSLEN appears to be static in nature.

Note: If you experience many buffer overflows, there are three things to do:

1. Increase the BUFCNT with F HIS,SERVICE,BUFCNT=xxxx,SAMPTYPE=PERSIST.
2. Decrease the SAMPFREQ with F HIS,SERVICE,SAMPFREQ=xxxx,.
3. Eliminate one or more profilers.

Example HISSERV

Example 32-1 provides a skeleton example of using the HISSERV function. This demonstrates profiling of basic counters.

Example 32-1 HISSERV profiler example

```
*********************************************************************
*                                                                   *
* project      : ITSO redbook SG24-5034                             *
*                                                                   *
* function     : Example of using HISSERV                           *
* requirements :                                                    *
*                                                                   *
* history      : 03.06.2013 Lutz       first creation               *
*                                                                   *
*********************************************************************
         PRINT NOGEN
HISTEST  CSECT
```

```
HISTEST  AMODE 31
HISTEST  RMODE ANY
         YREGS
         BAKR  R14,R0
         LR    R12,R15
         USING HISTEST,R12
*--------------------------------------------------------------------*
*  get storage                                                       *
*--------------------------------------------------------------------*
         LA    R2,LWORK
         STORAGE OBTAIN,                                            +
               LENGTH=(R2),                                         +
               LOC=24,                                              +
               ADDR=(R11)
         USING MYDSECT,R11
*--------------------------------------------------------------------*
*  Start HIS Service                                                 *
*--------------------------------------------------------------------*
         MODESET MODE=SUP,KEY=ZERO
         HISSERV REQUEST=PROFILE,                                   +
               ACTION=START,                                        +
               EXITRTN=HISEXIT,                                     +
               NAME=HISNAME,                                        +
               SAMPLE=NO,                                           +
               EVENT=HisEvnTyp_BasicCtrs,                           +
               OUTPROFILETKN=HISTKN,                                +
               RETCODE=HISRC,                                       +
               RSNCODE=HISRSN
*--------------------------------------------------------------------*
*  Loop                                                              *
*--------------------------------------------------------------------*
         LA       R5,1              * load loop counter             *
         MVC      WAIT60,=F'6000'   * set wait interval to 60 sec   *
HISLOOP  EQU      *                 * Begin loop                    *

         HISSERV REQUEST=QUERY,                                     +
               TYPE=EVENTDATA,                                      +
               PROFILETKN=HISTKN,                                   +
               CPUMASK=ALL,                                         +
               ANSAREA=HISANSW,                                     +
               ANSLEN=HISANSWL,                                     +
               RETCODE=HISRC,                                       +
               RSNCODE=HISRSN

* ------------------------------------------------------------------
*
* Place code here to analyze the output HISSERV REQUEST=QUERY
*
* ------------------------------------------------------------------

         STIMER WAIT,BINTVL=WAIT60   * go sleep for a minute         *
         BCT    R5,HISLOOP           * End loop                     *
*--------------------------------------------------------------------*
*  Stop HIS Service                                                  *
*--------------------------------------------------------------------*
```

```
         HISSERV REQUEST=PROFILE,                                      +
               ACTION=STOP,                                            +
               PROFILETKN=HISTKN
         MODESET MODE=PROB,KEY=NZERO
*------------------------------------------------------------------------*
*  release storage and exit program                                      *
*------------------------------------------------------------------------*
HISEND   LA    R2,LWORK
         STORAGE RELEASE,                                              +
               LENGTH=(R2),                                            +
               ADDR=(R11)
         SR    R15,R15
         PR
*------------------------------------------------------------------------*
HISNAME  DC CL8'ITSONAME'
HISEXIT  DC CL8'ITSOEXIT'
MYDSECT  DSECT
SAVE     DS 18F
WAIT6O   DS F
HISRC    DS F
HISRSN   DS F
HISTKN   DS CL16'O'
HISANSW  DS XL(HisEvn_Len+HisEvnCtr_Len)
HISANSWL DS AD(HisEvn_Len+HisEvnCtr_Len)
*HISANSW  DS CL1024
*HISANSWL DS D'1024'
         DS    OD
LWORK    EQU   *-MYDSECT
         HISYSERV
         HISYEXIT
         HISYSMPX
         END   HISTEST
```

The important thing to be aware of when using HISSERV is the calculation of the Answer Area called HISANSW in that program. You must declare enough memory for your request. The size depends on the number of online CPUs at a particular timothy HisAns_kLength value is the minimum value for the answer area. To allocate the correct size for the answer area, we recommend the following steps:

1. Allocate some storage for your request, minimum HisAns_kLength.
2. Execute the query.
3. If you get return code 04 with reason 401 (not enough memory), take these actions:
   a. Free storage from the failed request.
   b. Allocate storage length as HisEvnData_LengthRequired.
   c. Reissue the query.

After successfully issuing the query, you must inspect the data returned in HISANS. A sample CNT file is shown in Figure 32-1 on page 418.

```
HIS019I EVENT COUNTERS INFORMATION VERSION 4
FILE NAME: SYSHIS20130605.113525.000.CNT
COMMAND: MODIFY HIS,BEGIN
LOSS OF SAMPLE DATA ALERT: NO   SAMPLE BUFFER OVERFLOW COUNT: 0
LOSS OF COUNTER DATA ALERT: NO
STATE CHANGE: NO
MODEL: 2827-719  SEQCODE: 0000000000000XXXX
COUNTER VERSION NUMBER 1: 1   COUNTER VERSION NUMBER 2: 3

COUNTER SET= BASIC
COUNTER IDENTIFIERS:
   0: CYCLE COUNT
   1: INSTRUCTION COUNT
   2: L1 I-CACHE DIRECTORY-WRITE COUNT
   3: L1 I-CACHE PENALTY CYCLE COUNT
   4: L1 D-CACHE DIRECTORY-WRITE COUNT
   5: L1 D-CACHE PENALTY CYCLE COUNT

START TIME: 2013/06/05 11:35:25  START TOD: CB77A3C944D3A906
END TIME:   2013/06/05 11:45:25  END TOD:   CB77A605794F9D80
COUNTER VALUES (HEXADECIMAL) FOR CPU 00 (CPU SPEED = 5504 CYCLES/MIC):
   0-   3 00000017702EBDBC 000000048FFD9C43 00000000188275B2 000000067C4C5E5F
   4-   7 0000000015030906 0000000B539E18D0 ---------------- ----------------
```

Figure 32-1 Sample CNT File

32.2.2 Installation

Before you start the HIS data collection, you need to enable authorization to the sampling facilities and counter set types that you want to use through the Support Element (SE) console. For an example for that action, see Figure 32-2 on page 419. In the example we enable partition A1 for all available counters.

For information about how to set up the authorization of the sampling facilities and counter sets, see the *Support Element Operations Guide, Version 2.12.0, IBM zEnterprise System*, SC28-6920.

In addition, with the enhanced-monitor facility hardware released with z196 machines, the HIS function expands into a z/OS software event data collector that will be used by IBM for improved problem analysis. The z/OS event counters are viewed as an additional hardware counter set, and there is no authorization required to use the hardware.

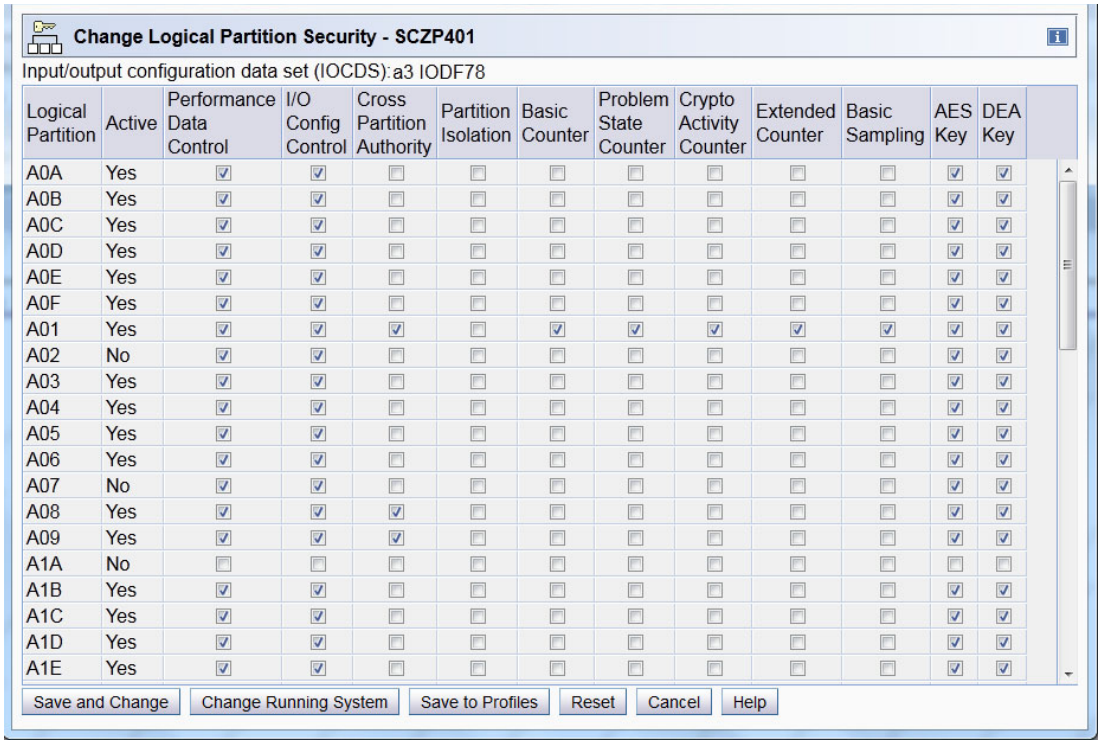


Figure 32-2 Enable Sampling dialog

In Figure 32-3 you will see a sample start procedure of HIS. You simple put these JCL in your proclib concatenation of your system.

```
//HIS     PROC
//HIS     EXEC PGM=HISINIT,REGION=0K,TIME=NOLIMIT
//*
//* You can specify an MVS command file to contain some or all of
//* the settings for the instrumentation run. The command file
//* must have fixed-length LRECL=80 records.
//*
//* If this option is desired,
//*  1. Replace 'DUMMY' below with the name of the MVS command
//*     file and its DISPOSITION.
//*  2. Specify the DDNAME keyword on the 'MODIFY HIS' command.
//*     For example:
//*       "MODIFY HIS,BEGIN,DDNAME=CMDFILE1"
//*
//CMDFILE1 DD DUMMY
//CMDFILE2 DD DUMMY
//SYSPRINT DD SYSOUT=*
```

Figure 32-3 HIS start procedure

Figure 32-4 on page 420 shows the startup of the HIS address space.

```
--- WEDNESDAY, 05 JUN 2013 ----
IEF695I START HIS      WITH JOBNAME HIS IS ASSIGNED TO USER IBMUSER , GROUP
$HASP373 HIS      STARTED
HIS002I HIS INITIALIZATION COMPLETE
```

Figure 32-4 HIS startup

Figure 32-5 shows the start of HIS sampling.

```
09.13.14 STC07508  HIS033I HIS SERVICE PARAMETER(S) ACCEPTED
09.13.14 STC07508  HIS011I HIS DATA COLLECTION STARTED
09.23.14 STC07508  HIS022I HIS DATA COLLECTION IS ENDING.  871
   871              OUTPUT FILE PREFIX: SYSHIS20130606.091314.
   871              TITLE= ''
09.23.14 STC07508  HIS012I HIS DATA COLLECTION ENDED
```

Figure 32-5 Start HIS sampling

Use the MODIFY hisproc command to manage collection of hardware event data for System z10 or later machines.

```
F HIS,BEGIN,TT='Lutz Counters',PATH='/HIS',CTRONLY,CTR=(B,E),SI=SYNC,
SC=SAVE
HIS033I HIS SERVICE PARAMETER(S) ACCEPTED
HIS011I HIS DATA COLLECTION STARTED
```

Use F hisproc,BEGIN to configure and start a run of data collection and F hisproc,END to stop the run. You must explicitly start each run of hardware data collection. You cannot set up data collection to run automatically. During a run of hardware data collection, the system writes the data to z/OS UNIX output files and to SMF record type 113, subtype 2.

The MODIFY Command writes the raw data to SMF record type 113 at the start, the end, and defined intervals during the data collection run, and writes different types of data to the z/OS UNIX output files at the end of the run. In Figure 32-6 on page 421 you see the output of the DISPLAY HIS command to show the current settings of HIS samplings.

```
D HIS
HIS015I 11.42.38 DISPLAY HIS 854
HIS      006B ACTIVE
COMMAND: MODIFY HIS,BEGIN,TT='Lutz Counters',PATH='/HIS',CTRONLY,CTR=(
         B,E),SI=SYNC,SC=SAVE
START TIME:  2013/06/12 11:34:02
END TIME:    ----/--/-- --:--:--
COMPLETION STATUS: --------
FILE PREFIX: SYSHIS20130612.113402.
COUNTER VERSION NUMBER 1: 1   COUNTER VERSION NUMBER 2: 3
COMMAND PARAMETER VALUES USED:
 TITLE=  Lutz Counters
 PATH=   /HIS
 COUNTER SET= BASIC,EXTENDED
 DURATION= NOLIMIT
 CTRONLY
 DATALOSS= IGNORE
 STATECHANGE= SAVE
 SMFINTVAL= SYNC
HISSERV STATUS: ACTIVE
 EVENT
  AUTHORIZED= BASIC,PROBLEM-STATE,CRYPTO-ACTIVITY,EXTENDED,ZOS
  ENABLED= BASIC,EXTENDED
 SAMPLE
  AUTHORIZED= BASIC
  ENABLED= NONE
  BUFCNT=    6 (PAGES/PROCESSOR)
  SAMPFREQ= 800000 (SAMPLES/MINUTE)
 PROFILER
  NAME      START                 QUERY             SAMPLE
  HISPROF   2013/06/12 11:34:02   00:01:50.925302   ------
```

Figure 32-6 D HIS output

MODIFY HIS command changes

The MODIFY HIS,SERVICE command allows changes to sampling parameters without having to start/stop a collection run. The following option were introduced with z/OS 2.1:

- SAMPTYPE: Indicate which sampling types should be collected
- SAMPFREQ: Indicate the sampling frequency samples are collected
- BUFCNT: Indicate the number of sampling buffers per CPU
- DDNAME: the referenced DD must only contain sampling parameters

In Figure 32-7 on page 422 you see an overview about the enhanced DISPLAY HIS command.

```
F hisproc,{BEGIN | B}
                    [,{TITLE | TT} ='textdata']
                    [,PATH='pathname']
                    [,{DURATION | DUR}=duration_value]
                    [,{CTRSET | CTR }={COMPLETE |
                                       SOFTWARE |
                                       HARDWARE |
                                       (ctr1,ctr2,...ctrn)}]
                    [,{DDNAME | DD}=ddname]
                    [,{CTRONLY | MAPONLY}]
                    [,{MAPVERBOSE | MAPV}]
                    [,{BUFCNT | BUF}={bufcnt | PERSIST ]
                    [,{SAMPFREQ | SF}=freq | PERSIST ]
                    [,{SAMPTYPE | ST}=samptype | PERSIST | NONE]
                    [,{DATALOSS | DL}={IGNORE | STOP}]
                    [,{MAPASID | MAS}={ALL | (asid1,asid2,...asidn)}]
                    [,{MAPJOB | MJOB}=(job1,job2,...jobn)]
                    [,{STATECHANGE | SC}= {SAVE | STOP | IGNORE}]
                    [,{SMFINTVAL | SI}={SYNC | intv}]
          {END | E}
          {SERVICE | S}
                      [ [,{DDNAME | DD}=ddname]
                        [,{BUFCNT | BUF}={bufcnt| PERSIST]
                        [,{SAMPFREQ | SF}=freq | PERSIST]
                        [,{SAMPTYPE | ST}=samptype| PERSIST | NONE] ]
                       [,{REFRESH | REFR}]
```

Figure 32-7 syntax of MODIFY HIS command

Also with this release, the MODIFY HIS,SERVICE,REFRESH was introduced to instructs the system to refresh the HISSERV service. That allows the service to begin using counter sets/sample types recently authorized through HMC. This command option is mutually exclusive with all other MODIFY HIS,SERVICE parameters.

Restriction: Changing the service parameters causes the service to update itself and Instrumentation data collection temporarily suspended for all profilers

Possible profiler's exit routines receives an UpdateStart callback followed by an UpdateDone callback.

All sampling parameters, either on MODIFY HIS,BEGIN or MODIFY HIS,SERVICE accept a value PERSIST. That means to keep the current setting. If you don't use these options, a parameter will reset it to its default.

The option CTRSET now collects either counters from hardware events or software events we show you in Table 32-3 on page 423.

Table 32-3 CTRSET option values

CTRSET	Description
CTRSET=ALL	Deprecated, is now an alias for CTRSET=HARDWARE. Did not want to include software counters in this group for compatibility reasons and avoid undesirable overhead in installations where CTRSET=ALL is being used.
CTRSET=HARDWARE	All hardware counter sets and any counter set which incurs minimal performance overhead will be added here. This is equivalent to CTRSET=ALL
CTRSET=SOFTWARE	All software counter sets, currently only the z/OS counter set. Any counter set which might incur a performance overhead will be added here
CTRSET=COMPLETE	All counter sets with the risk there may be a performance overhead. counter sets may be added here.

DISPLAY HIS changes

There are slightly changes in the command status section of the DISPLAY HIS command. The command shows what command was last issued for the HIS. Furthermore we have three sections as you can see in bold in Figure 32-6 on page 421. Profiler, this section shows what is currently in effect for the HISSERV service. The Profiler query time is average time between queries and the Profiler sample time is average number of microseconds it took to handle a sample callback.

32.2.3 Interactions and dependencies

The caller should include the HISYSERV macro to get equate symbols for the return and reason codes. The caller must include the HISYSERV macro to get a mapping of the output area provided by way of the ANSAREA parameter for REQUEST=QUERY. The caller must include the HISYEXIT macro to get a mapping of the parameter area passed to the exit routine specified by the EXITRTN parameter for REQUEST=PROFILE,ACTION=START requests. The caller must include the HISYSMPX macro to get a mapping of the parameter area passed to the exit routine specified by the EXITRTN parameter for REQUEST=PROFILE,ACTION=START requests, when the profiler requests sampling data (SAMPLE=YES).

32.3 SMF 113 Changes

The SMF Type 113 mapping is today defined in macro HISYSMFR which is found in SYS1.MACLIB.

32.3.1 Invocation and use

The fields are summarized in the following sections:

- SMF113_2_CTR: Main Section
  - SMF113_2_CSOF: Offset to counter set section
  - SMF113_2_CSON: Number of counter set sections
  - SMF113_2_CSLN: Length of counter set section
  - SMF113_2_CDOF: Offset to counter data section
  - SMF113_2_CDON: Number of counter data sections (total for all counter sets)
  - SMF113_2_CDLN: Length of counter data section
- SMF113_2_CSS: Counter Set Section
  - SMF113_2_CST: Counter Set Type (Basic, Problem, Crypto, Extended)
  - SMF113_2_CSP: Bitmap identifying which counters in a counter set are available. The idea is if a counter set only has counter 1 and 10 defined, space isn't wasted on nonexistent counters 2-9 (counter 1 and counter 10 are contiguous)
- SMF113_2_CSN: The number of counter data sections for this counter set
  - –SMF113_2_CDS: The Counter data section (8 bytes of data)

This change takes effect without any explicit invocation by the user.

32.4 Documentation and Reference

For more information, see the following IBM publications:

- Support Element Operations Guide, Version 2.12.0, IBM zEnterprise System, SC28-6920

  http://www.ibm.com/support/docview.wss?uid=pub1sc28692000
- These publications are available in PDF format in the z/OS V2R1 Elements and Features Library:

  http://www.ibm.com/systems/z/os/zos/library/bkserv/v2r1pdf/
  - *z/OS MVS Installation Exits*, SA23-1381
  - *z/OS MVS Programming: Authorized Assembler Services Reference EDT-IXG,* SA23-1373
  - *z/OS MVS System Commands*, SA38-0666
  - *z/OS MVS System Management Facilities (SMF)*, SA38-0667
  - *z/OS MVS System Messages Volume 6 (GOS-IEA)*, SA38-0673

33

System REXX

This chapter describes the System Restructured Extended Executor (REXX) changes that were introduced in IBM z/OS Version 2 Release 1 (2.1). It includes these sections:

- 33.1, “Running System REXX via the terminal monitor program” on page 426
- 33.2, “Documentation and reference” on page 427

© Copyright IBM Corp. 2014. All rights reserved.

33.1 Running System REXX via the terminal monitor program

There is a need for execs that are running in the System REXX environment to have the ability to obtain unsolicited messages. However, the AXRCMD function returns only the response to an issued command. Problems arise in these cases:

- When the command response is not issued with a command and response token (CART)
- When there are other messages that AXRCMD issues that are not associated with a command but it is still worth providing an automated response for them.

To resolve these problems, support added in z/OS 2.1 enables the use of the Terminal Monitor Program (TMP) as an option for running System REXX, rather than the TSO Environment Server (IKJTSOEV). As a result, the **CONSOLE** host command environment is supported and automation can be done for unsolicited messages.

TMP supports the **SUBMIT** command. The TSO Environment Service does not, so it requires the creation of an internal reader to submit JCL.

33.1.1 Installation

To use the TMP rather than the TSO Environment Service, IKJTSOxx must be updated in these ways:

- **AXRRXWKD** must be listed as an authorized command.
- The update must occur before System REXX initialization.
- Optionally, initialize the RACF **oper** parameter in the RACF database for exec invokers. This is required to use the console to receive unsolicited messages.

Note: If **AXRRXWKD** is removed from IKJTSOxx after System REXX initializes, Sysrexx reverts to using the TSO Environment Service when it detects the change

33.1.2 Invocation and use

The SYSREXX STATUS output has been enhanced to display whether the TMP environment is enabled, as shown in Figure 33-1.

```
AXR0200I SYSREXX STATUS DISPLAY      FRAME LAST   F      E   SYS=SY1
SYSTEM REXX STARTED AT 11.00.30 ON 09/18/2006
PARMLIB MEMBERS:      AXR00 AXR01
CPF:  @ (SYSTEM)      AXRUSER:  MEGA
TIMEINT:   30         TMP:      <ENABLED|NOT ENABLED>
SUBSYSTEM:            AXR
REQUESTS QUEUED:      0  ACCEPTING WORK
```

Figure 33-1 SYSREXX STATUS output

The **AXRINFO** command has been updated to display the TMP status by way of the RexxEnvType parliamentary. **AXRINFO('RexxEnvType')** returns one of the following responses:

- TSO=YES,TMP=NO
- TSO=YES,TMP=YES

- TSO=NO

The message and abend notice that are shown in Example 33-1 are issued when Sysrexx detects that AXRRXWKD is no longer listed as authorized.

Example 33-1 Responses when Sysrexx detects that AXRRXWKD is no longer listed

```
AXR0503I AXRRXWKD IS NO LONGER LISTED AS AUTHORIZED
IN IKJTSOXX. TMP IS NO LONGER ENABLED.

ABEND050 RSN=xxxx044B - AXRRXWKD is running unauthorized.
```

System REXX can be changed dynamically from using the TMP to the Environment Service without restarting SYSREXX. To do this, remove AXRRXWKD from IKJTSOxx to cause the creating of a AXRnn address space.

33.2 Documentation and reference

For more information, see these IBM publications, which are available in PDF format in the z/OS V2R1 Elements and Features Library (search by publication number):

http://www.ibm.com/systems/z/os/zos/library/bkserv/v2r1pdf/

- *z/OS MVS Authorized Assembler Services Guide*, SA23-1371
- *z/OS Security Server RACF Security Administrator's Guide*, SA23-2289
- *z/OS TSO/E REXX Reference*, SA32-0972
- *z/OS TSO/E Programming Services*, SA32-0973, which provides detailed information about the differences between the TMP and the TSO Environment Service

34

IBM Interactive System Productivity Facility

This chapter describes the Interactive System Productivity Facility (ISPF) enhancements, functions, and features that were introduced with IBM z/OS Version 2.1 and covers the following topics:

- 34.1, “Editor Unicode support” on page 430
- 34.2, “Editor compare enhancements” on page 432
- 34.3, “XTIOT support” on page 433
- 34.4, “Directory and member list enhancements” on page 434
- 34.5, “Swapbar visibility enhancement” on page 436
- 34.6, “ISPF support for PDSE version 2” on page 437
- 34.7, “Multiple logical screens when starting ISPF” on page 438
- 34.8, “Edit find and change support for regular expressions” on page 440
- 34.9, “ISPF support for scroll amounts greater than 9999” on page 441
- 34.10, “ISPF edit hilite command enhancement for JCL” on page 442
- 34.11, “ISPF filtering of z/OS UNIX path names” on page 443
- 34.12, “Expanded member count field” on page 445
- 34.13, “Default keyword for memlist service” on page 446
- 34.14, “ISPF editor expandable command field” on page 446
- 34.15, “Enhancement for the udlist command” on page 447
- 34.16, “Free unused space for multi-volume data sets” on page 448
- 34.17, “External data commands support ASCII and UTF-8” on page 448
- 34.18, “z/OS directory list utility” on page 449
- 34.19, “Documentation and reference” on page 452

© Copyright IBM Corp. 2014. All rights reserved.

34.1 Editor Unicode support

ISPF browse can be used to display Unicode data, but before z/OS 2.1, ISPF did not provide any capability to edit Unicode data.

- There were limited tools available on z/OS to view and edit Unicode data.
- More Unicode data is being stored on z/OS.

In z/OS 2.1, the ISPF editor has been enhanced to support the display and editing of data encoded as UTF-8.

Benefit: There is no need anymore to download Unicode data files from z/OS to a workstation to view or change the data.

34.1.1 Invocation and use

There is a Data Encoding option field available on Edit and View entry panels.

- If option 2 is specified ISPF converts the data from UTF-8 (CCSID 1208) to the CCSID of the user's terminal as shown in Figure 34-1.

```
  Menu  RefList  RefMode  Utilities  Workstation  Help
                              Edit Entry Panel

ISPF Library:
   Project . . . _
   Group . . . . _________ . . . _________ . . . _________ . . . _________
   Type  . . . . _________
   Member  . . . _________        (Blank or pattern for member selection list)

Other Partitioned, Sequential or VSAM Data Set, or z/OS UNIX file:
   Name . . . . . _____________________________________________________________ +
   Volume Serial . . _______    (If not cataloged)

Workstation File:
   File Name  . . _____________________________________________________________

                                        Options
Initial Macro  . . . . _________         _  Confirm Cancel/Move/Replace
Profile Name . . . . . _________         _  Mixed Mode
Format Name  . . . . . _________         _  Edit on Workstation
Data Set Password  . .                   _  Preserve VB record length
Record Length  . . . . _____
Line Command Table . . _________        Data Encoding
                                        _  1. ASCII
                                           2. UTF-8



Command ===> _____________________________________________________________
 F1=Help      F3=Exit      F6=Retrieve F10=Actions  F12=Cancel
```

Figure 34-1 Data Encoding option field available on Edit and View

- There are line and primary commands **EU** (Edit Unicode) and **VU** (View Unicode) available with the z/OS UNIX directory list utility (ISPF option 3.17), shown in Figure 34-2 on page 431.

```
 Menu  Utilities  View  Options  Help
                          z/OS UNIX Directory List      Row 149 to 169 of 188

Pathname . : /u/hering
EUID . . . : 888
Command  Filename        Message          Type Permission Audit  Ext  Fmat
-------------------------------------------------------------------------------
_______  ubin.pax                         File rw-r-----  fff--- --s- ----
eu_____  unicode.sample.                  File rw-r--r--  fff--- --s- ----
_______  usr.spool                        Syml rwxrwxrwx  fff---
_______  xx                               File rw-r--r--+ fff--- --s- ----
```

Figure 34-2 primary commands eu and vu

- If a z/OS UNIX file is tagged with CCSID 1208 the editor automatically invokes Unicode support for the **E** or **V** line command.
- If the EU primary or line command is used to edit a z/OS UNIX file ISPF will set the file tag to CCSID 1208 when the file is saved. Here is a primary command example:

  eu unicode.new

- Figure 34-3 shows the file information data after saving the file.

```
                          z/OS UNIX File Information
Command ===>

Pathname  . . : /u/hering/unicode.new

General Data                              Mode Fields
 File Type . . : File                      Permissions . : 700
 File Size . . : 35                        Set User ID . : NO
 Links . . . . : 1                         Set Group ID  : NO
 Inode . . . . : 25F                       Sticky Bit  . : NO
 File Format . : bin
 Last Modified : 2013/06/30 11:23:50      Extended Attributes
 Last Changed  : 2013/06/30 11:23:50       Shared AS . . : YES
 Last Accessed : 2013/06/30 11:23:26       APF Auth  . . : NO
 Created . . . : 2013/06/30 11:23:26       Pgm Control . : NO
 CCSID . . . . : 1208                      Shared Lib  . : NO
 Text Convert  : NO
                                          Audit
Owner                                      Auditor . . . : ---
 File  . . . . : HERING(888)               User  . . . . : fff
 Group . . . . : TTY(2)
                                          Device Data
                                           Device Number : 39
                                           Major Device  :
                                           Minor Device  :
```

Figure 34-3 Showing z/OS file information data

34.1.2 Interactions and dependencies

z/OS Unicode conversion services must be active and the required conversion data must be available for loading by the services.

34.2 Editor compare enhancements

Before z/OS 2.1, there were several problems with comparing data sets, for example:

- It was not possible to enter long compare commands in the command field.
- The **compare** command did not support uncataloged data sets.

With z/OS 2.1, there are solutions for these problems.

- A pop-up panel can be displayed where all parameters for the compare command can be entered.
- A parameter has been added to the compare command for specifying the volume for an uncataloged data set.

Benefit: Users can process long compare commands and compare data in uncataloged data sets.

34.2.1 Invocation and use

The pop-up panel displayed when the compare command is entered without any parameters is changed to include a target data set or file name field and other fields for all the parameters that can be specified with the compare command.

- It can be used when there is insufficient space in the command field to enter the parameters with the compare command.
- The compare command is changed to support a VOL parameter. The VOL parameter is used when the target data set to be compared is uncataloged.
- A volume field is added to the compare settings pop-up panel.

Figure 34-4 shows the changed panel with the format.

```
              Edit Compare Settings and/or Command Parameters    Top of data

SuperC Options:                          Display options:
  Enter "/" to select option               Lines displayed
  _  Case Insensitive Compare              with EXCLUDE  . . . 5   (0 - 12)
  _  Ignore Reformat Differences           Label Prefix  . . . 0   (A - Y)
  _  Data Contains DBCS Characters
                                           Use a label prefix of 0 to enable
                                           special coloring when edit
                                           highlighting is enabled.

Compare Command Parameters:
Name . . . . . _____________________________________________________________ +
Volume . . . . . ______
  Enter "/" to select option             Enter "/" to select option
  _  Exclude                             _  Save
  _  SYSIN                               _  Set SYSIN data set

Enter END to save changes. If NAME is set, Compare will run
Command ===> ____________________________________________________________________
 F1=Help    F3=Exit    F12=Cancel
```

Figure 34-4 Edit compare settings and command parameters

34.3 XTIOT support

Before z/OS 2.1, there was a lack of ISPF support of XTIOT.

- ISPF did not support the processing of data sets dynamically allocated with an XTIOT.
- Also ISPF did not use an XTIOT when allocating a data set.
  - Therefore all TIOT entries had to be below the line and UCBs were captured below the line.

With z/OS 2.1, these problems are solved by the following changes:

- Provide support for ISPF services to process data sets dynamically allocated with an XTIOT
- Provide the ability for ISPF to request an XTIOT when dynamically allocate data sets.

Benefit: This achieves a virtual storage constraint relief because TIOT entries and UCBs do not need to be below the line.

34.3.1 Invocation and use

A keyword `ENABLE_XTIOT` is added to the ISPF configuration table.

- A value of `YES` causes ISPF to process data sets dynamically allocated with an XTIOT and to use an XTIOT when dynamically allocating data sets.
- If YES is specified ISPF checks that parameter `NON_VSAM_XTIOT` in the DEVSUPxx PARMLIB member has a value of YES before enabling the XTIOT support.
- NO is the default value for `ENABLE_XTIOT`.
- The **`ISRDDN`** utility is changed to display a column with a title of **X** to indicate data sets allocated with an XTIOT. The data set name is also highlighted in yellow.
- The `LMINIT` service will set a return code of 8 if the data set to be processed is allocated with an XTIOT and XTIOT support is not fully enabled.

34.3.2 Interactions and dependencies

ISPF XTIOT support is only enabled if XTIOT processing is enabled for the z/OS system by using the `NON_VSAM_XTIOT` parameter in the DEVSUPxx PARMLIB member.

34.3.3 Installation

A keyword, ENABLE_XTIOT, is added to the ISPF configuration table. To enable ISPF support of XTIOT, the ENABLE_XTIOT keyword must be set to yes; in addition, the NON_VSAM_XTIOT keyword in the DEVSUPxx member must be set to YES.

34.4 Directory and member list enhancements

Searching for strings in z/OS UNIX files can be difficult.

- It requires knowledge of UNIX commands like **grep**.
- Commands executed against a member in a list do not have access to the data in the prompt field.

In z/OS 2.1, ISPF provides enhancements for these tasks.

- There is a z/OS UNIX directory list primary command for searching the files in a directory.
- The data in the prompt field is passed as a parameter to a command entered against a member in the enhanced member list.

Benefits:

- It is easy to search z/OS UNIX files for a string.
- Commands can process the data in the prompt field of the enhanced member list.

34.4.1 Invocation and use

Details are described regarding the search function and the use of the prompt field.

Using the Srchfor command

Srchfor is provided as a primary command for the z/OS UNIX directory list utility. Example 34-1 shows the syntax for the command.

Example 34-1 Srchfor command syntax

```
srchfor [ search_string ]
```

The following conditions are relevant to this command:

- The **srchfor** command is available for directory lists that are displayed by using ISPF options 1, 2, 3.17, the **UDLIST** command, and the DIRLIST service.
- The command works in a similarly to the **srchfor** commands for the data set list utility (ISPF option 3.4) and member lists.
- The **srchfor** command invokes the SuperC utility to search for data strings in the regular files in the currently displayed directory.
- Specifying the **search_string** parameter is optional.

  When it is specified, the search is performed using the current settings from the z/OS UNIX directory Srchfor options panel.
- No case conversion is performed on **search_string**.

  For example, if the string specified is `z/OS` and the `Any Case` process option is not used, the search will not identify any regular files that contain the "`Z/OS`" string.
- If a string is not specified with the command, the z/OS UNIX directory Srchfor options pop-up panel is displayed, as shown in Figure 34-5 on page 435.
- The z/OS UNIX directory Srchfor options panel allows several specifications:
  - Multiple (up to 6) search strings can be provided.
  - Process options can be set.
  - Output options can be specified.

- The operands `WORD`, `SUFFIX`, and `PREFIX` can be specified after each search string.
- The **C** (continuation) operand can be used to specify that both the current and previous string must be found on the same line to constitute a match.
- Options are offered to save the first Srchfor string as the current Edit or browse Find string.
- The report that is generated for the search by SuperC is saved in the listing data set.
- Selecting the ASCII option causes the data in the input files to be converted from ASCII to EBCDIC before the data is checked for the search string.

  The ASCII code page is assumed to be ISO 8859-1 (CCSID 819).
- The terminal code page is used as the EBCDIC code page.

  If the terminal code page cannot be determined, code page 1047 is used.

```
                    z/OS UNIX Directory Srchfor Options

You are about to search the regular files in this directory.
Specify search string(s) and options and press ENTER to run the search.
Enter the END or the CANCEL command to cancel the search.
                                                                    More:      +
  ==> ____________________________________________________________________
  ==> ____________________________________________________________________
  ==> ____________________________________________________________________
  ==> ____________________________________________________________________
  ==> ____________________________________________________________________
  ==> ____________________________________________________________________

Listing DSN: 'HERING.SRCHUDL.LIST'

Select Process Options with /          Select Output Options with /
_  Mixed Mode                          _  View output
_  Any case                            _  Save output
_  ASCII
_  Set EDIT FIND string
Command ===> ____________________________________________________________
 F1=Help       F3=Exit       F6=Retrieve   F10=Actions    F12=Cancel
```

Figure 34-5 z/OS UNIX directory Srchfor options

When entering a command such as **`srchfor Executing`** to look specifically for a mixed-case string that says `Executing`, all hits are marked with “String(s) found” message, as shown in Figure 34-6.

```
Pathname . : /u/hering
EUID . . . : 888
Command  Filename        Message          Type Permission Audit  Ext  Fmat
-------------------------------------------------------------------------------
_________ .profile       String(s) found  File rwxr--r--  fff--- --s- nl
_________ .setup                          File rwxr-x---  fff--- --s- ----
_________ .setup.commands                 File rwxr-xr-x  fff--- --s- ----
```

Figure 34-6 Marked hits for a previous search

Using the prompt field

The enhanced member list provides a 9-character line command field to accommodate TSO commands, CLISTs, and REXX EXECs.

- Before z/OS 2.1, a TSO command, CLIST, and REXX procedure entered against a member, the quoted, fully qualified data set name and member was passed as an argument.

- With z/OS 2.1, if the prompt field for the member contains non-blank data that does not start with an * (an asterisk), the prompt field data is added to the argument that is passed to the command.
- Data starting with an * are not passed in the argument (for example, ISPF messages shown for a previous command that was entered against this entry).

34.5 Swapbar visibility enhancement

The *swapbar* is a row on the terminal screen displaying point-and-shoot fields that help the user navigate between ISPF logical screens.

- Users can easily forget about using the swapbar because the swapbar fields can get “lost” among the fields for an ISPF panel.
- In z/OS 2.1, a function allows you to modify the format of the swapbar display.

Benefit: Users are more inclined to use the swapbar for navigation because they can make it stand out on the screen display.

34.5.1 Invocation and use

The **swapbar** command is enhanced to display a pop-up panel where the user can specify options for customizing the swapbar format. Example 34-2 shows this format.

Example 34-2 Syntax of swapbar command to show the pop-up menu

```
swapbar /
```

Figure 34-7 shows the swapbar pop-up menu that is displayed as a result of the command.

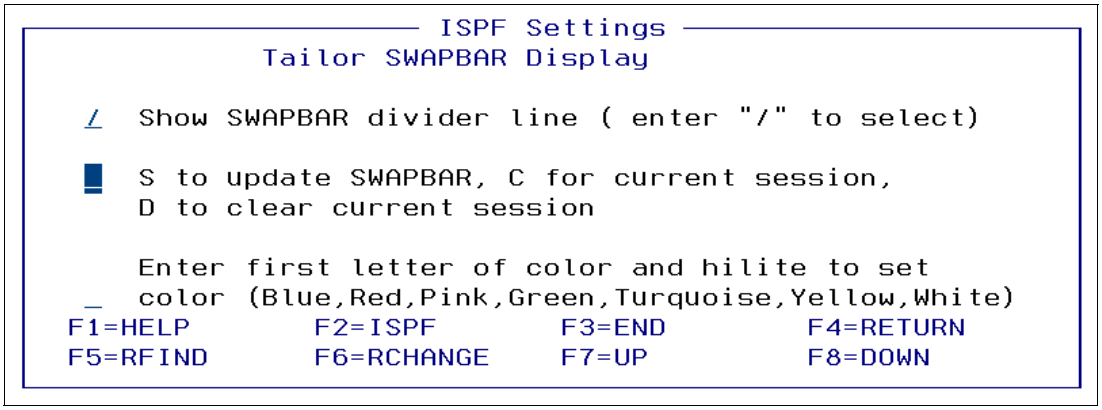


Figure 34-7 The swapbar menu display

If the option for showing the swapbar divider line is selected, a divider line is displayed between the ISPF logical screen and the swapbar, as shown in Figure 34-8 on page 437.

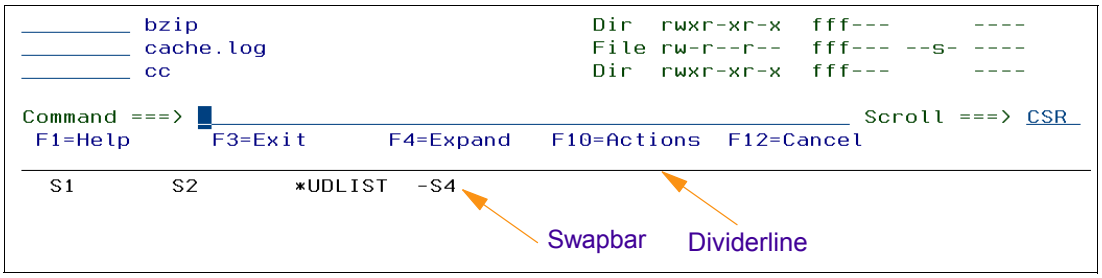


Figure 34-8 Showing the swapbar with the divider line

The following hints for swapbar settings are also useful:

- It is possible to specify color and highlight (hilite) settings for the entire swapbar or for a single swapbar entry.
- Entering an `S` in the update action field causes the entire swapbar to use the specified color and highlight settings.

  The settings for the entire swapbar are saved in the user's system profile.
- Entering a `C` in the update action field causes just the swapbar entry for the current logical screen to use the specified color and highlight settings.

  The settings for a swapbar entry are in effect for only the current ISPF session.
- Entering a `D` in the update action field clears the color and highlight settings for the swapbar entry for the current logical screen.
- The changes to the swapbar settings take effect upon exiting the pop-up panel.

34.6 ISPF support for PDSE version 2

An interactive function is needed that supports the creation of partitioned data set extended (PDSE) version 2 data sets.

- The ISPF data set utility (ISPF option 3.2) is enhanced to support the creation of PDSE version 2 data sets.
- This eliminates the need to submit a job for creation of a PDSE version 2 data set.

34.6.1 Invocation and use

A *data set version* field on the *Allocate data set* panel has been added. As Figure 34-9 on page 438 shows, specifying a value of `2` with data set name type of `LIBRARY` creates a PDSE version 2 data set.

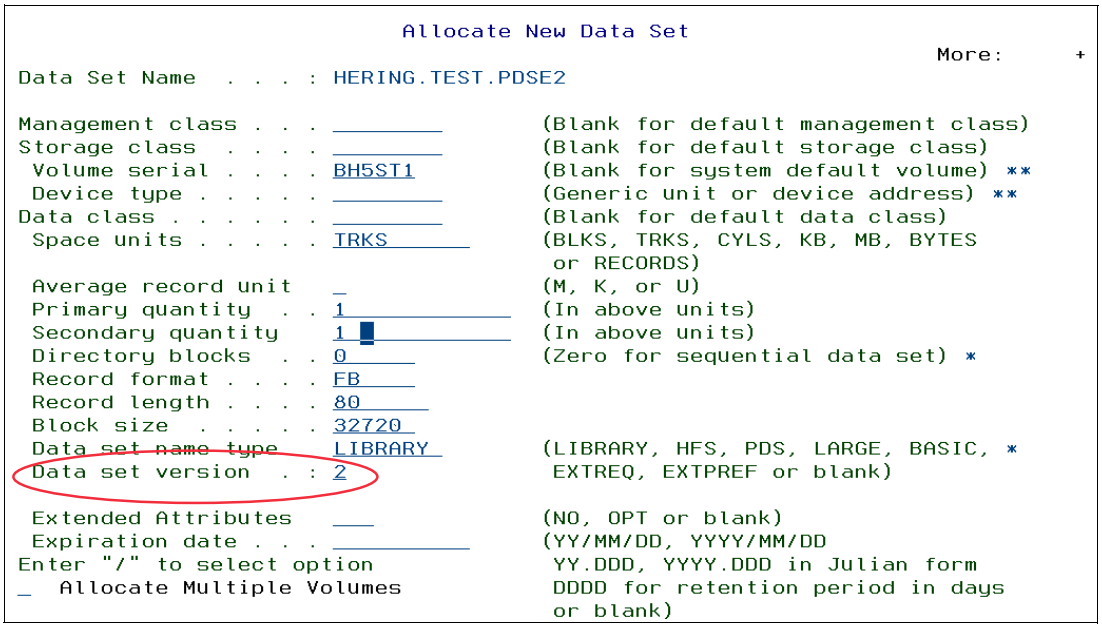


Figure 34-9 Updated allocate data set panel

These are the other improvements in z/OS 2.1:

- The data set information panels are updated to show the version for a PDSE data set.
- The DSINFO service is updated to return the PDSE version number in the ISPF variable ZDSDSNV.
- The LMDLIST service is updated to return the PDSE version number in the ISPF variable ZDLDSNV.

34.7 Multiple logical screens when starting ISPF

ISPF allows users to have up to 32 logical screens, but there is no automation for the creation of multiple logical screens.

- Users must start each logical screen separately for an ISPF session.
- Users generally use the same set of logical screens for each ISPF session.

With z/OS 2.1, it is possible to define a set of logical screens that are automatically started when ISPF is invoked.

This capability provides these benefits:

- It enhances productivity.
- You do not need to go through the process of starting each required logical screen.
- You can define different sets of logical screens when using ISPF for different tasks.

34.7.1 Invocation and use

The following points are relevant for understanding this support:

- An ISPF profile variable is used to define a series of commands to start ISPF logical screens when ISPF is started.

- The variable must contain the `ISPF` identifier, followed by the command delimiter, and then the command stack that is used to start the logical screens. For example:
  - Variable name: `MYSTART`
  - Variable contents: `ISPF;2;START 3.4;START 10;START S;LOG;SWAP 1`
- The name of the variable is specified as an option with the ISPF or ISPSTART command, as this example shows:

  ```
  ispf mystart
  ```
- With this command and the setting of the variable, you get the following screens:
  - ISPF screen 1 is ISPF option 2 (edit).
  - ISPF screen 2 is ISPF option 3.4 (dslist).
  - ISPF screen 3 is ISPF option 10 (sclm).
  - ISPF screen 4 is the System Display and Search Facility (SDSF) log panel (assuming that S is the option for SDSF on the primary menu).
  - The **`swap`** command is used to make the edit panel the initial screen that is displayed when ISPF is invoked.
- If a variable name is not specified with ISPF or ISPSTART, the default profile variable, ZSTART, is used for the initial command stack.

  If ZSTART is not found or does not contain the ISPF identifier, ISPF starts normally.
- The `BASIC` keyword for the ISPF or ISPSTART command can be used to start ISPF normally at the primary panel. Using this keyword bypasses the start of multiple logical screens. Syntax example:

  ```
  ispf basic
  ```

=XALL command

A **`=XALL`** command helps terminate all logical screens with one command, with the following conditions:

- The **`=X`** command is propagated to every logical session to terminate each application that supports **`=X`**.
- If **`=X`** is not supported, the termination process halts on that logical screen.
- After that logical window closes, **`=XALL`** processing continues for each remaining logical window.

ISPF initialization command table entry keyword

There is a configuration table keyword: `ISPF_INIT_COMMAND_TABLE_ENTRY`. The following stipulations apply:

- Define an ISPF command that can be used to set or override the value of the variable that contains the initial command stack.
- The value can be set on the *Modify ISPF sitewide defaults* panel of the ISPCCONF command.
- This `ISPF_INIT_COMMAND_TABLE_ENTRY` must specify a command in an ISPF command table, as this example shows:

  ```
  ISPFINI   0  SELECT CMD(ISPINIT)
  ```
- ISPF invokes the command only when the value of the keyword is changed, as in this example:

  ```
  VERSION_LEVEL_OF_SITEWIDE_DEFAULTS
  ```

- The command invokes a REXX, CLIST, or program to handle the process of setting or overriding the initial command stack, as shown in Figure 34-10.

```
/* REXX to set variable ZSTART ----------------- */
Trace O
/************************************************/
/* Update variable ZSTART with the desired value. */
/************************************************/
zstart = "ISPF;3.4;START 13.12;START 2;SWAP 1"
Address ISPEXEC "VPUT (ZSTART) PROFILE"
/************************************************/
/* Start ISPF with the command stack set in       */
/* ZSTART.                                        */
/************************************************/
zstart = "3.4;START 13.12;START 2;SWAP 1"
Address ISPEXEC "DISPLAY PANEL(ISPBLANK) COMMAND(ZSTART)"
```

Figure 34-10 REXX to set variable ZSTART

34.8 Edit find and change support for regular expressions

Regular expressions have become a well understood way to define search patterns. Unfortunately, before z/OS 2.1, the ISPF editor find, change, and seek commands did not support regular expressions. However, in z/OS 2.1, those commands allow the search string to be defined by using a regular expression.

Note: This provides far greater flexibility with specifying a search pattern to the ISPF editor.

34.8.1 Invocation and use

A regular expression is specified as a quoted string, preceded or followed by the letter `R`. For example, using the following command finds the words lane and line:

```
find r'l[ai]ne' word
```

Keep these factors in mind, too:

- The regcomp() and regexec() C runtime library functions are used to compile the regular expression and to check each logical record for strings that match the regular expression.
- The regular expression formats supported by ISPF are documented in the *XL C/C++ Runtime Library Reference* under the regcomp() function.
- When a TN3270 emulator that supports CCSIDs 0037, 1047, and 0273 is used, ISPF calls the C/C++ setlocale() function before compiling the regular expression so that special symbols within the regular expression are correctly interpreted.

Figure 34-11 on page 441 lists a sample REXX edit macro that uses `seek` with a regular expression.

```
/* REXX SEEKREGX ------------------------------------------------- */
/*      for demonstration of seek with regular expression           */
/* ------------------------------------------------------------------ */
Address ISREDIT
"MACRO"
"SEEK r'l[ai]ne' all word"
Say "Seek RC=" rc
"(seektot,seeklns) = SEEK_COUNTS"
Say "Total hits ..............:" seektot
Say "Number of lines with hits:" seeklns
Exit
```

Figure 34-11 REXX for demonstration of using regular expressions

This REXX is executed next as an ISPF edit macro from the edit command line for a data set or member with the content shown in Figure 34-12.

```
this line contains line
blah blah blah
blah blah blah
blah blah blah
blah line blah line
blah blah blah
blah blah blah
here is the value lane here
blah blah blah
blah lane blah
```

Figure 34-12 Sample data set for testing REXX seekregx

Figure 34-13 shows the results when running seekregx against this data set.

```
 Seek RC= 0
 Total hits ..............: 00000006
 Number of lines with hits: 00000004
```

Figure 34-13 Results of executing the seekregx macro

Note: APAR OA42815 has been opened (and a fix is available) to enhance the ISPF edit FIND, CHANGE, and SEEK regular expression support by expanding the number of TN3270 code pages used to set the locale for compilation of a regular expression from the initial set of 037, 273 or 1047.

34.9 ISPF support for scroll amounts greater than 9999

Before z/OS 2.1, the limitation of scroll amounts to values of 9999 or less made navigation in ISPF functions cumbersome.

z/OS 2.1 has support for scroll amounts greater than 9999.

Benefit: This increases productivity as a result of more efficient navigation within data and lists when using ISPF.

34.9.1 Invocation and use

ISPF is enhanced to support the input of scroll amounts up to 9,999,999.

Note: The scroll fields on panels have not been changed to support the display and input of scroll amounts greater than 9999.

- The values for ISPF configuration keywords, `SCROLL_MIN` and `SCROLL_MAX`, can now be 7-digit numbers.
- For table display processing, the function pool variable, *`ZTDAMTL`*, is 8 bytes in length to accommodate scroll values greater than 9999.
- Current function pool variable, *`ZTDAMT`*, also contains the scroll amount when it is 4 characters or fewer. It is set to zero when the scroll amount is more that 4 characters.

34.9.2 Migration and coexistence

Note: The `SCROLL_MAX` value in the configuration table must be explicitly reset.

Follow these steps to reset the value:

1. From the ISPF main menu, go to option 7.3 and enter `L ZVERMOD`. Make a note of the value and return to the main menu.
2. Start ISPCCONF, either by using **`TSO ISPCCONF`** or from option 6. You will need a key file.
3. Select option 1, “Create/Modify Settings and Regenerate Keyword File.”
4. Select option 4, and change the “Sitewide Defaults Version Level” number to at least one number higher than the value in step 1.
5. Page down to change the value of “Maximum Scroll Value” and set “Reset Scroll Values” set to On with a / in the input field.
6. Press pf3 twice to get an Edit window. Use the **`find`** command to check for uncommented `RESET` keywords that might cause unwanted changes.
7. Press pf3, and then use option 4 to compile the configuration load module. Make it available to all ISPF users and restart your ISPF session.
8. Go to option 7.3 again and verify that the value of `ZVERMOD` has changed.

34.10 ISPF edit hilite command enhancement for JCL

The ISPF editor’s **`hilite`** command provides language-sensitive coloring for JCL. This assists users in coding valid JCL statements. However, before z/OS 2.1, it does not show the invalid use of lowercase characters.

z/OS 2.1 solves that problem. The **`hilite`** command now identifies lowercase characters that are used incorrectly in JCL.

Benefit: Productivity increases with highlighting for invalid statements that contain lowercase characters.

34.10.1 Invocation and use

The **edit hilite** command highlights lowercase characters that are used invalidly in JCL. Figure 34-14 shows a sample edit session using highlighting.

```
EDIT       HERING.ZFS.JOB.CNTL(IOEFSUTL) - 01.01            Columns 00001 00072
****** ***************************** Top of Data ******************************
000001 //ZFSUTL   JOB ,'zFS Utility',NOTIFY=&SYSUID.,REGION=0M
000002 /*JOBPARM SYSAFF=SC74
000003 //*MAIN SYSTEM=SC75
000004 //* ---------------------------------------------------------------------
000005 //SALVAGE  EXEC   PGM=IOEFSUTL,REGION=0M,
000006 //          PARM='salvage -aggregate HERING.TEST.RW.ZFS -verifyonly')
000007 //SYSPRINT DD   sysout=*
000008 //STDOUT   DD   SYSOUT=*
000009 //stderr   DD   SYSOUT=*
000010 //SYSUDUMP DD   SYSOUT=*
000011 //CEEDUMP  DD   SYSOUT=*
000012 //* ---------------------------------------------------------------------
****** ***************************** Bottom of Data ***************************
```

Figure 34-14 JCL with invalidly used lowercase characters highlighted

34.11 ISPF filtering of z/OS UNIX path names

In previous releases of z/OS, to display a z/OS UNIX directory list, the exact path name for the directory needs to be specified, so you must know the exact path name.

In z/OS 2.1, you can use pattern matching characters when entering a directory path name. A list of all files and directories with names that match the pattern is displayed.

Benefit: This enhances productivity as it is easier to locate directories and files.

34.11.1 Invocation and use

The global or pattern-matching characters shown in Table 34-1 can now be specified in a path name that is entered to get a z/OS UNIX directory list display.

Table 34-1 Global or pattern-matching characters

Character	Description
?	This matches any single character.
* (asterisk)	This matches multiple characters.
[	This opens a set of single characters.
]	This closes the set of single characters.

Notes:

- Each character in the set, enclosed in [square brackets], can match a single character at the position specified.
- The glob() C runtime library function is used to build a list of files and directories with path names that match the specified pattern.

Version 2.1 offers these additional enhancements:

- The dirlist service is changed to support pattern-matching characters in the path name that is passed by the caller.
- The **udlist** command allows pattern-matching characters in the specified path name.
- The ISPF data set list utility (ISPF option 3.4) supports the display of a z/OS UNIX directory list. A directory path name or path name pattern can now be entered in the dsname level field.

Figure 34-15 shows the list from the sub-directories in `/usr` that have a first character of i or `l`, all files with an 8-character name starting with c, and an extension beginning with c, d, or h.

```
                          z/OS UNIX Directory List          Row 1 to 18 of 18

List . . . : /usr/[il]*/[c]???????.[cdh]*
EUID . . . : 888
Command  Pathname        Message          Type Permission Audit  Ext  Fmat
-------------------------------------------------------------------------------
_________ /usr/include/ce                 File rw-r--r--  fff--- --s- ----
_________ /usr/include/ce                 File rw-r--r--  fff--- --s- ----
_________ /usr/include/ce                 File rw-r--r--  fff--- --s- ----
_________ /usr/include/cs                 Syml rwxrwxrwx  fff--- --s- ----
_________ /usr/include/cs                 Syml rwxrwxrwx  fff--- --s- ----
_________ /usr/include/cs                 Syml rwxrwxrwx  fff--- --s- ----
_________ /usr/include/cs                 Syml rwxrwxrwx  fff--- --s- ----
_________ /usr/lib/cdserp                 Syml rwxrwxrwx  fff--- --s- ----
_________ /usr/lib/cdsibm                 Syml rwxrwxrwx  fff--- --s- ----
_________ /usr/lib/cdskwu                 Syml rwxrwxrwx  fff--- --s- ----
_________ /usr/lib/cdsnsp                 Syml rwxrwxrwx  fff--- --s- ----
```

Figure 34-15 Listing UNIX entries for /usr/[il]/[c]???????.[cdh]**

filter command

A directory list **filter** primary command can be used to append to or replace the current path name filter. Figure 34-16 shows the syntax of the command.

```
FILTER string APPEND|REPLACE
```

Figure 34-16 Syntax of the filter command

- Using APPEND, which is the default, adds *string* to the end of the current filter.

  If the current filter is **/u/harry/test***, the **filter p** command changes the filter to **/u/harry/test*p** and the list is rebuilt.
- Using REPLACE replaces the current filter with `string`.
- The REFRESH command restores the list, based on the initial filter.

Figure 34-17 on page 445 shows a z/OS UNIX directory list before using a filter command.

```
                          z/OS UNIX Directory List          Row 1 to 11 of 11

List . . . : /usr/lib*/c[es]*
EUID . . . : 888
Command  Pathname                    Message           Type Permission Audit
-------------------------------------------------------------------------------
_________ /usr/lib/ceasapii.dll                        File rwxr-xr-x  fff---
_________ /usr/lib/ceasapis.dll                        File rwxr-xr-x  fff---
_________ /usr/lib/ceasapit.dll                        File rwxr-xr-x  fff---
_________ /usr/lib/ceasapit.x                          File rw-r--r--  fff---
_________ /usr/lib/ceasapiw.dll                        File rwxr-xr-x  fff---
_________ /usr/lib/csnpcapi.so                         Syml rwxrwxrwx  fff---
_________ /usr/lib/csnpca3x.so                         Syml rwxrwxrwx  fff---
_________ /usr/lib/csnpca64.so                         Syml rwxrwxrwx  fff---
_________ /usr/lib/cssmmanp.dll                        Syml rwxrwxrwx  fff---
_________ /usr/lib/cssmusep.dll                        Syml rwxrwxrwx  fff---
_________ /usr/lib/cssm32.dll                          Syml rwxrwxrwx  fff---
****************************** Bottom of data ********************************
```

Figure 34-17 Listing UNIX entries for /usr/lib/c[es]**

Figure 34-18 shows the list after executing the **`filter.s*`** command. This command reduces the filtered list to only those entries with an extension that starts with the letter `s`.

```
                          z/OS UNIX Directory List            Row 1 to 3 of 3

List . . . : /usr/lib*/c[es]*.s*
EUID . . . : 888
Command  Pathname                    Message           Type Permission Audit
-------------------------------------------------------------------------------
_________ /usr/lib/csnpcapi.so                         Syml rwxrwxrwx  fff---
_________ /usr/lib/csnpca3x.so                         Syml rwxrwxrwx  fff---
_________ /usr/lib/csnpca64.so                         Syml rwxrwxrwx  fff---
****************************** Bottom of data ********************************
```

Figure 34-18 Listing UNIX entries for /usr/lib/c[es]*.s**

34.12 Expanded member count field

In releases before z/OS 2.1, the member count field for member lists is limited to a value of 99999. Because the member count field is truncated, it is invalid for partitioned data sets with more than 99999 members.

In z/OS 2.1, the member count field is expanded to seven characters to allow the display of values up to 9,999,999.

Benefit: A correct value for the number of members in partitioned data sets is provided.

34.12.1 Invocation and use

The member count fields for member lists are expanded from five to seven digits to accommodate partitioned data sets with up to 9,999,999 members, as shown in Figure 34-19 on page 446.

```
EDIT          HERING.ZFS.BPXBATCH.JOB.CNTL.VERY.LON Row 0000001 of 0000003
          Name      Prompt        Size   Created          Changed           ID
_________ #ENVVARS                 6   2009/04/16  2009/04/16 20:28:51  HERING
_________ RXBATCH                 39   2009/04/16  2009/04/23 17:06:33  HERING
_________ SAMPLE                  46   2002/04/18  2009/04/16 20:49:27  HERING
          **End**
```

Figure 34-19 Showing expanded member count field

Expanded member count fields have reduced the display size of the data set name field, which is now scrollable when the name is too long for the field. The cursor can be placed on the data set name field, and the pfkey can be used to scroll the field to the right. The result of doing so is shown in Figure 34-20.

```
EDIT          ZFS.BPXBATCH.JOB.CNTL.VERY.LONG.NAME  Row 0000001 of 0000003
          Name      Prompt        Size   Created          Changed           ID
_________ #ENVVARS                 6   2009/04/16  2009/04/16 20:28:51  HERING
_________ RXBATCH                 39   2009/04/16  2009/04/23 17:06:33  HERING
_________ SAMPLE                  46   2002/04/18  2009/04/16 20:49:27  HERING
          **End**
```

Figure 34-20 Showing the right part of long data set names

34.13 Default keyword for memlist service

Before z/OS 2.1, the member list displayed when using the memlist service did not provide a default action when the **S** line command was used.

In z/OS 2.1, a keyword is added to the memlist service. It allows the caller to define a line command to be executed when the **S** line command is entered.

Benefit: This enables member lists to be displayed by using the memlist service and to work in a way that is consistent with the enhanced member lists available in ISPF.

34.13.1 Invocation and use

In the example shown in Figure 34-21, line command **E** is used when the user enters S for a member in the member list display for HERING.ZFS.BPXBATCH.JOB.CNTL.

```
/* REXX */
Address ISPEXEC
"lminit dataid(c) data set('HERING.ZFS.BPXBATCH.JOB.CNTL')"
"memlist dataid("c") default(e)"
```

Figure 34-21 Small REXX using the default keyword of the memlist service

34.14 ISPF editor expandable command field

Occasionally, editor commands such as the **change** command are too long to be entered in the command field. In z/OS 2.1, you can expand the editor panel command field.

Benefit: This enhances productivity because users are able to enter long and complex editor primary commands.

34.14.1 Invocation and use

The input field on the command line of the edit display panels can now be expanded.

Use the **zexpand** to display a pop-up window with the command input field expanded to a length of 255 characters.

Notes:

- PF4 is set to invoke the **zexpand** command in the ISPF-supplied edit key lists.
- The pop-up window is only for editing primary commands. Use **cmde** for other commands.

Normal processing for a long edit command involves the following steps:

1. Start entering the long edit command. If it will not fit into the standard field size, proceed with the next steps.
2. Press the pfkey that has the **zexpand** command assigned to display the pop-up window with the expanded command field.
3. Enter the command that you want to use.
4. Press PF3 to return to the edit panel.
5. Press the enter key to run the command.

34.15 Enhancement for the udlist command

In releases before z/OS 2.1, there was a problem with **UDLIST** path names provided. z/OS UNIX path names are usually lowercase. Most command fields are defined with `CAPS(ON)`, which caused any path name specified with the **UDLIST** command to be converted to uppercase. As a result, the directory might not be found.

In z/OS 2.1, if the directory for the path name specified with the **udlist** command is not found, ISPF converts the path name to lowercase and runs the **UDLIST** command again.

Benefit: This improves productivity because it increases the likelihood of the **udlist** command finding the requested directory.

34.15.1 Invocation and use

No user changes are needed to implement this improved search. However, a directory named **MixedCaseDir** still cannot be displayed.

34.16 Free unused space for multi-volume data sets

Data Facility Storage Management Subsystem (DFSMS) has provided support for freeing space in multi-volume data sets for a long time. However, before z/OS 2.1, it was an invalid command for multi-volume data sets.

With z/OS 2.1, DFSMS enhances the freeing of space for SMS-managed multi-volume data sets. The data set list **free** line command can be used to free unused space from data sets.

Note: This increases productivity, because it simplifies the task of freeing unused space in multi-volume data sets.

34.16.1 Invocation and use

The ISPF data set list utility (ISPF option 3.4) is changed to no longer reject the **F** (free) line command when it is issued against a multi-volume data set.

The existing free space logic in ISPF works for multi-volume data sets and picks up the enhancements for DFSMS.

34.17 External data commands support ASCII and UTF-8

The ISPF editor provides external data commands to obtain data from or store data in other files. External data commands do not account for the other file using a different data encoding.

In z/OS 2.1, the external data commands allow specifying the data encoding for the other file.

Note: This improves productivity, because you can easily exchange data between files with different data encoding.

34.17.1 Invocation and use

EBCDIC, ASCII, and UTF8 keywords are available now for the **create**, **replace**, **copy**, **move**, and **cut** primary editing commands.

- For **create** and **replace** commands, the keyword specifies the encoding that the editor will convert the data to before saving in the external file.
- For **copy** and **move** commands, the keyword identifies the encoding for the source file. The editor converts from this encoding to either the character set specified for the file being edited or to the terminal character set.
- For the **cut** command, the keyword specifies the encoding that the editor converts the data to before saving in the clipboard.
- The **paste** command detects the character set of the data in the clipboard converts the data copied into the edited file. A ASIS keyword can be used to prevent the **paste** command from performing the conversion.
- The **submit** command always converts a file designated as ASCII or UTF8 to the terminal CCSID (which is EBCDIC) before writing the data to the internal reader.

34.18 z/OS directory list utility

Before z/OS 2.1, there were several disadvantages in working with z/OS directory list utility:

- A line command must have been entered against each file when a user wanted to perform the same processing against a group of files.
- The limited width of the file name column made it difficult to identify files with long names.
- No support was provided to save and retrieve commonly used z/OS UNIX commands across ISPF sessions.
- The utility could browse only the output from a z/OS UNIX command. It was impossible to use the added features of the ISPF editor with command output.

With z/OS 2.1, there are solutions for these problems:

- You can enter z/OS UNIX directory list line commands in blocks.
- You can extend the width of the file name column.
- A command shell for the input, saving, and retrieval of z/OS UNIX commands is provided.
- You can specify whether to use Browse or Edit to display output from a z/OS UNIX command.

Benefit: Each of these enhancements improves the usability of the z/OS UNIX directory list utility.

34.18.1 Invocation and use

This section explains use of the functions.

z/OS UNIX directory list utility block line commands enhancement

This allows the same line command to be executed against multiple files at once. The following information related to this command is helpful to know:

- The implementation is similar to the block line command support in the ISPF data set list utility (ISPF option 3.4).
- The start and end of the block is indicated by the user typing two forward slash (//) characters in the line command fields for the start and end of the block.
- The line command must immediately follow the two forward slashes at either the start or end of the block.
- All line commands, including z/OS UNIX commands, TSO commands, CLISTs, and REXX execs can be invoked as block commands.

A simple example is shown in Figure 34-22 on page 450, with the results shown in Figure 34-23 on page 450.

```
                          z/OS UNIX Directory List             List 60 of 201

Pathname . : /u/hering
EUID . . . : 888
Command  Filename                    Message          Type Perm Audit  Ext  Fmat
-------------------------------------------------------------------------------
________ no.access                                    Dir  700  fff---       ----
//d_____ no.longer.needed.01                          File 600  fff--- --s- ----
________ no.longer.needed.02                          File 600  fff--- --s- ----
________ no.longer.needed.03                          File 600  fff--- --s- ----
________ no.longer.needed.04                          File 600  fff--- --s- ----
________ no.longer.needed.05                          File 600  fff--- --s- ----
________ no.longer.needed.06                          File 600  fff--- --s- ----
________ no.longer.needed.07                          File 600  fff--- --s- ----
________ no.longer.needed.08                          File 600  fff--- --s- ----
//_____ no.longer.needed.09                           File 600  fff--- --s- ----
________ no.longer.needed.10                          File 600  fff--- --s- ----
________ notes.ini                                    File 777  fff--- --s- ----
```

Figure 34-22 Using block line commands, part 1

Figure 34-23 shows the results after pressing the enter key.

```
                          z/OS UNIX Directory List             List 60 of 201

Pathname . : /u/hering
EUID . . . : 888
Command  Filename                    Message          Type Perm Audit  Ext  Fmat
-------------------------------------------------------------------------------
________ no.access                                    Dir  700  fff---       ----
_________ no.longer.needed.01        Deleted
________ no.longer.needed.02         Deleted
________ no.longer.needed.03         Deleted
________ no.longer.needed.04         Deleted
________ no.longer.needed.05         Deleted
________ no.longer.needed.06         Deleted
________ no.longer.needed.07         Deleted
________ no.longer.needed.08         Deleted
________ no.longer.needed.09         Deleted
________ no.longer.needed.10         Deleted
________ notes.ini                                    File 777  fff--- --s- ----
```

Figure 34-23 Using block line commands, part 2

z/OS UNIX directory list utility file name column enhancement

This enables you to extend the size of the field used to display the file name beyond the current limit of 30 characters. Keep these points in mind:

- The maximum size now supported for the size of the file name field is 110 characters.
- This value is entered in the “Width of the file name column” field on the z/OS UNIX directory list options panel.
- If the value specified is larger than the current screen width minus 50, ISPF uses the screen width minus 50 for the size of the file name field.

z/OS UNIX directory list utility command shell enhancement

This gives you the ability to enter, save, and retrieve z/OS UNIX commands.

- It works in a way similar to the method used to enter, save and retrieve TSO commands in the ISPF command shell (ISPF option 6). Keep these points in mind:
- It is Invoked by entering a **/** (forward slash) in the primary command field of the z/OS UNIX directory list panel.
- It displays the z/OS UNIX directory list command entry panel.

- This provides a 255 character length command field for entering long z/OS UNIX and TSO commands.
- It has a list of point-and-shoot fields showing the last 10 z/OS UNIX commands entered from the z/OS UNIX directory list utility.
- The user can control retrieval from and updates to the list.

Figure 34-24 shows the z/OS UNIX directory list command entry panel.

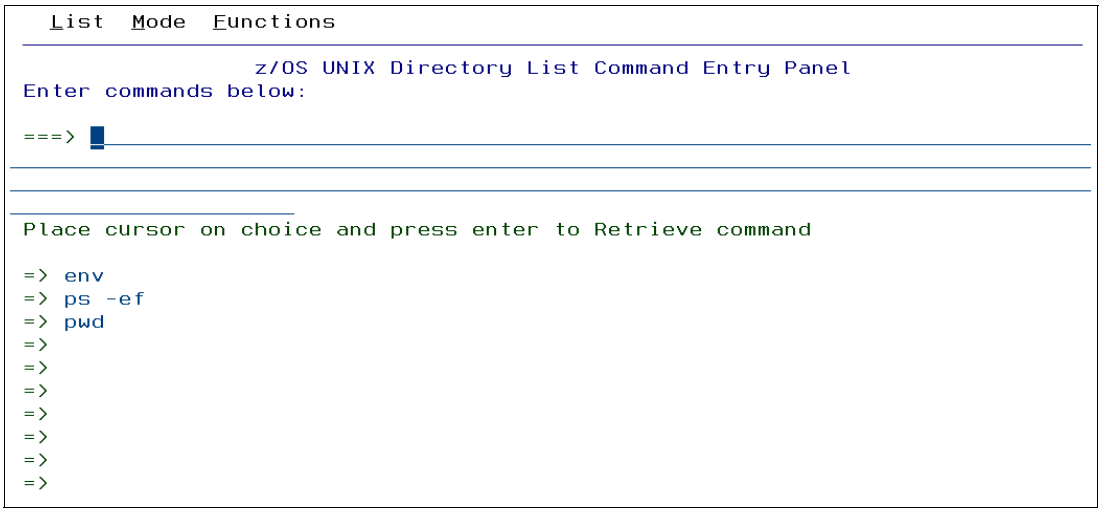


Figure 34-24 z/OS UNIX directory list command entry panel

The drop-down menus of the command entry panel work just like those of the ISPF command shell in the following ways:

- The List drop-down menu controls whether ISPF adds entries to the list of commands.
- The Mode drop-down menu controls whether a command selected from the list is retrieved to the command field, run immediately, or deleted from the list.
- The Functions drop-down menu provides an option to remove deleted or null entries from the command list.

z/OS UNIX directory list utility enhancement for command output

The Output Mode field on the z/OS UNIX Directory List Options panel, shown in Figure 34-25 on page 452, allows you to specify whether the Browse or the View function of ISPF is used to display output from a z/OS UNIX command.

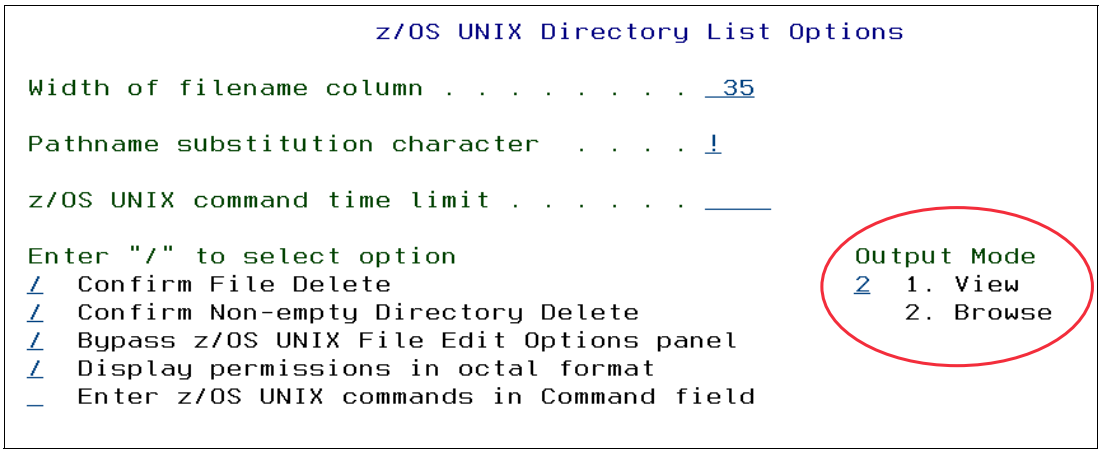


Figure 34-25 output mode field on the z/OS UNIX directory list options

34.19 Documentation and reference

For more information, see these IBM publications, which are available in PDF format in the z/OS V2R1 Elements and Features Library (search by publication number):

http://www.ibm.com/systems/z/os/zos/library/bkserv/v2r1pdf/

- *z/OS V2R1 ISPF Edit and Edit Macros*, SC19-3621
- *z/OS V2R1.0 ISPF Dialog Developer's Guide and Reference*, SC19-3619
- *z/OS V2R1.0 ISPF Planning and Customizing*, GC19-3623
- *z/OS V2R1.0 ISPF Services Guide*, SC19-3626
- *z/OS V2R1.0 ISPF User's Guide Volume I*, SC19-3627
- *z/OS V2R1 ISPF User's Guide Volume II*, SC19-3628
- *z/OS V2R1.0 XL C/C++ Runtime Library Reference*, SC14-7314
- *z/OS XL C/C++ Runtime Library Reference*

35

z/OS UNIX System Services

This chapter describes the z/OS UNIX System Services (z/OS UNIX or USS) enhancements, functions, and features that were introduced with IBM z/OS Version 2 Release 1 (2.1):

- 35.1, “Pipes enhancements” on page 454
- 35.2, “The zlsof command” on page 459
- 35.3, “I/O count correction in SMF records” on page 463
- 35.4, “Automount enhancements” on page 465
- 35.5, “TFS enhancements” on page 469
- 35.6, “Unicode services use” on page 472
- 35.7, “REXX enhancements” on page 476
- 35.8, “Avoiding access loss for previous default users” on page 483
- 35.9, “Documentation and reference” on page 487

© Copyright IBM Corp. 2014. All rights reserved.

35.1 Pipes enhancements

For context, review the following basic points first:

- A z/OS UNIX pipe is an I/O channel that a process can use to communicate with another process, with another thread, in this same process or another process, or, in some cases, with itself.
- Data can be written into one end of the pipe and read from the other.
- A pipe can be named or unnamed:
  - A named pipe is created by using the **mkfifo** shell command, and it is opened by the application.
  - An unnamed pipe us created by using the `pipe()` callable service.

Before z/OS 2.1, z/OS pipe processing had several problems:

- The system limit of 8,730 pipes was artificially low.
- Unpredictable application failures could be seen when the pipe limit was exceeded.
- Failures could happen that were hard to analyze and debug by finding the root cause.
- It was hard to analyze the pipe use, in general.

z/OS 2.1 includes the following changes:

- The pipe limit has been increased to 15 K.
- The LIMMSG system limit monitoring has been extended to include MAXPIPES.
- A user-based MAXPIPEUSER limit has been provided to limit the use.
- The **D OMVS,PIPES** command is added to summarize the pipe use.

Several benefits result from these changes:

- The pipe capacity has been doubled.
- The pipe use can be monitored and easily reviewed.
- The MAXPIPEUSER provides better control of the pipe use.

Notes:

- An opened pipe requires approximately 132 K of system resources to manage.
- There is a practical limit to the number of pipes that can be opened in the system at any point in time, because the pipe storage is allocated from a dedicated 2 G data space.

35.1.1 Invocation and use

This section explains pipe limit monitoring, pipe use limits by user, and using the **D OMVS,PIPES** command.

Pipe limit monitoring

You can easily enable MAXPIPES limit monitoring support by using either the `LIMMSG(SYSTEM)` or `LIMMSG(ALL)` PARMLIB statement.

Notification of high pipe use begins at 85%, as the following example shows, use with repeated notification at every 5% increase:

```
BPXI039I SYSTEM LIMIT MAXPIPES HAS REACHED 85% OF ITS   CURRENT CAPACITY OF 15360
```

When the shortage is relieved, you get notified again, as this example shows:

```
BPXI042I RESOURCE SHORTAGE FOR MAXPIPES HAS BEEN RELIEVED
```

When the MAXPIPES use value exceeds 100%, the BPXF266E message is issued:

```
BPXF266E THE NUMBER OF ACTIVE PIPES HAS REACHED THE SYSTEM LIMIT.
```

Note: This message is deleted by the DOM macro when use falls below 85%.

The current system pipe use can be viewed by using **D OMVS,LIMITS** as shown in Figure 35-1.

```
D OMVS,LIMITS
BPXO051I 15.52.19 DISPLAY OMVS 062
OMVS     0011 ACTIVE             OMVS=(1C)
SYSTEM WIDE LIMITS:         LIMMSG=NONE
                   CURRENT  HIGHWATER     SYSTEM
                     USAGE      USAGE      LIMIT
MAXPROCSYS              46         57        900
MAXUIDS                  8          9        200
MAXPTYS                  1          2        800
...
MAXPIPES                21         23      15360
```

Figure 35-1 D OMVS,LIMITS command results

Limiting pipe use by user

The MAXPIPEUSER setting limits the pipe use per user. The intent is to prevent a user or job from acquiring a disproportionate number of pipe resources.

The limitation is configured by setting the MAXPIPEUSER PARMLIB statement in BPXPRMxx, with these stipulations to consider:

- The range is between 256 and 8730.
- The default value is 8730.
- A MAXPIPEUSER limit of 8730 is always used for the UID=0 user.

The value can be dynamically changed by using either **SETOMVS** or **SET OMVS**, as shown in Figure 35-2.

```
SETOMVS MAXPIPEUSER=256
BPXO015I THE SETOMVS COMMAND WAS SUCCESSFUL.
```

Figure 35-2 SETOMVS MAXPIPEUSER=256 command example

The current MAXPIPEUSER value can be viewed by using the **D OMVS,O** command, as shown in Figure 35-3 on page 456.

```
BPXO043I 13.31.35 DISPLAY OMVS 970
OMVS     0011 ACTIVE             OMVS=(1C)
CURRENT UNIX CONFIGURATION SETTINGS:
MAXPROCSYS      =        900    MAXPROCUSER     =      32767
MAXFILEPROC     =      65535    MAXFILESIZE     = NOLIMIT
MAXCPUTIME      =       1000    MAXUIDS         =        200
MAXPTYS         =        800    MAXIOBUFUSER    =       2048
MAXMMAPAREA     =      40960    MAXASSIZE       = 2147483647
MAXTHREADS      =     100000    MAXTHREADTASKS  =      32767
MAXCORESIZE     =    4194304    MAXSHAREPAGES   =     131072
IPCMSGQBYTES    = 2147483647    IPCMSGQMNUM     =      10000
IPCMSGNIDS      =        500    IPCSEMNIDS      =        500
IPCSEMNOPS      =         25    IPCSEMNSEMS     =       1000
IPCSHMMPAGES    =      25600    IPCSHMNIDS      =        500
IPCSHMNSEGS     =        500    IPCSHMSPAGES    =     262144
SUPERUSER       = BPXROOT       FORKCOPY        = COW
STEPLIBLIST     =
USERIDALIASTABLE=
PRIORITYPG VALUES: NONE
PRIORITYGOAL VALUES: NONE
MAXQUEUEDSIGS   =       1000    SHRLIBRGNSIZE   =   67108864
SHRLIBMAXPAGES  =       4096    VERSION         = Z21RD1
SYSCALL COUNTS  = NO            TTYGROUP        = TTY
SYSPLEX         = YES           BRLM SERVER     = N/A
LIMMSG          = NONE          AUTOCVT         = OFF
RESOLVER PROC   = DEFAULT       LOSTMSG         = ON
AUTHPGMLIST     = NONE
SWA             = ABOVE         NONEMPTYMOUNTPT = NOWARN
SERV_LINKLIB    =
SERV_LPALIB     =
ALTROOT         =
MAXUSERMOUNTSYS =          0    MAXUSERMOUNTUSER=          0
MAXPIPEUSER     =       8730    PWT             = ENV
```

Figure 35-3 D OMVS,O display for MAXPIPEUSER

Figure 35-4 shows an example of displaying the MAXPIPEUSER value by using z/OS UNIX commands from the shell.

```
$> cn "d omvs,o" | grep MAXPIPEUSER
 MAXPIPEUSER     =        256    PWT             = ENV
$>
```

Figure 35-4 D OMVS,O command to display the max pipe user value

Note: The **cn** command is available as an additional download at the following IBM International Technical Support Organization (ITSO site.) See Appendix A, “Material available to download” on page 535.

When the MAXPIPEUSER limit is reached, the BPXF270I message that is shown in Figure 35-5 is issued.

```
BPXF270I THE MAXPIPEUSER LIMIT OF 256 HAS BEEN REACHED BY 262
USER HERING, UID=888
```

Figure 35-5 BPXF270I message about maximum user limit reached

Note: The BPXF270I message is issued the first time that the MAXPIPEUSER limit is reached for the user. The message is not issued again until the user has closed all pipes and then hits the limit again.

Future pipe() requests fail with errno EMFILE and errnojr JrMaxPipesUser (xxxx0649), as shown in Figure 35-6.

```
$> cn "SETOMVS MAXPIPEUSER=256"
 BPXO015I THE SETOMVS COMMAND WAS SUCCESSFUL.
SH> /* Now opening pipes until a usage of 255 is reached */
SH> s "pipe pp."
OMVS Return Value (retval) = 0
SH> /* Now the usage of 256 open pipes has been reached */
SH> s "pipe pp."
OMVS Return Value (retval) = -1
OMVS Return Code (errno )  = 7C
OMVS Reason Code (errnojr) = 5970649
OMVS Return Code Explanation - EMFILE: Too many files are open for this process
OMVS Reason Code Explanation - JrMaxPipesUser: The maximum number of pipes has
been exceeded for this user.
Rc(-1)
SH> exit
$>
```

Figure 35-6 Error indication on pipe request when MAXPIPEUSER limit is reached

D OMVS,PIPES command

The syntax of the command is shown in Example 35-1.

Example 35-1 Syntax of D,OMVS,PIPES

```
D OMVS,PIPES [ ,ALL | ,RESET | ,{UID | U}=uid ]
```

With the **D OMVS,PIPES** command, you can view the two highest-use users, as the sample in Figure 35-7 on page 458 shows.

```
D OMVS,PIPES
BPXO073I 16.53.18 DISPLAY OMVS 142
OMVS     0011 ACTIVE             OMVS=(1C)
PIPE OWNER SUMMARY        MAXPIPEUSER=256
                         CURRENT   HIGHWATER
USERID            UID    USAGE      USAGE
IZUSVR            131        8          8
HERING            888        6        256
HIGHWATER USER:
  USERID=HERING UID=888 HIGHWATER USAGE=256
```

Figure 35-7 D OMVS,PIPES

The **D OMVS,PIPES,ALL** command views all pipe users, as shown in Figure 35-8.

```
D OMVS,PIPES,ALL
BPXO073I 16.54.44 DISPLAY OMVS 146
OMVS     0011 ACTIVE             OMVS=(1C)
PIPE OWNER SUMMARY        MAXPIPEUSER=256
                         CURRENT   HIGHWATER
USERID            UID    USAGE      USAGE
*MULTNAM            0        1          5
ARNSVR            205        5          5
IZUSVR            131        8          8
HERING            888        6        256
HIGHWATER USER:
  USERID=HERING UID=888 HIGHWATER USAGE=256
```

Figure 35-8 D OMVS,PIPES,ALL

As shown in Figure 35-9, you reset the `HIGHWATER USE` information with **D OMVS,PIPES,RESET**.

```
D OMVS,PIPES,RESET
BPXO073I 16.55.47 DISPLAY OMVS 150
OMVS     0011 ACTIVE             OMVS=(1C)
PIPE OWNER SUMMARY        MAXPIPEUSER=256
                         CURRENT   HIGHWATER
USERID            UID    USAGE      USAGE
IZUSVR            131        8          0
HERING            888        6          0
NO HIGHWATER USER INFORMATION
```

Figure 35-9 D OMVS,PIPES,RESET

To display use information for a specific UID, use this command, as shown in Figure 35-10 on page 459:

```
D OMVS,PIPES,U=uid
```

```
D OMVS,PIPES,U=888
BPXO074I 16.56.38 DISPLAY OMVS 158
OMVS     0011 ACTIVE             OMVS=(1C)
TOTAL CURRENT USAGE=6
                CURRENT    SYSTEM
      PID        USAGE      NAME
    83952447         4     SC74
    67174983         2     SC74
```

Figure 35-10 D OMVS,PIPES,UID=888

Notes regarding the `D OMVS,PIPES,U=`*`uid`* command:

- PIDs for the user with open pipes are displayed. The output is sorted by the CURRENT USAGE column.
- Up to 10 processes for the user are displayed. If there are more processes with open pipes, the informational text “ADDITIONAL PID ENTRIES OMITED” is displayed.
- The SYSTEM NAME indicates the location of the application that opened the pipe. This is needed for a z/OS UNIX shared file system configuration.
- Named pipes (FIFOs) can be opened by a remote user. Unnamed pipes are always local.

35.1.2 Migration and coexistence

In a z/OS UNIX shared file system configuration, you must consider coexistence with earlier z/OS releases and these factors:

- In a z/OS UNIX shared file system configuration, the MAXPIPEUSER default of 8730 should not be changed until all systems are at the current release level.
- An open() of a named pipe (FIFO) from a lower-level system displays with USERID=* and UID=*, as shown in Figure 35-11, because it is unknown.

```
D OMVS,PIPES
BPXO073I 17.05.26 DISPLAY OMVS 596
OMVS     000F ACTIVE             OMVS=(2D)
PIPE OWNER SUMMARY        MAXPIPEUSER=256
                         CURRENT   HIGHWATER
USERID            UID    USAGE      USAGE
*                   *      256        256
HIGHWATER USER:
  USERID=* UID=* HIGHWATER USAGE=256
```

Figure 35-11 D OMVS,PIPES output

35.2 The zlsof command

Before z/OS 2.1, there was no supported UNIX shell-based or TSO-based utility to display processes that are using pipe resources. Now, the **`zlsof`** command, has been ported and enhanced. This offers several benefits:

- It provides information about open pipe, file series, and socket use.
- It provides the capability to filter output based on ASID, PID, userid, jobname, file, or file system.
- It displays BRLM lock holders and waiters.
- It displays deleted but in-use files.
- It provides file system read/write monitoring capability.

35.2.1 Invocation and use

The **zlsof** command can be executed in the following environments:

- Shell command
- TSO command
- System REXX command (from a console)

The information that is provided depends on the authority of the user:

- Unauthorized users can display only information for their own processes.
- Authorized users can display all file use.

Syntax of the zlsof command

Example 35-2 shows the syntax.

Example 35-2 zlsof command syntax

```
zlsof [-p[pids]|[-a[asids]|[-j[jobs]] [-u[users]] [-c] [-d] [-t] [-i] [-l] [-n]
      [-su] [-v] [-m maxtime] [-rw[seconds]] [pathname|pipe|socket]
```

There are three possible operands: `pathname`, `socket`, and `pipe`. These options are described in Table 35-1. The following conditions apply:

- If no operand is specified, all opened files for processes that the user has authority to view are displayed.
- If the operand is a mountpoint directory, all open files in use for the file system are displayed.
- If a socket or a pipe is specified, summary use per PID is displayed.

Note: The (*) notation in the following table means that

Table 35-1 zlsof options

Option	Description
`-p` *pid*	PID filter
`-j` *job*	Job filter
`-a` *asid*	ASID filter
`-u` *user*	User filter
`-c`	Omits the current directory and root
`-d`	Displays all deleted, but open files[a]
`-t`	Shows the tally of file types open by process

Option	Description
-i	Shows the inode numbers instead of file names
-n	Derives the real path names for files and is disabled after *maxtime* seconds, as specified with -m option
-su	Sets the effective UID to 0 (see footnote)
-l	Indicates files with byte range locks (see footnote)
-v	Verbose mode
-m *maxtime*	No longer derives the file names after maxtime seconds and is used in conjunction with -n option
-rw *seconds*	Monitors matching file systems for read and write activity for a specified time; seconds specifies interval length (see footnote)

a. Use requires that you must have UID=0 or be permitted to access the BPX.SUPERUSER resource in the FACILITY class.

Filtering on userid

Figure 35-12 displays all files used by user CEA.

```
$> zlsof -u cea -su
zlsof version=130111
Searching for all file usage by user CEA
Command       PID User File System             Mountpoint  Inode/file
CEAPSRVR 16842763 CEA  PLEX75.SYSPLEX.ROOT.ZFS /           r 1
                       CEA.HFS                 /u/cea      c 1
CEAPSRVR 16842763 CEA                          socket: 1
zlsof End of output
$>
```

Figure 35-12 Example 1, filter on userid

Show pipe use

Figure 35-13 lists all processes with open pipes for their own unauthorized user ID.

```
$> echo $(id -un)'('$(id -u)')'
HERING(888)
$> zlsof pipes
zlsof version=130111
Searching for use of PIPE
Command       PID User   File System Mountpoint  Inode/file
OMVS     67174983 HERING             pipe: 2
zlsof End of output
$>
```

Figure 35-13 Example 2, shows pipe use

Show socket use

Figure 35-14 on page 462 lists all processes for user TCP/IP with open sockets.

```
$> zlsof socket -u tcpip -su
zlsof version=130111
Searching for use of SOCKET by user TCPIP
Command       PID User  File System Mountpoint  Inode/file
EZBTCPIP 16842769 TCPIP             socket: 1
EZACFALG    65554 TCPIP             socket: 1
EZASASUB    65555 TCPIP             socket: 1
PORTMAP  33619995 TCPIP             socket: 2
FTPD        65565 TCPIP             socket: 2
FTPD        65566 TCPIP             socket: 2
zlsof End of output
$>
```

Figure 35-14 Example 3, show socket use

Monitoring file system use

Figure 35-15 shows output for monitoring the file system mounted at `/etc`.

```
$> zlsof -rw 30 -su /etc
zlsof version=130111
Searching for use of file system OMVS.SC74.ETC
Command                PID User     File System   Mountpoint  Inode/file
/usr/sbin/syslogd 16842766 OMVSKERN OMVS.SC74.ETC /SC74/etc   229 /etc/log
                                    OMVS.SC74.ETC /SC74/etc   229 /etc/log
/usr/sbin/inetd      65558 OMVSKERN OMVS.SC74.ETC /SC74/etc   950
/etc/inetd.pid
FTPD                 65565 TCPIP    OMVS.SC74.ETC /SC74/etc/  c .
FTPD                 65566 TCPIP    OMVS.SC74.ETC /SC74/etc/  c .

30 second monitor results
     Reads     Writes File System
        16          0 OMVS.SC74.ETC
zlsof End of output
$>
```

Figure 35-15 Example 4, monitoring file system use

Using zlsof from a console by using System REXX

If you log on to a console as an OMVS privileged user, you can list information by using **`zlsof`** with System REXX, as shown in Figure 35-16 on page 463.

```
F AXR,ZLSOF -SU '/etc'
AXR0500I AXREXX OUTPUT DISPLAY 513
EXECNAME=ZLSOF REQTOKEN=00004000000000000CB89E488612ECCD2
zlsof version=130111
Searching for use of file system OMVS.SC74.ETC
Command                PID User     File System   Mountpoint  Inode/f+
ile
/usr/sbin/syslogd 16842766 OMVSKERN OMVS.SC74.ETC /SC74/etc   229 /et+
c/log
                                    OMVS.SC74.ETC /SC74/etc   229 /et+
c/log
/usr/sbin/inetd      65558 OMVSKERN OMVS.SC74.ETC /SC74/etc   950 /et+
c/inetd.pid
FTPD                 65565 TCPIP    OMVS.SC74.ETC /SC74/etc/  c .
FTPD                 65566 TCPIP    OMVS.SC74.ETC /SC74/etc/  c .
zlsof End of output
```

Figure 35-16 Example 5; using zlsof from a console

Tip: The **zlsof** command executes only on the local system, not Parallel Sysplex-wide. Therefore, using **route *all,f axr,zlsof** is a way to run it on all systems in parallel.

Changes to callable and REXX services

There are two changes to callable and REXX services that the **zlsof** command uses:

- Changes are implemented in the BPX1GTH callable service, Get thread data.

  PgthaFileData output now returns the file diagnostic name if PgthaFilePath is also set. Review the BPXYPGTH mapping macro for details.

- The z/OS UNIX REXX procinfo() function is extended to return file path name data. There is a `FILEPATH` request type for this purpose. The output returns two suffixes:
  - The `PATH` suffix is the returned portion of the path name.
  - `PATHTRUNC` is set to 1 if the returned path is truncated.

35.3 I/O count correction in SMF records

Installations need a better way of knowing which files or directories are heavily accessed. This can be used for determining performance issues related to heavily accessed file systems. The current SMF type 92 subtype 11 (close) records tend to produce too much output and do not give a good indication of what files are heavily accessed.

z/OS 2.1 provides these solutions to this problem:

- Sockets and character special files have been moved from subtype 11 records to subtype 16 records. These files no longer appear in subtype 11.
- Subtype 17 has been created for accesses to regular files and directories.

This provides the following benefits:

- It clears subtype 11 records from the large amount of these records caused by sockets and character special files.

- It maintains records for sockets and character special files in the subtype 16 if needed.
- It creates a quick reference for accesses to files.

35.3.1 Invocation and use

This section describes changes to the SMF record types and messages.

Close records for sockets and character special files

The following changes were introduced in z/OS 2.1:

- Sockets and character special file close records no longer appear in type 92 subtype 11 records. If you still want to collect this information, edit the SMFPRMxx PARMLIB member to include type 92 subtype 16, as this example shows:

  ```
  SYS(TYPE(92(16)),...)
  ```

  Subtype 16 records use the same mapping as subtype 11 records.
- A constant was introduced:

  ```
  SMF92#CLSSOCCHARSPEC EQU 16
  ```

Records for accesses to files

Keep these factors in mind for the file access records:

- They record the amount of times that a regular file or directory is accessed and is written on these two occasions:
  - When the internal representation of the file is deleted
  - At the end of the SMF global recording interval (ENFQ3704)

  Note: The count of accesses to the file is cleared after writing the record.

- To write these SMF records, edit your SMFPRMxx PARMLIB member to include type 92 subtype 17, as this example shows:

  ```
  SYS(TYPE(92(17)),...)
  ```
- There is also a constant that has been introduced:

  ```
  SMF92#FILEACC  EQU 17
  ```

Mapping for subtype 17

The mapping for subtype 17 is defined in BPXYSMFR, which is shown in Figure 35-17 on page 465.

```
SMF92FAC  DS   0F
SMF92FAWT DS   BL8       This can be time when internal file control
                         block is released. If SMF92FAIT is on,
                         denotes SMF interval time.
SMF92FAFT DS   X         File type as defined in BPXYFTYP
SMF92FAFG DS   0B        record flag
SMF92FAIT EQU  X'80'     When on SMF92FAWT is SMF interval time
               ORG SMF92FAFG+1
               DS CL2    Reserved
SMF92FAIN DS   F         Inode number
SMF92FADN DS   F         Unique device number
SMF92FATI DS   F         Total accesses to file during interval
SMF92FAPN DS   CL64      Pathname
```

Figure 35-17 Mapping for subtype 17 as defined in BPXYSMFR

Message for subtype 17

There is a message for subtype 17 that occurs only when there is an internal error. When this message is shown, the subtype 17 recording gets disabled, but it can be re-enabled.

The message is shown in Figure 35-18.

```
BPXF265I RECORDING FOR SMF TYPE 92 SUBTYPE 17 HAS BEEN DISABLED

Explanation: An unexpected error occurred while attempting to record an SMF type 92
             subtype 17 record.
System Action: Recording is disabled for SMF type 92 subtype 17 records.
Operator Response: None
Sysprog Response: This is an internal error. Contact your IBM support center. To
                  continue recording SMF type 92 subtype 17 records, first disable the
                  recording of subtype 17 records in SMF. Then wait at least 30 seconds
                  after the end of the next SMF interval before re-enabling the
                  recording of subtype 17 records. For more information about SMF
                  recording, see the topic on customizing SMF in z/OS MVS System
                  Management Facilities (SMF).
```

Figure 35-18 Message BPXF265I about subtype 17

35.3.2 Migration and coexistence

SMF type 92 subtype 11 records no longer contain information about sockets or character special files.

Use SMF type 92 subtype 16 records to get close information about sockets and character special files.

35.4 Automount enhancements

In z/OS 2.1, several automount improvements were introduced:

- There is a capability to recognize last-use information for the AUTOMOUNT managed file systems.

- You can specify root permission for the z/OS file system (zFS) data set.
- There is a generic name match on upper case only.
- The owning system is made identical to the parent if the AUTOMOVE setting is UNMOUNT.
- Support for specifying to use EUID and EGID for the data set owner is added.
- Use of system symbols in the master file is now supported.
- Serialization of automount policy load and append is now possible.
- Mounting a nonexistent data set is no longer attempted.

35.4.1 Invocation and use

This section will help you use the capabilities.

Recognizing last use information

Before z/OS 2.1, there was no capability to recognize the last-use information of the automount managed file systems.

- Now a **-f** option is added to the **/usr/sbin/automount** command to display the last-use information of the automount managed file system. This makes problem determination easier.
- The output of using the **-f** option lists mount point, state, timer, EUID, PID, and job name.

  The state has two values: DURATION and DELAY. If the file system is not mounted, a message is displayed saying that no information can be found.

  The timer is the minutes left for this file to be in this state. The timer value shown can be NOLIMIT if this is specified in the policy.

Figure 35-19 shows a sample of the output when using the **-f** option.

```
$> /usr/sbin/automount -q | tail +2 | head -19
/u

name                 *
filesystem           <uc_name>.ZFS
type                 ZFS
allocuser            space(1,1) storclas(OPENMVS) pathperm(755)
mode                 rdwr
duration             nolimit
delay                10
$> ls -1 /u | grep hering
hering
$> /usr/sbin/automount -f HERING.ZFS
/u/hering
STATE: Duration   TIMER: nolimit
EUID:888  PID:83951844  JOBNAME:HERING
$> ls -1 /u | grep haimo
$> /usr/sbin/automount -f HAIMO.ZFS
FOMF0161I No information found
$>
```

Figure 35-19 Using option -f with /usr/sbin/automount and showing setting pathperm

Specifying zFS root permission

Before z/OS 2.1, the root permission was hard-coded to 750 for the zFS data set that is created by automount. You had to change it manually.

Now, a `PATHPERM` keyword can be added in an ALLOCANY or ALLOCUSER statement to specify the root permission for the zFS. This improves the usability.

Note: Specifying PATHPERM is only for zFS.

In Figure 35-19 on page 466, the **`/usr/sbin/automount -q`** command shows a sample policy that uses `PATHPERM`.

Generic match on uppercase only

Before z/OS 2.1, there was no capability to have matching generic names in uppercase.

- A automount policy statement, `charcase`, allows matching names that are all uppercase. This improves the usability.
- When using the `charcase` keyword in a map file, it has three possible values: `upper`, `lower`, and `asis`.
- You cannot specify `charcase` with the keyword `lowercase`, because those two keywords are mutually exclusive.
- Using `charcase=asis` is equivalent to `lowercase=no`. This is the default.
- Specifying `charcase=lower` is equivalent to `lowercase=yes`.

Figure 35-20 shows an example specifying `charcase=upper`.

```
$> cat /etc/auto.upper.master
/temp.test /etc/auto.upper.map
$> cat /etc/auto.upper.map
name            *
type            ZFS
filesystem      <uc_name>.TEST.ZFS
mode            rdwr
charcase        upper
duration        nolimit
delay           10
$> sudo /usr/sbin/automount -a /etc/auto.upper.master
FOMF0107I Processing file /etc/auto.upper.map
FOMF0108I Managing directory /temp.test
FOMF0108I Managing directory /u
$> cd /temp.test/hering
cd: /temp.test/hering: EDC5129I No such file or directory. (errno2=0x05190050)
$> cd /temp.test/Hering
cd: /temp.test/Hering: EDC5129I No such file or directory. (errno2=0x05190050)
$> cd /temp.test/HERING
$> sudo /usr/sbin/mount -qv /temp.test/HERING
----A- HERING.TEST.ZFS /temp.test/HERING
$>
```

Figure 35-20 Generic match on uppercase only

Owning system made identical to parent if mounted as UNMOUNT

Before z/OS 2.1, in a z/OS UNIX file system sharing environment, if an automount file system (*AMD/...) was mounted from SY2 and its parent file system was mounted from SY1, OMVS shutdown on SY1 would fail to unmount the parent even if the automove attribute was UNMOUNT.

In such a situation in z/OS 2.1, the owning system of the automount file system is made identical to the owning system of the parent. This improves the usability.

There are no external changes. Figure 35-21 shows that the change happens automatically.

```
$> echo We are on system $(sysvar SYSNAME).
We are on system SC74.
$> df -v /SC75/temp.test | grep Automove=
File System Owner : SC75        Automove=U      Client=N
$> cat /etc/auto.upper.master
/SC75/temp.test /etc/auto.upper.map
$> sudo /usr/sbin/automount -a /etc/auto.upper.master
FOMF0107I Processing file /etc/auto.upper.map
FOMF0108I Managing directory /SC75/temp.test
FOMF0108I Managing directory /u
$> df -v /SC75/temp.test | grep Automove=
File System Owner : SC75        Automove=U      Client=N
$> sudo /usr/sbin/mount -qv /SC75/temp.test
----U- *AMD/SC75/temp.test /SC75/temp.test
$>
```

Figure 35-21 Owning system made identical to parent if mounted as UNMOUNT

Support for specifying to use EUID and EGID for the data set owner

Before z/OS 2.1, is was possible to use UID and GID only for the owner of the file system. This caused problems in task-level security.

In z/OS 2.1, the EUID keyword can be used in an ALLOCANY or ALLOCUSER statement to indicate that the owner should be set to EUID and EGID. This improves the usability.

An example of using EUID in an ALLOCUSER statement is shown in Figure 35-22.

```
$> cat /etc/auto.uid.map | grep allocuser
allocuser            space(1,1) storclas(OPENMVS) euid
$>
```

Figure 35-22 Specifying to use EUID and EGID for a file system

Support for using system symbols in the master file

In z/OS 2.1, support has been added to recognize system symbols in the master file. This improves the usability.

Symbols are resolved when loading the AUTOMOUNT policy, as shown in Figure 35-23 on page 469.

```
$> cat /etc/auto.upper.master
/&SYSNAME/temp.test /etc/auto.upper.map
$> sudo /usr/sbin/automount -a /etc/auto.upper.master
FOMF0107I Processing file /etc/auto.upper.map
FOMF0108I Managing directory /SC74/temp.test
FOMF0108I Managing directory /u
$> sudo /usr/sbin/mount -qv /SC74/temp.test
----U- *AMD/SC74/temp.test /SC74/temp.test
$>
```

Figure 35-23 Using system symbols in the master file

Serialization of automount policy load and append

Before z/OS 2.1, automount policy loads or appends were not serialized. In a z/OS UNIX file system sharing environment, processing from different systems at the same time could cause policy overlays.

In z/OS 2.1, policy loads or appends are serialized, which improves usability and prevents that problem.

Attempts to mount non-existent file systems

Before z/OS 2.1, delays might have occurred when automount attempted to mount many non-existent file systems.

In z/OS 2.1, automount does not attempt to mount a data set if it does not exist. In that situation, the automount request fails before it calls a mount operation. This improves performance.

35.4.2 Migration and coexistence

In z/OS UNIX file system sharing environment, all of the systems must be running at level z/OS 2.1 or later for the `PATHPERM`, `EUID`, and `CHARCASE` keywords to work correctly.

35.5 TFS enhancements

Before z/OS 2.1, space management for a temporary file system (TFS) could be difficult because there were no warnings when space was critical and no capability to dynamically increase the size. Also, a TFS did not provide reason codes for error conditions.

Now, a TFS issues messages similar to HFS and ZFS as space becomes critical, and the TFS provides unique reason codes for error conditions. Also, the TFS can be configured to increase in size. Both of these improvements simplify the management of TFS file systems.

35.5.1 Invocation and use

To use the capabilities, consider the following factors:

- TFS global defaults can be specified in the FILESYSTYPE statement in BPXPRMxx. Example:

```
FILESYSTYPE TYPE(TFS) ENTRYPOINT(BPXTFS)
ASNAME(TFS,'SUB=MSTR')
PARM('-fsfull(80,5) -ea 10 -em 10')
```

- By using **MODIFY**, global defaults can be changed or a file system can be increased manually, as these two examples show:

```
F tfs,fsfull(80,5)
F tfs,grow dev
```

- There is limited diagnostic query information available by using MODIFY, for example:

```
F TFS,Q
```

- TFS-specific defaults can be specified on the mount parameter. Example:

```
MOUNT FILESYSTEM(DEV) TYPE(TFS) MOUNTPOINT('/dev')
PARM('-s100 -fsfull(80,5) -ea(10) -em(10)')
```

Changed messages

There are several and changed messages:

- BPXTF107I *filesystem* SIZE=*size* MAX FILE SIZE=*size2* FSFULL(*threshold*,*increment*)

  This message is issued after a successful mount.

  It replaces message BPXT007I.

- The following messages are returned for FSFULL monitoring:

```
BPXTF009E FILESYSTEM EXCEEDS percent% FULL: name
BPXTF010E FILESYSTEM IS FULL: name
BPXTF011I FILESYSTEM IS NOW BELOW percent% FULL: name
```

- The following messages are **MODIFY** responses:

```
BPXTF012I GLOBAL SETTINGS: fsfull(threshold,increcmet) ea growa em growm
BPXTF014I FILESYSTEM EXTENDED ea count em count2 name
BPXTF015I MANUAL EXTENDS EXCEEDED FOR name
```

Defining a TFS type and mounting a file system

In this section, we use a scenario to demonstrate a TFS prepared in the PARMLIB member BPXPRMTF, as shown in Figure 35-24.

```
FILESYSTYPE TYPE(TFS2)
   ENTRYPOINT(BPXTFS)
   ASNAME(BPXTEMP,'SUB=MSTR')
   PARM('-fsfull(80,5) -ea 10 -em 10')
```

Figure 35-24 PARMLIB member BPXPRMTF

Figure 35-25 shows activation of the TFS.

```
SET OMVS=(TF)
IEE252I MEMBER BPXPRMTF FOUND IN SYS1.PARMLIB
BPXO032I THE SET OMVS COMMAND WAS SUCCESSFUL.
...
BPXTF012I GLOBAL SETTINGS: fsfull(80,5) ea 10 em 10
```

Figure 35-25 Starting TFS2

Figure 35-26 shows a sequence of mounting and filling a TFS.

```
$> sudo /usr/sbin/mount -f /tfstest -t tfs2 -o '-s 1' /tmp/tfstest
$> df -k /tmp/tfstest
Mounted on     Filesystem               Avail/Total    Files      Status
/SC74/tmp/tfstest (/tfstest)               900/1024       225        Available
$> # Creating a file /tmp/tfstest/testfile.1 with size 400K
$> df -k /tmp/tfstest
Mounted on     Filesystem               Avail/Total    Files      Status
/SC74/tmp/tfstest (/tfstest)               488/1024       122        Available
$> cp /tmp/tfstest/testfile.1 /tmp/tfstest/testfile.2
$> cp /tmp/tfstest/testfile.1 /tmp/tfstest/testfile.3
$> df -k /tmp/tfstest
Mounted on     Filesystem               Avail/Total    Files      Status
/SC74/tmp/tfstest (/tfstest)               3764/5120      941        Available
$> echo The TFS has been grown by $(rexx "say (5120-1024)/1024")MB.
The TFS has been grown by 4MB.
$>
```

Figure 35-26 Mounting and filling a TFS

In parallel, the messages listed in Figure 35-27 are found in the SYSLOG or OPERLOG.

```
 BPXTF006I TFS MOUNTED /tfstest
 BPXTF107I SIZE=1,048,576 MAX FILE SIZE=2,147,483,648 FSFULL(80,5)
 EA=10 EM=13
 ...
*BPXTF009E FILESYSTEM EXCEEDS 80% FULL: /tfstest
*BPXTF009E FILESYSTEM EXCEEDS 85% FULL: /tfstest
*BPXTF009E FILESYSTEM EXCEEDS 89% FULL: /tfstest
 ...
*BPXTF009E FILESYSTEM EXCEEDS 94% FULL: /tfstest
*BPXTF009E FILESYSTEM EXCEEDS 99% FULL: /tfstest
*BPXTF009E FILESYSTEM EXCEEDS 98% FULL: /tfstest
*BPXTF009E FILESYSTEM EXCEEDS 97% FULL: /tfstest
*BPXTF009E FILESYSTEM EXCEEDS 97% FULL: /tfstest
*BPXTF009E FILESYSTEM EXCEEDS 96% FULL: /tfstest
*BPXTF009E FILESYSTEM EXCEEDS 95% FULL: /tfstest
 BPXTF011I FILESYSTEM IS NOW BELOW 80% FULL: /tfstest
 BPXTF014I FILE SYSTEM EXTENDED ea 9 em 13 /tfstest
```

Figure 35-27 Messages shown in parallel

Figure 35-27 shows that as a result of the automatic grow, the number for the remaining automatic grows went down by 1.

35.5.2 Installation

There are two actions required to use the functions:

- The `FILESYSTYPE` statement for TFS in the BPXPRMxx PARMLIB member needs to be changed with additional specifications.
- To allow the use of the Modify support, the TFS must be run in a colony address space.

35.6 Unicode services use

The problem is that files can reside in the different systems or geographic locations from which they originated. Conversion between code pages might be necessary.

In z/OS 2.1, there is a solution for this problem:

- z/OS UNIX files can now be tagged with any code page.
- Automatic conversion occurs when file I/O is issued by a program.
- z/OS UNIX uses z/OS Unicode services to allow a program to read or write a file that is tagged with a code page and to convert file data to and from the program's code page.

Note: z/OS UNIX now fully participates in text conversion and supports auto conversion for all IBM code pages.

The design is based on the z/OS 2.1 Enhanced ASCII function, which has these characteristics:

- Each program or thread has a code page. The default coded character set identifier (CCSID) is 1047.
- UNIX files can have code pages, which means they have file tags.
- The runtime environment can be enabled for automatic conversion.

35.6.1 Invocation and use

A CCSID can be assigned to a thread or program with the following conditions:

- The default is 1047.
- You can use **_BPXK_PCCSID=*ccsid***.
- A C program can be compiled with the ASCII option. Set CCSID to 819.
- With `fcntl(fd,f_control_cvt,...)`, a program CCSID for an open file can be assigned.

This creates the following possibilities for tagging a file:

- The default is no tag, which means that no conversion is done.
- A tag can be set by using the **chtag** command:

  ```
  /bin/chtag -tc 1208 file1
  ```

- A tag can be propagated by using the **cp** command:

  ```
  /bin/cp file1 file2
  ```

- A C program can be compiled with run option `FILETAG(...,AUTOTAG)`.
- By using `fcntl(fd,f_settag,...)`, a file CCSID for that file can be assigned.
- By using `fcntl(fd,f_control_cvt,...)`, CCSID is assigned only for the duration of this Open action.

Enabling the conversion environment

You can activate a conversion environment in several ways:

- To enable it at the system level, these are the options:
  - Set `AUTOCVT(ALL)` in a BPXPRMxx PARMLIB member to activate at z/OS UNIX startup (see the Attention note that follows, however).
  - Activate it dynamically by using the **SET OMVS=(xx)** command.

- As an alternative, use **SETOMVS AUTOCVT=ALL**.

  Command **D OMVS,OPTIONS** displays the PARMLIB values as in previous versions.

- For a z/OS UNIX shell session or a program, you have these options:
  - In a shell, use the **export _BPXK_AUTOCVT=ALL** command (see the Attention note that follows, however).
  - In a program, use the following statement:

    `setenv("_BPXK_AUTOCVT","ALL",1);`

- For a single open file before first read or write, using `fcntl(fd,f_control_cvt,....)` enables conversion for this open file only.

Attention: It is best to enable the conversion environment at the finest granular level possible.

Using `AUTOCVT(ALL)` in BPXPRMxx can have undesired effects, because all users and programs are enabled for conversion. In that case, any tagged file would be converted during UNIX reads and writes.

Figure 35-28 shows an example using the function at the shell level.

```
$> unset _BPXK_AUTOCVT
$> echo "aaaaaaaaaa\nöäüßÖÄÜ\nzzzzzzzzzz" | iconv -f1047 -t273 > testfile.273
$> chtag -tc273 testfile.273
$> _BPXK_AUTOCVT=ALL cat testfile.273
aaaaaaaaaa
öäüßÖÄÜ
zzzzzzzzzz
$>
```

Figure 35-28 Using auto conversion at the shell or command level

Details about the implementation

The function is implemented in the z/OS UNIX logical file system (LFS).

Standard read/write I/O is supported, but virtual file system I/O is not supported.

Unlike enhanced ASCII, character special file conversion is not allowed.

Note: Existing ASCII programs using standard streams need auto conversion set to `ON`.

REXX syscall commands such as this example are also supported:

`f_settag, f_control_cvt, environment()`

Here are syntax examples of important environment variables used during conversion:

- For enabling auto conversion of a tagged file:

  `_BPXK_AUTOCVT=OFF | ON | ALL`

- For identifying the program CCSID:

  `_BPXK_PCCSID=`*`ccsid`*

- For Unicode services conversion:

```
_BPXK_UNICODE_TECHNIQUE=(LMRECO-9)
```

- For character substitution:

```
_BPXK_UNICODE_SUB=YES | NO
```

- For a malformed character action:

```
_BPXK_UNICODE_MAL=YES | NO
```

Tip: Set these environment variables before the first read or write operation.

There is a pass-through of Unicode services conversion errors. z/OS UNIX reason code `E400ccrr` and `E401ccrr` are Unicode services errors with a return code of `cc` and a reason code of `rr`.

The z/OS UNIX LFS gets three large buffers to hold and convert the data. The following conditions apply:

- For reads, any extra data is discarded when the I/O completes.
- For reads and writes, any ending partial character is cached for the next read or write.

Note: The `close()` function can cause the loss of a (cached) character.

- Only converted data is supplied to the program or file.
- The read or write return value reflects the bytes supplied to or from the program, not the amount read from or written to the file.
- The internal buffers used for conversion are stored in a persistent kernel above the bar storage (64-bit).

Excessive use can cause significant paging and virtual 31 storage accumulation, so the following option is useful:

- Using the PARMLIB keyword `MAXIOBUFUSER(nnnnn)` limits each user's storage for conversion to *nnnnn* MB:
  - Setting `MAXIOBUFUSER(2G)` means 2 P (petabytes) of storage.
  - `MAXIOBUFUSER(2048)` indicates 2 G. This is the default.
  - The **SETOMVS MAXIOBUFUSER=*nnnnn*** command is supported.

There are some of the multi-threaded effects:

- Each thread that shares a single open file must have the same program CCSID.
- Simultaneous reads or writes that result in partial characters can cause an error.
- Additional threads use temporary above-the-bar storage for buffers.

Problems with lseek

There are several problems that can result from using the lseek() function:

- Using lseek() to go to a non-character boundary, followed by a read or write
- Using seek() from one code page to another (MBCS files)
- Using lseek() to go to a non-character boundary, causing a false positive conversion

Notes:

- An lseek() to other than the current position or file beginning results in a subsequent I/O conversion error for DBCS or MBCS files.
- Character boundary problems occur only with DBCS and MBCS code pages.
- Sequential reading or writing is the preferred I/O method when converting.

REXX example conversion, one byte at a time

The REXX example used here converts one byte at a time. Figure 35-29 lists the contents.

```
/* REXX - convert 1 byte at a time for a tagged file      */
address syscall
parse arg file
'open' file o_rdonly
fd = retval
ccsid = '0000'x
'f_control_cvt (fd) cvt_setcvtall ccsid ccsid'
do forever
  'read (fd) buf 1'
  if retval=-1 then do
    address sh 'bpxmtext' errno
    address sh 'bpxmtext' errnojr
    'lseek (fd) 0' seek_cur
    say 'file position for this error =' retval
    leave
  end
  if retval = 0 then leave
end
exit
```

Figure 35-29 REXX example converting one byte at a time

BPXMTEXT in z/OS 2.1:

- The BPXMTEXT utility can now display information for error numbers as 1 - 4 hexadecimal characters.
- Using option `-d` shows the reason or error numbers as a decimal numbers.

Figure 35-30 on page 476 shows the output when using REXX for specific file (unifile) that is tagged with character code set 1200.

```
$> ls -T unifile
t 01200       T=on  unifile
$> _BPXK_AUTOCVT=ALL cat unifile
cat: FSUM6180 file "unifile": EDC5122I Input/output error. (errno2=0xE4000406)
$> _BPXK_UNICODE_SUB=YES _BPXK_AUTOCVT=ALL cat unifile
cat: FSUM6180 file "unifile": EDC5122I Input/output error. (errno2=0xE4000407)
$> _BPXK_UNICODE_SUB=YES myrexx unifile
122(7Ax) EIO: An I/O error occurred
An error occurred while reading or writing a file that was being converted
using z/OS Unicode Services. The last two bytes of this reason code correspond
to the Unicode return code (04) and reason code (07) respectively. See z/OS
Unicode Serivces User's Guide and Reference for more information.
file position for this error = 12
$>  wc -c unifile
         13    unifile
$> _BPXK_AUTOCVT=ALL _BPXK_PCCSID=1200 myrexx unifile
$>
```

Figure 35-30 Using the REXX for a file tagged with character code set 1200

The sample processing demonstrates possible problems, especially lseek(), as described previously. For additional information, we provide the explanation for the Unicode return and reason codes in Table 35-2.

Table 35-2 Explanation for the Unicode return and reason codes

Return code	Reason code	Meaning and action
4	6	A character was found in the source buffer which cannot be converted into a TO-CCSID character and the CUNBNPRM_Sub_Action flag specifies terminate with error. ... Turn on the Sub_Action flag to replace the invalid character with the target substitution character and call the conversion service again.
4	7	An incomplete character was found in the source buffer. This error happens when not all bytes of a multi-byte character are found in the source buffer. For example, the incomplete character can be found at the end of the source buffer if only the first byte of a double-byte character fits into the buffer. ... Make sure that the missing bytes are in the source string.

35.7 REXX enhancements

The following enhancements to z/OS UNIX REXX services are described in this section:

- Changes for Bpxwdyn()
- Changes for Bpxwunix()
- Several syscall services not yet available to REXX
- Enhancements to the *Address TSO* environment

35.7.1 Invocation and use

Using these enhancements is described in more detail in the subsections that follow.

Changes for Bpxwdyn()

Before z/OS 2.1, `Bpxwdyn()` was limited in the environments that it could be used in and the functions that it could perform.

In z/OS 2.1, this has been addressed, and Bpxwdyn() has been enhanced accordingly:

- Bpxwdyn() has been extended so that it may be called from non-REXX environments for the information retrieval functions.

  It enabled a simple interface for the non-REXX applications to do so.

- Additional SVC99 keys have been added, which enables Bpxwdyn() to be more broadly used.

  There are several keywords. See *z/OS V2R1.0 Using REXX and z/OS UNIX System Services* in the IBM Knowledge Center for the additional supported keys:

  `https://ibm.biz/BdFd74`

- Almost all information retrieval keys are now supported.

  Note: For example, the *TU* key allows you to build an allocation text unit from a specified string. Using `alloc tu(000100010002c1c2)` is the same as `alloc fi(ab)`.

- Some debug aids have been added:
  - The *DUMP* key can be used to obtain a TDUMP before or after the SVC99.
  - The *DIAG* key can be used to show information such as the version, Bpxwdyn() input, key parsing information, and the generated text units.

Several enhancements allow better calling of Bpxwdyn() outside of REXX:

- Bpxwdyn() had always supported non-REXX calls with the restriction that information was not returned to the application for keys that return information. Multiple parameters can now be specified.
- On input, the additional parameters specify parameter area length and the text key.
- On output, the parameters contain information that is returned by SVC99 for the corresponding key.

  See *z/OS V2R1.0 Using REXX and z/OS UNIX System Services*, SA23-2283, in the IBM Knowledge Center for the parameter format:

  `https://ibm.biz/BdFd74`

Example 35-3 provides the sample C program from the manual.

Example 35-3 Sample C program named calloc.c for dynalloc processing

```
typedef int EXTF();
#pragma linkage(EXTF,OS)
#include <stdlib.h>
#include <stdio.h>
void main() {
EXTF *bpxwdyn=(EXTF *)fetch("BPXWDYN ");
int i,j,rc;
char *alloc="alloc delete";
```

```
typedef struct s_rtarg {
   short len;
   char  str[260];
   } RTARG;
RTARG ddname = {9,"rtddn"};
RTARG dsname = {45,"rtdsn"};
RTARG msg    = {3,"msg"};
RTARG m[4]   = {258,"msg.1",258,"msg.2",258,"msg.3",258,"msg.4"};
rc=bpxwdyn(alloc,&dsname,&ddname,&msg,&m[0],&m[1],&m[2],&m[3]);
if (rc!=0) printf("bpxwdyn rc=%X\n",rc);
if (*ddname.str) printf("ddname=%s\n",ddname.str);
if (*dsname.str) printf("dsname=%s\n",dsname.str);
for (i=0,j=atoi(msg.str);i<j && i<4;i++)
   printf("%s\n",m[i].str);
}
```

After compilation, it can be used for demonstration from a z/OS UNIX shell, as shown in Example 35-4.

Example 35-4 C program calloc

```
$> calloc
ddname=SYS00001
dsname=SYS13174.T131552.RA000.HERING3.R0100087
$>
```

Bpxwunix() changes

Before z/OS 2.1, it was very difficult to run a login shell or background process through Bpxwunix(). Also, the Bpxwunix() line length was too restrictive.

In z/OS 2.1, two changes solve these problems:

- Two additional arguments have been added to Bpxwunix() to direct it to run a login shell or a background process.
- The Bpxwunix() line length has been increased from 2K to 16K.

This simplifies starting an asynchronous process from REXX and allows for long output lines.

In Table 35-3, the arguments are listed and briefly described.

Table 35-3 Arguments for Bpxwunix()

Argument	Values	Explanation
6	0 1	A login shell is not used. As before, **/bin/sh -c** is used. This is the default. A login shell is used. The **/bin/sh -L**c command is used instead.
7	0 1	The command runs in the foreground synchronously. This is the default. The command runs in the background, under /bin/nohup, asynchronously.

Tips:

- Capturing output for a background process through Bpxwunix() is not supported.
- Consider redirecting stdout and stderr files.

Syscall services available to REXX

In releases before z/OS 2.1, several sets of useful services were not available to REXX programs. The following sets of services are now available:

- Message queues
- Shared memory
- Shared memory locks
- File shares through the file server interfaces

Note: This extends the types of applications that can be implemented in REXX.

Message queue services include those shown in Table 35-4.

Table 35-4 Message queue services syscall commands

Message queue service	Function
msgget	Locates or creates a message queue.
msgsnd	Queues a message in a message queue.
msgrcv	Receives a message from a message queue.
msgstat	Gets the status information for a message queue.
msgset	Sets attributes for a message queue (qid, uid, gid, permissions and maximum size on queue).
msgrmid	Removes a message queue.

Note: See *z/OS V2R1.0 Using REXX and z/OS UNIX System Services* for exact syntax and use of the message queue commands:

https://ibm.biz/BdFd74

Figure 35-31 on page 480lists the content of a small REXX example.

```
$> cat $(command -v rexx.msg)
/* REXX rexx.msg */
Parse Arg msg
Address Syscall
"msgget TSTQ"
If retval<>-1 Then "msgrmid (retval)" /* assures queue not existent */
"msgget TSTQ"
qid=retval
Call Ckerr "msgget"
type = "00000002"x
"msgsnd" qid "msg" Length(msg) "1 type"
Call Ckerr "msgsnd"
type = "00000000"x
"msgrcv" qid "input 20 1 type"
Call Ckerr "msgrcv"
Say input
Say "type=" C2x(type)
Exit
Ckerr:
If retval<>-1 Then Return
Say "Error on" Arg(1)
Say retval errno errnojr
Address SH "bpxmtext" errno
Address SH "bpxmtext" errnojr
Exit 1
$>
```

Figure 35-31 Code listing for rexx.msg

Figure 35-32 shows two cases of using the REXX. One gets an error because the message is too long.

```
$> rexx.msg "A too long sample message..."
Error on msgrcv
-1 91 708030D
145(91x) E2BIG: The parameter list is too long, or the message to receive was
too large for the buffer
BPXGXQRC 02/19/13
JRMsq2Big: The message to receive was too large for the buffer and MSG_NOERROR
was not specified

Action: Increase the buffer size, or specify MSG_NOERROR.
$> rexx.msg "A sample message..."
A sample message...
type= 00000002
$>
```

Figure 35-32 Using rexx.msg

Shared memory services include those shown in Table 35-5.

Table 35-5 Shared memory service syscall commands

Shared memory service	Function
shmget	Locates or creates a shared memory segment.
shmat	Attaches a shared memory segment.
shmdt	Detaches a shared memory segment.
shmstat	Gets the status information for a shared memory segment.
shmset	Sets attributes for a shared memory segment (id, uid, gid, and permissions).
shmrmid	Removes a shared memory segment.

Notes:

- The REXX `storage()` function can be used to read and modify the storage.
- See *z/OS V2R1.0 Using REXX and z/OS UNIX System Services* for exact syntax and use of the shared memory syscall commands:

  https://ibm.biz/BdFd74

Shared memory lock services include those shown in Table 35-6.

Table 35-6 Shared memory lock service syscall commands

Shared memory lock service	Function
shmlkinit	Creates a shared memory lock.
shmlkdestroy	Deletes a shared memory lock.
shmlkobtain	Locks a shared memory lock.
shmlkrelease	Unlocks a shared memory lock.

Shared memory locks have the following characteristics:

- They can be obtained as shared or exclusive.
- They can be created with optional deadlock detection.
- They can be created with optional recursive locking.

Note: See *z/OS V2R1.0 Using REXX and z/OS UNIX System Services* in the IBM Knowledge Center for the syntax and use of the shared memory lock syscall commands:

https://ibm.biz/BdFd74

Virtual file system (VFS) server syscall commands are shown in Table 35-7.

Table 35-7 VFS server syscall commands

VFS syscall command and parameters	Function or explanation
v_open	Opens an existing file.
vntoken	A variable name that contains the vnode token.
accessintent	A number used to specify the intended access mode: o_rdonly, o_wronly, or o_rdwr.
denyvalue	A number used to specify access modes to be denied: o_rdonly, o_wronly, or o_rdwr.
opentoken	A variable name that will contain the open token on successful completion of the service.
v_close	Closes a file previously opened using v_open.
vntoken	A variable name that contains the vnode token.
opentoken	A variable name that contains the open token as returned from the v_open.
v_setattro	Same as v_setattr but parm 2 is an open token.
vntoken	A variable name that contains the vnode token.
opentoken	A variable name containing the opentoken from v_open.
attribute_list	A list of attributes to be set and their values.

Figure 35-33 shows a sample REXX trying to get a file that was previously resolved to vntok exclusively and truncate it to size 0.

```
/* REXX Trunc2zero */
Address Syscall
"v_open vntok" o_rdwr o_rdwr "opentok"
if retval=-1 then do
  say "Entry is probably in use."
  exit 1
end
"v_setattro vntok opentok" st_size 0
"v_close vntok opentok"
```

Figure 35-33 Sample REXX to truncate a file to zero

Enhancements to the Address TSO environment

Before z/OS 2.1, open file descriptors were not inherited by the Address TSO co-process.

z/OS 2.1 adds an environment variable that directs `Address TSO` to pass open file descriptors. This allows for a TSO program run through `Address TSO` to access file descriptors opened by the calling REXX program.

The function is enabled by setting environment variable `BPXWRFD=YES`.

Restriction: Only descriptors 10-99 can be passed to the TSO program.

The environment variable can be set outside of the REXX program, or the REXX program can use the `Environment()` function to set the variable.

35.8 Avoiding access loss for previous default users

Many installations recognized that they had used the BPX.DEFAULT.USER support even more than they desired. When this profile is removed and is no longer available, all of those users and service functions lose access to their files.

Now there is a tool to keep this access without the need to clear up everything.

35.8.1 Invocation and use

There are several steps involved. The most important are these two:

1. List all of the files that have the (previous) default UID or GID assigned
2. Change access permission to keep the access for the default users after getting their own OMVS segment.

List the entries

When searching for the files in a z/OS UNIX file system sharing environment, it is important to search only in file systems that are owned locally. Otherwise, this could result in an undesired and huge cross-system coupling facility (XCF) traffic.

Attention: This is also true for a z/OS file system (zFS) that is mounted as Parallel Sysplex-aware. The zFS DIO support does not include complex metadata processing. The goal of DIO is access to the data of the UNIX entries.

For each file system, search according to these guidelines:

- If it is a zFS, search on the system that owns the zFS files.
- If it is not a zFS, search on the system than owns z/OS UNIX files.

A utility named XFIND processes **`Find`** commands by searching only the owning system. Example 35-5 shows a sample **`XFIND`** processing in a two-way Parallel Sysplex.

Example 35-5 Listing the entries

```
ROUTE *ALL,F AXR,XFIND,T=0 '/ -user oedfltu -o -group oedfltg'
 F AXR,XFIND,T=0 '/ -user oedfltu -o -group oedfltg'
 F AXR,XFIND,T=0 '/ -user oedfltu -o -group oedfltg'
 XFIND on MCEVSI  : See USS files /u/hering/XFIND.MCEVSI.* for results.
 XFIND on MCEVSM  : See USS files /u/hering/XFIND.MCEVSM.* for results.
...
 XFIND on MCEVSI  : Processing ended.
 XFIND on MCEVSM  : Processing ended.
```

This creates or overwrites the following files in the XFIND user's home directory:

- XFIND.MCEVSI.error_log
- XFIND.MCEVSI.msg_log
- XFIND.MCEVSI.uss_entries
- XFIND.MCEVSM.error_log
- XFIND.MCEVSM.msg_log
- XFIND.MCEVSM.uss_entries

Note: You can also run the tool with the name **`lfind`** locally from a z/OS UNIX shell.

Set up group ACLs for the default user files that are found

The second step is to modify the access authorization for all of the UNIX entries that you find. We provide a tool named *defa4all* (DEFault user entries Access for ALL users) for this purpose. The task is done either by modifying the other permission bits or by introducing a group, creating ACLs, and giving access to this group.

We show only the ACL approach, because authorization by other bits is not used when the user keeps access through the group owner. If the group is not the old default group and the group permission is lower, we do not get all of the permissions that we need.

First, create a group, because a large number of users needs to be connected to it. See Example 35-6.

Example 35-6 Creating a group with a unique GID

```
tso addgroup defa4all omvs(gid(999999)) owner(sys1) supgroup(sys1) universal
```

Attention: Later, you need to connect all of the users that you want to have access to the default files to this group.

Now, look at the files' XFIND.MCEVSI.uss_entries and XFIND.MCEVSM.uss_entries, and select the one in MCEVSM. You can now work with the list inside without any XCF processing, because all involved file systems are mounted in MCEVSM. For this example, we worked in this system, so we had them mounted (or zFS-owned) locally.

Syntax and sample processing of the tool is shown in Figure 35-34 on page 485.

```
F AXR,DEFA4ALL
DEF4A005I DEFA4ALL on MCEVSM                                    - 095
DEFA4ALL ? - displays this help information
DEFA4ALL   - displays this help information
DEFA4ALL uss_entry_list defuser defgrp - all default user or default
-            group entries in the list of files will be made available
-            via appropriate OTHER permission bits set; uss_entry_list
-            is a UNIX file containing z/OS UNIX entry names; defuser
-            and defgrp are the default user and default group
DEFA4ALL uss_entry_list defuser defgrp aclgroup - will do the same by
-            creating ACL entries for the given group with name
-            aclgroup (or GID value)
F AXR,DEFA4ALL,T=0 '/u/hering/XFIND.MCEVSM.uss_entries' OEDFLTU OEDFLTG
DEFA4ALL
...
DEF4A009I UNIX entries processed:        11
DEF4A003I DEFA4ALL processing has been finished.
```

Figure 35-34 Setting up ACL group ACLs

More tips for using the utility:

- Using the parameter setting **T=0** is advisable because the routine might run for a longer time if many files are involved.
- Be careful about too many default user files in read-only mounted file systems, because the file system needs to be switched to and from read-write mode to allow the change.
- For monitoring progress, a message is displayed after every 10000 entries that have been processed and changed.
- The routine can be used from UNIX, TSO, TSO/ISPF, or as a SYSREXX. Always run it with a z/OS UNIX entries file on the system where the file has been filled, using **XFIND** or **lfind** as shown here previously.

At this point, it is up to the organization to choose which users to connect to the ACL group DEFA4ALL and at which time.

Installation

Follow instructions in this section to install the **XFIND** and **DEFA4ALL** utilities.

XFIND

The **XFIND** utility is available from this IBM Redbooks repository:

ftp://www.redbooks.ibm.com/redbooks/SG247035/

1. First, be sure to read the information in the ITSO_Tools.pdf file. It describes how to get and install a utility that is provided on that page.
2. Then, retrieve the XFIND.pdf file, which describes the tool, and the XFIND.TXT file, which contains the installation job.
3. Use the instructions in the ITSO_Tools.pdf file to transfer and install **XFIND** for use as a SYSREXX and z/OS UNIX routine.

DEFA4ALL

The DEFA4ALL utility is available from the additional material page of this book (`defa4all.zip`):

`ftp://www.redbooks.ibm.com/redbooks/SG248140/`

Figure 35-35 provides a sample for installing the tool after you have it retrieved to your workstation without converting it.

```
C:\...\Downloads>ftp wtsc74
Connected to wtsc74.itso.ibm.com.
...
User (wtsc74.itso.ibm.com:(none)): hering
331-Password: xxxxxxxx
...
230-220-FTPMVS1 IBM FTP CS V2R1 at WTSC74.ITSO.IBM.COM, 20:36:45 on 2013-06-28
230-HERING is logged on.  Working directory is "HERING.".
230
ftp> quote site sbd=(1047,819)
200 SITE command was accepted
ftp> cd 'plex75.rexxexec'
250 The working directory "PLEX75.REXXEXEC" is a partitioned data set
ftp> put defa4all
200 Port request OK.
125 Storing data set PLEX75.REXXEXEC(DEFA4ALL)
250-Statistics have been updated for the PDS member that was transferred
250 Transfer completed successfully.
ftp: 12960 bytes sent in 0,05Seconds 270,00Kbytes/sec.
ftp> cd /usr/local/bin
250 HFS directory /usr/local/bin is the current working directory
ftp> put defa4all
200 Port request OK.
125 Storing data set /usr/local/bin/defa4all
250 Transfer completed successfully.
ftp: 12960 bytes sent in 0,01Seconds 1440,00Kbytes/sec.
ftp> quit
221 Quit command received. Goodbye.
...
C:\...\Downloads>
```

Figure 35-35 Installing DEFA4ALL utility

Notes:

- The PLEX75.REXXEXEC data set is a common SYSREXX library in this Parallel Sysplex.
- If you do not have the required access authority to `/usr/local/bin`, you can copy it to your home directory and then move it authorized to the z/OS UNIX directory.
- Change the permission bits for `/usr/local/bin/defa4all` to 755.

35.9 Documentation and reference

See the following IBm publications and resource for more information:

- Appendix A, “Material available to download” on page 535.
- *Character Data Representation Architecture*, SC09-2190

  http://www.ibm.com/software/globalization/cdra
- ITSO z/OS UNIX Tools page:

  ftp://www.redbooks.ibm.com/redbooks/SG247035/
- z/OS MVS section of the z/OS 2.1.0 IBM Knowledge Center (search by number):

  https://ibm.biz/BdFEZP
  - *z/OS MVS Initialization and Tuning Reference*, SA23-1380
  - *z/OS MVS System Commands*, SA38-0666
  - *z/OS MVS System Messages, Vol 3 (ASB-BPX)*, SA38-0670
  - *z/OS V2R1.0 MVS System Management Facilities (SMF)*, SA38-0667
  - *z/OS V2R1.0 Unicode Services User’s Guide and Reference*, SA38-0680
- *z/OS UNIX System Services* of the z/OS 2.1.0 IBM Knowledge Center:

  https://ibm.biz/BdFEZv
  - *z/OS UNIX System Services Command Reference*, SA23-2280
  - *z/OS UNIX System Services File System Interface Reference*, SA23-2285
  - *z/OS UNIX System Services Messages and Codes*, SA23-2284
  - *z/OS UNIX System Services Planning*, GA32-0884
  - *z/OS UNIX System Services Programming: Assembler Callable Services Reference*, SA23-2281
  - *z/OS Using REXX and z/OS UNIX System Services*, SA23-2283
- *z/OS XL C/C++ section of the z/OS 2.1.0 IBM Knowledge Center:*
  - *z/OS XL C/C++ Run-Time Library Reference*, SC14-7314

36

dbx, the IBM z/OS debugger for UNIX source code

This chapter describes the enhancements, functions, and features that were introduced with IBM z/OS Version 2.1 dbx, the debugger for z/OS UNIX System Services (z/OS UNIX):

- 36.1, “Debug optimized code” on page 490
- 36.2, “Documentation and reference” on page 496

© Copyright IBM Corp. 2014. All rights reserved.

36.1 Debug optimized code

In z/OS 2.1, you can debug routines that were “inlined” into the program. Combined with the implicate program snapshot support, this provides a solution to debugging optimized code.

Notes:

- This is the second phase of a solution to allow you to debug optimized code.
- The benefit is that **dbx** can now debug production level code.

36.1.1 Invocation and use

The subsections that follow provide details of the support, with examples.

Inline support

Note: The examples that follow use the dbx facilities to stop code execution for debugging in the inline function called `foo()`.

Example 36-1 shows an inline function.

Example 36-1 Sample routine example.c

```
$> cat example.c
volatile int x,l,y;
inline int foo(void)
{
   x = 2;
   return(x);
}
int main()
{
   volatile int ti;
   l = 0;
   ti = 0;
   ti = foo();
   ti = 0;
   for( ti = 0; ti < 2; ti++)
             x = foo();
}
$> c99 -O2 -qdebug=nohook -g9 example.c
$>
```

The **a.out** executable is created by this compilation.

Note: The **a.out** routine is the dbx default when no name is specified.

Stop in the routine

Stopping in this `foo()` routine is shown in Example 36-2.

Example 36-2 Stopping in the foo routine

```
$> dbx
FDBX1111: dbx args: dbx64
FDBX1112: Current Directory: /u/hering/cc
FDBX0089: dbx for z/OS with 64-bit support.
...
FDBX0252: enter object file name (default is 'a.out', ^D to exit):
[ enter ]
FDBX6432: Processing load module "a.out"
FDBX9997: The loaded module does not contain module map which may lead to bad pe
rformance. Suggest to use dbgld to create module map to the executable before de
bugging.
FDBX6421: Loaded debug data from "/u/hering/cc/example.dbg"
FDBX0150: Debug target is 31-bit.

(dbx64) stop in foo
[1] stop in 'int foo()'
(dbx64) cont
[1] stopped in 'int foo()' at line 4 in file "example.c"  ($t1)
    4      x = 2;
(dbx64) list
    5      return(x);
    6   }
    7   int main()
    8   {
    9      volatile int ti;
   10      l = 0;
   11      ti = 0;
   12      ti = foo();
   13      ti = 0;
   14      for( ti = 0; ti < 2; ti++)
(dbx64) cont
[1] stopped in 'int foo()' at line 5 in file "example.c"  ($t1)
    5      return(x);
(dbx64) cont
[1] stopped in 'int foo()' at line 5 in file "example.c"  ($t1)
    5      return(x);
(dbx64) c
FDBX0114: program exited
(dbx64) quit
$>
```

Stop at line

Example 36-3 on page 492 shows similar processing with the same startup. The Stop command is set for a specific line in `foo`.

Example 36-3 Stop in foo at line

```
(dbx64) list 1,7
    1   volatile int x,l,y;
    2   inline int foo(void)
    3   {
    4      x = 2;
    5      return(x);
    6   }
    7   int main()
(dbx64) stop at 4
[1] stop at "example.c":4
(dbx64) c
[1] stopped in 'int foo()' at line 4 in file "example.c"  ($t1)
    4      x = 2;
(dbx64) c
[1] stopped in 'int foo()' at line 4 in file "example.c"  ($t1)
    4      x = 2;
(dbx64) c
[1] stopped in 'int foo()' at line 4 in file "example.c"  ($t1)
    4      x = 2;
(dbx64) c
FDBX0114: program exited
(dbx64)
```

Local variable support

First, a file, `example2.c`, is displayed and compiled, as shown in Example 36-4.

Example 36-4 Displaying and compiling example2.c

```
$> cat example2.c
int x = 2, y = 3;
inline int foo(int aaa, int bbb)
{
    int local_v = 1;
    long vvvv = 100;
    aaa = aaa * bbb;
    local_v = aaa + bbb + x;
    bbb = bbb + 7;
    return aaa / bbb + local_v;
}
int main()
{
    x = foo (x, y);
    x = x * y;
    x = foo (x, x * y);
    y = x + y;
    x = foo (y - x, y + x);
    return x + y;
}
$> c99 -O2 -g9 example2.c
$>
```

Several variable declarations have been added in the `foo` inline function in Example 36-5 to illustrate local variable support.

Example 36-5 Local variable support

```
(dbx64) stop in foo
[1] stop in 'int foo()'
(dbx64) cont
[1] stopped in 'int foo()' at line 4 in file "example2.c"  ($t1)
    4       int local_v = 1;
(dbx64) list
    5       long vvvv = 100;
    6       aaa = aaa * bbb;
    7       local_v = aaa + bbb + x;
    8       bbb = bbb + 7;
    9       return aaa / bbb + local_v;
   10   }
   11
   12   int main()
   13   {
   14       x = foo (x, y);
(dbx64) stop at 9
[2] stop at "example2.c":9
(dbx64) cont
[2] stopped in 'int foo()' at line 9 in file "example2.c"  ($t1)
    9       return aaa / bbb + local_v;
(dbx64) p local_v
11
(dbx64) p vvvv
100
```

Templates

In Example 36-6, the file, `template.C`, is displayed and compiled.

Example 36-6 Displaying and compiling template.C

```
$> cat template.C
template <class T>
T GetMax (T a, T b)
{
    T result;
    result = (a>b)? a-1 : b-1;
    return (result);
}
int main ()
{
    int i=10, j=60, k;
    long l=70, m=50, n;
    k=GetMax<int>(i,j);
    n=GetMax<long>(l,m);
    return 0;
}
$> xlC -qdebug=nohook template.C
$>
```

Debugging `template.C` is shown in Example 36-7.

Example 36-7 Debugging template.C

```
(dbx64) list 1,8
    1   template <class T>
    2   T GetMax (T a, T b)
    3   {
    4       T result;
    5       result = (a>b)? a-1 : b-1;
    6       return (result);
    7   }
    8   int main ()
(dbx64) stop at 5
[1] stop at "template.C":5
(dbx64) c
[1] stopped in 'int GetMax<int>(int a, int b)' at line 5 in file "template.C"  ($t1)
    5       result = (a>b)? a-1 : b-1;
(dbx64) c
[1] stopped in 'long GetMax<long>(long a, long b)' at line 5 in file "template.C"  ($t1)
    5       result = (a>b)? a-1 : b-1;
(dbx64) c
FDBX0114: program exited
(dbx64)
```

Member functions

Debugging member functions is shown in Example 36-8.

Example 36-8 Member functions debugging

```
(dbx64) file classx.h
(dbx64) list
    1   class x
    2   {
    3      public:
    4        int a,b;
    5        int add()
    6        {
    7            int k;
    8            k = a+b;
    9            return k;
   10        }
(dbx64) stop at 8
[1] stop at "classx.h":8
(dbx64) c
[1] stopped in 'int add()' at line 8 in file "classx.h"  ($t1)
    8            k = a+b;
(dbx64) delete 1
(dbx64) stop in x::add
[2] stop in 'int x::add()'      File /u/dbxteam/classx.h, Line 8
(dbx64) c
[2] stopped in 'int add()' at line 8 in file "classx.h"  ($t1)
    8            k = a+b;
```

Note: This `classx.h` file can be included in hundreds of `*.C` files.

List with executable lines highlighted

Example 36-9 shows listing code lines with executable lines highlighted.

Example 36-9 List with executable lines highlighted

```
(dbx64) set $showcodelines
(dbx64) list 1,12
     1   class x
     2   {
     3      public:
     4        int a,b;
     5        int add()
     6        {
     7            int k;
```

Each line with an asterisk (*****) in front of it represents a location that a breakpoint may be set.

Note: By using implicit program snapshots at high levels of optimization and low levels of debug possibilities, breakpoints can may be set on only certain executable lines. Those lines are identified by an asterisk (*****).

Limited scope support

Example 36-10 shows limited scope support.

Example 36-10 Limited scope support

```
(dbx64) list 1,12
     1   int i = 5;
     2
     3   int main()
     4   {
     5      printf(“%d\n”, i );
     6
     7      int i;
     8      i = 2;
     9      printf(“%d\n”, i );
    10
    11   }
(dbxs64) step
stopped in main at line 5 in file "l.c"  ($t1)
    5       printf("%d\n", i );
(dbx64) p i
5
(dbx64) step 2
5
stopped in main at line 9 in file "l.c"  ($t1)
    9       printf("%d\n", i );
(dbx64) p i
2
(dbx64)
```

36.1.2 Migration and coexistence

These two factors are relevant for using this debug support:

- The **stop in <routine>** command now stops at the first statement in the routine, not the first line.
- The code must be recompiled in z/OS 2.1.

36.2 Documentation and reference

For more information, see these IBM publications in the IBM Knowledge Center for z/OS 2.1.0:

- *z/OS V2R1.0 UNIX System Services Command Reference*, SA23-2280

  http://ibm.co/1oNTs3y
- *z/OS V2R1.0 UNIX System Services Messages and Codes*, SA23-2284

  http://ibm.co/1kedEr5
- *z/OS V2R1.0 UNIX System Services Programming Tools*, SA23-2282

  http://ibm.co/1sHFoJU
- *z/OS V2R1.0 UNIX System Services User's Guide*, SA23-2279

  http://ibm.co/1qRblA7

37

z/OS UNIX shell and utilities

This chapter describes the enhancements, functions, and features in the UNIX System Services (z/OS UNIX) shell and utilities that were introduced with IBM z/OS Version 2 Release 1 (2.1). It includes these sections:

- 37.1, “ASCII or Unicode option on a subset of the utilities” on page 498
- 37.2, “Documentation and reference” on page 502

© Copyright IBM Corp. 2014. All rights reserved.

37.1 ASCII or Unicode option on a subset of the utilities

In previous releases, there were problems when a z/OS UNIX shell and utilities user wanted to control the text conversion of input files that were used by the commands or to run tagged shell scripts under different SBCS locales.

z/OS 2.1 provides these solutions:

- The **–W filecodeset=*codeset*,pgmcodeset=*codeset*** option is added to several shells and utilities commands to enable text conversion.

  This is consistent with support added to **vi** and **ex** in z/OS Version 1.13.

- The **–B** has option is added on several shell and utilities commands to disable automatic text conversion.

  This is consistent with other commands that already have this override support.

- A **_TEXT_CON**V environment variable enables or disables text conversion.

  With automatic conversion enabled, tagged shell scripts can be run under different single-byte character set (SBCS) locales.

These additions provide several benefits:

- They offer more detailed control of text conversion. No file tagging nor environment or system setup are required.
- You can easily override the system's automatic text conversion.
- You can easily enable or disable text conversion for all shell and utilities commands that provide control of text conversion.
- You can easily run tagged shell scripts under SBCS locales.

37.1.1 Invocation and use

The **–W filecodeset=*codeset*,pgmcodeset=*codeset*** option can be used with the commands that are listed in Table 37-1.

Table 37-1 Commands with added option -W

cat	cmp	comm	cut	diff
dircmp	ed	egrep	expand	fgrep
file	grep	head	more	paste
sed	strings	tail	unexpand	uniq
wc				

These two conditions apply to this option:

- The option keywords are case sensitive.
- The only supported values for **pgmcodeset** are IBM-1047 and 1047.

Details of the -W option

The following details for the -W option are also important to keep in mind.

- It performs text conversion from one code set to another when reading from or writing to the file.

- The **filecodeset** and **pgmcodeset** options can be used on files with any file tag.
- If **pgmcodeset** is specified but **filecodeset** is omitted, the default file code set is ISO8859-1, even if the file is tagged with a different code set. The default program code set is IBM-1047.
- When standard input (stdin) is used as an input text file and stdin is not associated with a terminal, the **–W filecodeset** and **pgmcodeset** option is applied to stdin.

Details of the -B option

The **–B** option can be used with the commands that are listed in Table 37-2.

Table 37-2 Commands with added option -B

cat	comm	cut	diff	dircmp
ed	egrep	expand	fgrep	grep
more	paste	sed	unexpand	unique
wc				

Keep these details in mind if you use this option:

- It disables the automatic text conversion of tagged input files. This option is ignored if the filecodeset or pgmcodeset options (**–W** option) are specified.
- When standard input (stdin) is used as an input text file, and stdin is not associated with a terminal, **–B** disables the automatic conversion of stdin.
- The **head**, **strings** and **tail** commands were changed to disable automatic conversion of stdin.

Details of the _TEXT_CONV environment variable

Support for the _TEXT_CONV environment variable can be used with the commands that are listed in Table 37-3.

Table 37-3 Commands that support the _TEXT_CONV environment variable

cat	cmp	comm	cut	diff
dircmp	ed	egrep	ex	expand
fgrep	file	grep	head	more
pack	paste	sed	strings	tail
unexpand	uniq	wc	vi	

Keep these details in mind if you use this option:

- It contains text conversion information for the command.
- The supported value keywords are FILECODESET, PGMCODESET, and DISABLE.

> **Note:** DISABLE means to disable the automatic conversion of tagged files.

- It applies to all commands that support the file code set and the pgmcodeset option by using the **-W** or **-B** option.
- The _TEXT_CONV environment variable is ignored when the filecodeset or pgmcodeset options that are using the **–W** option or the **–B** option are specified.

- The **pack** command supports only the _TEXT_CONV=DISABLE setting.

Important: All commands that support either the **-W** option or the **-B** option perform the requested text conversion (from FILECODESET to PGMCODESET or DISABLE), regardless of the file being used. This is because all automatic text conversion and file tagging is ignored.

Further details

Keep these details in mind if you use these options:

- The **-W** option, **-B** option, and _TEXT_CONV environment variable apply only to the primary input text files that are processed by the command.
- Text conversion for files that are used by the command for reference purposes, such as file lists, configuration, control information, and so on are not affected by the **-W** option, **-B** option, or the _TEXT_CONV envvar.
- Any output (standard output stdout or output files) that is produced by these commands is not affected by the support. The only exceptions to this are output files that are the same as or associated with the primary input files.

Note: For example, the editor commands **ex**, **vi**, **ed**, **sed**, and so on use this exception.

Precedence rules

Remember the following precedence rules:

- The **–W filecodeset=**codeset**,pgmcodeset=**codeset option overrides the **–B** option, the _TEXT_CONV environment variable, and the system's automatic text conversion.
- The **–B** option overrides the _TEXT_CONV environment variable and the system's automatic text conversion.
- The _TEXT_CONV environment variable overrides the system's automatic text conversion. If the DISABLE value is used, along with either the FILECODESET or PGMCODESET values, the DISABLE value is ignored.
- If the **–W filecodeset=**codeset, **pgmcode set=**codeset option, the **–B** option, and the _TEXT_CONV environment variable aren't specified, the system's automatic text conversion rules apply.

Examples

Figure 37-1 shows how to display the type of an untagged text file that contains ISO8859-1 characters.

```
$> ls -T script.ascii
- untagged    T=off script.ascii
$> file -W filecodeset=ISO8859-1,pgmcodeset=IBM-1047 script.ascii
script.ascii:   commands text    - Bourne or POSIX shell script
$>
```

Figure 37-1 Type display for an untagged text file that contains ISO8859-1 characters

Figure 37-2 on page 501 displays the line count of a file that contains EBCDIC characters when automatic conversion is enabled and the file is incorrectly tagged as ISO8859-1.

```
$> echo $_BPXK_AUTOCVT
ON
$> ls -T myMisTaggedFile
t ISO8859-1   T=on  myMisTaggedFile
$> wc -l myMisTaggedFile
      0    myMisTaggedFile
$> wc -lB myMisTaggedFile
     10    myMisTaggedFile
$>
```

Figure 37-2 File is incorrectly tagged as ISO8859-1

Figure 37-3 shows how to perform text conversion from the ASCII code set ISO8859-1 to the EBCDIC code set IBM-1047 for all supported commands.

```
$> export _TEXT_CONV="FILECODESET(ISO8859-1),PGMCODESET(IBM-1047)"
$> cat script.ascii
#! /bin/sh
 echo Hello
$>
```

Figure 37-3 Text conversion from ASCII code set ISO8859-1 to EBCDIC code set IBM-1047

Executing scripts

In previous releases, shell scripts were limited by the rule that the code page in which a shell script is encoded must match the code page of the locale in which it is run.

z/OS 2.1 enables you to run shell scripts that are tagged when automatic conversion is enabled and the locale is an SBCS.

Example 37-1 shows a script that was created in code page IBM-273 and tagged accordingly.

Example 37-1 Script example

```
$> locale | grep LC_SYNTAX
LC_SYNTAX="C"
$> ls -T $(command -v script.273)
t IBM-273     T=on  /u/hering/bin/script.273
$> export _BPXK_AUTOCVT=ALL
$> cat $(command -v script.273)
#! /bin/sh
cat .profile | wc -l | awk '{ print $1 }'
$> script.273
72
$>
```

UNICODE conversion environments

The environments affect the conversion result. This was known already from the **iconv** command and the iconv() function, which have used Unicode Services since z/OS V1.9.

The following conditions apply to the _BPXK_UNICODE_xxxx environment variables:

- Setting the envvar `_BPXK_UNICODE_TECHNIQUE=x (x=R,E,C,L,M,0-9)` can override the default conversion technique when Unicode Services is called. The default value or sequence is `LMREC`.

Unicode definitions:

R	Roundtrip
E	Enforced subset
C	Customized subset
L	Language Environment behavior
M	Modified for special use
0 - 9	User-defined conversions

- Setting envvar `_BPXK_UNICODE_SUB=(YES|NO)` indicates whether the Unicode Services substitute character action is to be applied during translation.
- Setting envvar `_BPXK_UNICODE_MAL=(YES|NO)` indicates whether the Unicode Services malformed character action is to be applied during translation.

37.1.2 Migration and coexistence

Several commands already supported the **`-B`** option. The **`head`**, **`strings`**, and **`tail`** commands did not disable autoconversion of tagged files for standard input.

In z/OS 2.1, these commands are changed to support the **`-B`** option for standard input. Example 37-2 shows a sample with the file referenced in Figure 37-2 on page 501 that is incorrectly tagged.

Example 37-2

```
$> tail +9 < myMisTaggedFile
$> tail -B +9 < myMisTaggedFile
aaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaa
$>
```

37.2 Documentation and reference

The following IBM publications are also helpful:

- *UNIX System Services Command Reference*, SA23-2280, which describes the updates of the shell and utilities commands. *Appendix L, Controlling text conversion for z/OS UNIX shell commands*, describes details about controlling text conversion for shell and utility commands.

  https://ibm.biz/BdFEaS

- *Unicode Services User's Guide and Reference*, SA22-7649-14

  http://ibm.co/1tnk3FQ

38

Enhancements to zFS, the IBM System z file system

This chapter describes the IBM System z file system (zFS) enhancements, functions, and features that were introduced with IBM z/OS Version 2 Release 1 (2.1). It includes the following sections:

- 38.1, “zFS extended or large directory support” on page 504
- 38.2, “Improved zFS salvager” on page 513
- 38.3, “Enhancements for zFS quiesce and unquiesce processing” on page 516
- 38.4, “zFS backup bit” on page 518
- 38.5, “zFS auditfid default setting” on page 519
- 38.6, “Elimination of multi-file system aggregates and cloning” on page 520
- 38.7, “Renamed z/OS file system” on page 521
- 38.8, “Using non-SMS zFS aggregates with extended addressability” on page 521
- 38.9, “Documentation and reference” on page 523

© Copyright IBM Corp. 2014. All rights reserved.

38.1 zFS extended or large directory support

zFS has a performance problem with a directory that has many entries (over 10,000 to millions of entries) due to the current linear search method. A permanent solution is needed to solve this problem.

In z/OS 2.1, zFS supports a directory format (V5) that uses an extensible hashing scheme. The new, optional Version 1.5 aggregate format is available. The current field version is 1.4.

There are multiple ways to convert to the version of the file system.

The functions provide several benefits:

- Large directory performance is improved.
- An increased maximum aggregate size and number of subdirectories is available.
- There is an improved directory `__readdir2()` time for both Version 1.4 and 1.5 aggregates.

> **Note:** The `__readdir2()` function reads a directory entry and gets file information (similar to information of **`ls -l`**). It is also known as `readdir+`.

38.1.1 Invocation and use

Following information is provided how to implement and use the changed functions.

Version 1.5 aggregates

The maximum aggregate size is increased from 4 TB up to 16 TB as for other VSAM clusters. The maximum number of subdirectories is now 4294967293. It was 65535 (=64k-1) before.

- A version 1.5 aggregate can have both the old (v4) and version directory formats (v5).
- Newly created directories will always be of type version 5.
- Conversion programs support changing the aggregate version to 1.5 and optionally converting the directories.

> **Note:** Version 1.4 aggregates remain supported and converting to version 1.5 is optional.

commands and parameters

There is a **`zfsadm convert`** command, which is shown in Figure 38-1.

```
zfsadm convert [{-path <pathname> | -aggrversion <aggrname>}] [-level] [-help]
```

Figure 38-1 zfsadm convert command

- Using the `-path` parameter, you can convert the version 4 directory `pathname` to version 5. This also changes the version of the aggregate to 1.5, if necessary.
- Using the `-aggrversion` parameter, you can change aggregate `aggrname` to version 1.5. No directories are converted.

IOEPRMxx options are shown in Figure 38-2 on page 505.

format_aggrversion=4 | 5
- The default aggregate version remains 1.4.

change_aggrversion_on_mount=On | Off
- Changes aggregate version to 1.5 on RW mount.
- No existing directories are converted.

converttov5=On | Off
- Changes the aggregate version to 1.5 and converts the root directory at mount time.
- Each v4 directory will be converted to v5 when it is first accessed.
- This can be dynamically changed by using the **zfsadm config** command.

Figure 38-2 IOEPRMxx options

Figure 38-3 shows the syntax of the **zfsadm fileinfo** command.

```
zfsadm fileinfo -path <pathname> [{-globalonly | -localonly | -both}] [-level]
[-help]
```

Figure 38-3 zfsadm fileinfo command

It displays additional file or directory information, including whether a directory is version 4 or version 5. Sample output is shown in Figure 38-4.

```
$> zfsadm fileinfo .
   path: /u/hering/.
   ***   global data   ***
   fid                    1,1          anode                  1860,516
   length                 8192         format                 BLOCKED
   1K blocks              8            permissions            755
   uid,gid                888,2        access acl             0,0
   dir model acl          0,0          file model acl         0,0
   user audit             F,F,F        auditor audit          N,N,N
   set sticky,uid,gid     0,0,0        seclabel               none
   object type            DIR          object linkcount       52
   object genvalue        0x00000000   dir version            4
   dir name count         na           dir data version       796
   dir tree status        na           dir conversion         na
   file format bits       na,na,na     file charset id        na
   file cver              na           charspec major,minor   na
   direct blocks          0x00002908
   indirect blocks        none
   mtime       Jun 10 22:13:39 2013   atime        Jun 12 13:41:20 2013
   ctime       Jun 10 22:13:39 2013   create time  Sep  8 11:56:49 2006
   reftime     none
$>
```

Figure 38-4 Sample output for command zfsadm fileinfo

Several format programs have been enhanced.

- A IOEPRMxx option **format_aggrversion** has been added.
  - The default is version 4.
  - The setting is honored in ISHELL and with AUTOMOUNT processing.
- A change was made to the **zfsadm format** command to support version settings.
  - There are options **-version5** and **-version4**.
  - If no version has been specified, the value for **format_aggrversion** in IOEPRMxx is honored.
- A similar change has been added to batch utility **IOEAGFMT**.
  - There are options **-version5** and **-version4**.
  - If no version has been specified, the value for **format_aggrversion** in IOEPRMxx is honored.

Note: IOEAGFMT now requires zFS being active.

- The zFS format aggregate API supports a flag **af_aggrversion**.
  - The value can be 0, 4 or 5.
  - If the value is 0, the IOEPRMxx option **format_aggrversion** is used.
- There are options in **zfsadm config** and **zfsadm configquery** commands corresponding to the IOEPRMxx options:
  - **format_aggrversion**,
  - **change_aggrversion_on_mount** and
  - **converttov5**.
- Using zFS mount parms **CONVERTTOV5** and **NOCONVERTTOV5** override IOEPRMxx options **converttov5** and **change_aggrversion_on_mount**.
- Command **zfsadm aggrinfo -long** displays the aggregate version and the converttov5 attribute as shown in Figure 38-5.

```
$> sudo /usr/sbin/mount -f HERING.TEST.ZFS -oconverttov5 test
$> zfsadm aggrinfo hering.test.zfs -long | head -l4
HERING.TEST.ZFS (R/W COMP): 750 K free out of total 36000
version 1.5
auditfid C2C8F5E2 E3F20184 0000
sysplex-aware, converttov5
$>
```

Figure 38-5 Sample for zfsadm aggrinfo -long

Updated zFS APIs

Several zFS APIs provide additional information, can be used with functionality or are with z/OS 2.1 as shown in Table 38-1 on page 507.

Table 38-1 Updated zFS APIs

API	Update information
List File Information	Shows additional information about files or directories, including whether a directory is version 4 or 5
List Aggregate Status (Version 2)	Supports version 1.5 aggregates by using AGGR_STATUS3
Grow Aggregate	Supports version 1.5 aggregates by using AGGR_ID version 3
List File System Status	Supports version 1.5 aggregates by using FS_STATUS2

Notes:

- The List File Information API is with z/OS 2.1.
- With the exception of that List File Information API, which uses `__w_pioctl()`, all use `pfsctl()`.

Figure 38-6 shows an example using data with API List Aggregate Status (Version 2) by adding information regarding setting converttov5.

Note: The rxlsaggr utility is a zFS utility that is available v1.13 level as part of the zFS tools package on the following IBM Redbooks FTP page:

`ftp://www.redbooks.ibm.com/redbooks/SG247035/`

The version of rxlsaggr on the tools disk will get updated to reflect additional information according to the zFS updates in z/OS 2.1.

```
$> rxlsaggr hering.test.zfs
HERING.TEST.ZFS                                SC74      R/W Sysplex-aware
Monitored for full  . . : Disabled
block security  . . : Enabled
HFS compatibility . . . : Enabled
Auto-extend . . . . . . : Enabled
Number of fragments . . :      36000
8K Blocks available . . :       4500
Aggregate free space KB :        750
Number free 8K blocks . :         88
Number free 1K fragments:         46
Log file size in KB . . :        112
File system table in KB :         32
Bitmap file size in KB  :          8
Auditfid  . . . . . . . : C2C8F5E2 E3F20184 0000
Disk format version . . : 1.5
Converttov5 . . . . . . : Yes
$>
```

Figure 38-6 Showing aggregate information with setting converttov5

batch utility IOEFSUTL

There is a batch utility named IOEFSUTL that supports four commands for non-mounted file systems or aggregates.

IOEFSUTL format

Command **ioefsutl format** is a batch utility that formats a VSAM linear data set to become a zFS compatibility mode aggregate.

```
ioefsutl format -aggregate name [-size blocks] [-logsize blocks]
[-owner uid|name] [-group gid|name] [-perms number] [-grow blocks][-overwrite]
[{-version4 | -version5}] [-help] [-level]
```

Figure 38-7 ioefsutl format (acts as IOEAGFMT)

- Using explicit options **-version4** or **-version5** format the aggregate as version 4 or 5.
- If no explicit version is specified the IOEFSPRM option **format_aggrversion** is used.

Attention: IOEFSUTL requires zFS to be active if no version has been specified explicitly.

IOEFSUTL salvage

Command **ioefsutl salvage** is a batch utility that scans an aggregate and reports inconsistencies. This utility is known as the salvager:

- Using the salvage specification, it acts as IOEAGSLV.
- This supports version 1.4 and version 1.5 aggregates and version 4 and 5 directories.

```
ioefsutl salvage -aggregate name [-verifyonly] [-level] [-help]
```

Figure 38-8 Using ioefsutl for salvage processing

IOEFSUTL converttov5

Command **ioefsutl converttov5** is a batch utility that converts a version 1.4 aggregate to a version 1.5 aggregate:

- This is to convert aggregate version 1.4 to 1.5.
- It converts each v4 directory to v5, the directory format, provided option **-aggrversion_only** has not been specified.

```
ioefsutl converttov5 -aggregate name [-aggrversion_only] [-verbose] [-level]
[-help]
```

Figure 38-9 Converting a version 1.4 aggregate to version 1.5

IOEFSUTL converttov4

Command **ioefsutl converttov4** is a batch utility that converts a version 1.5 aggregate to a version 1.4 aggregate:

- This is the only way to convert aggregate version 1.5 to 1.4.
- It converts each v5 directory to v4, the old directory format.

```
ioefsutl converttov4 -aggregate name [-verbose] [-level] [-help]
```

Figure 38-10 Converting a version 1.5 aggregate to version 1.4

Restriction: It cannot convert backward in the following cases.

- A directory in the file system has a size greater than 4 G
- The file system size is greater than 4 TB
- A directory has more than 64k-1 subdirectories

Guidelines for v4 to v5 conversions

Here, we provide a few hints for conversion to the enhanced z/OS 2.1 formats:

- Version 1.4 aggregates and v4 directory format are still supported. You are not required to convert all directories and aggregates to v5.
- If an installation exports zFS to NFS or SMB, use offline conversion or the CONVERTTOV5 zFS mount parameter.
- If there are no plans to convert all file systems, the following suggestions might be helpful:
  - Determine the most active file systems by using MVS command **`modify zfs,query,filesets`** or the wjsfsmon tool.

    Note: The wjsfsmon tool is available from the z/OS UNIX Tools page:

    `http://www.ibm.com/systems/z/os/zos/features/unix/bpxa1ty2.html`

  - Identify file systems with large directories (more than 10,000 entries or a directory size greater than 800 KB).
  - Consider conversion performance. For example, on a z9 machine, you can expect a rate of about 3500 entries to be converted per second.

Sample processing for identifying large zFS directories

The following is a suggestion on how to large zFS directories. Note that it is not provided as a utility just as a list of steps that need to be done.

Step 1: Listing the locally mounted zFS

First, find the zFS aggregates owned locally by zFS. A sample is shown in Figure 38-11 on page 510.

```
$> zfsadm lsaggr -system $(sysvar SYSNAME) | tail +2 | awk '{print $1}'
PLEX75.SC74.SYSTEM.ZFS
HERING.ZFS
IZUSVR.HFS
BBN.V7RO.CONFIG1.ZFS
OMVS.SC74.VAR.WBEM
DAVISRD.HFS
HERING.LARGEDIR.V4
SYSPROG.ZFS
HERING.TEST.ZFS
OMVS.SC74.TDSSRV1.HFS
JES2.ZFS
OMVS.SIZUDATA
OMVS.SC74.WEB.PRINTWAY.ZFS
$>
```

Figure 38-11 Listing the locally mounted zFS

Note: It is recommended to process find commands which result in metadata searches on the zFS owning system as this is fully processed on the owning system.

Step 2: Finding the mount directories for the zFS

A sample is shown is shown in Figure 38-12 using either utility rxdowner or the mount command. You can also simply look into the ISHELL to find the information.

```
$> rxdowner -f HERING.LARGEDIR.V4 | grep "MP Directory :" | awk '{print $4}'
/u/hering/largedir.v4
$> /usr/sbin/mount -qv / 2>/dev/null | grep HERING.LARGEDIR.V4 |
> awk '{print $3}'
/u/hering/largedir.v4
$>
```

Figure 38-12 Finding the mount directory for a zFS

Note: The rxlsaggr utility is a zFS utility that is available at the v1.13 level as part of the zFS tools package on the following IBM Redbooks FTP page:

ftp://www.redbooks.ibm.com/redbooks/SG247035/

Step 3: Finding the large version 4 directories

Two sample commands are provided in Figure 38-13 on page 511 to retrieve this information.

```
$> find /u/hering/largedir.v4 -type d -size +1600 -xdev
/u/hering/largedir.v4
$> zfsadm fileinfo /u/hering/largedir.v4 | grep "dir version " |
> awk '{ print $5 "=" $6 }'
version=4
$>
```

Figure 38-13 Finding the large directories being of version 4

Notes:

- The **find** option `-size +1600` means to look for sizes above 1600 512-byte blocks.
- This **find** command can run long if a large v4 directory is hit and processing continues running down the structure.

Testing performance with large directories and conversion

Figure 38-14 provides a sequence of commands showing changed performance with large directories and with conversion from directories from version 4 to version 5.

```
$> echo We are running on system $(sysvar SYSNAME).
We are running on system SC74.
$> rxdowner -a hering.largedir.v4
zFS Owner    : SC74     - Aggregate read-only=N, sysplex-aware=Y
$> time find /u/hering/largedir.v4 -level 0 | wc -l

real.16m38.57s
user.10m32.75s
sys.3m30.91s
1000001

$> time zfsadm convert /u/hering/largedir.v4
IOEZ00791I Successfully converted directory /u/hering/largedir.v4 to version 5 format.

real.0m16.95s
user.0m 7.17s
sys.0m 2.39s

$> time find /u/hering/largedir.v4 -level 0 | wc -l

real.2m19.97s
user.1m11.21s
sys.0m23.73s
1000001

$>
```

Figure 38-14 Working with and converting a zFS v4 directory to v5

Figure 38-14 shows several facts, as listed hereafter.

Notes:

- zFS HERING.LARGEDIR.V4 is mounted locally and has exactly 1,000,000 entries in its root directory.
- We see a great change in performance after the conversion.
- The time needed for doing the conversion for 1,000,000 entries is acceptable.

Further conversion suggestions

In order to change only the aggregate version to version 1.5 and no directory conversion you can choose one of the following possibilities.

- You can change the version on specific file systems by using the following command:

  `zfsadm convert -aggrversion` *`aggrname`*

- You can change the version on all file systems that are mounted on a particular image by using the IOEPRMxx option `change_aggrversion_on_mount`.

You can decide to do conversion upon access (and also change aggregate version to v1.5). To do so, use one of the following methods:

- Use the `CONVERTTOV5` mount parameter on specific file systems.
- Set the `converttov5` mount attribute on all file systems that are mounted on a particular image by use of the IOEPRM*xx* option `converttov5`.

  Use the following command to convert a specific directory:

  `zfsadm convert -path` *`pathname`*

  Note: This also changes the aggregate version to V1.5.

  For a full conversion, changing the aggregate version to v1.5 and converting all directories to v5, use the IOEFSUTL **`converttov5`** offline utility.

38.1.2 Migration and coexistence

There are several considerations that need to be mentioned:

- The zFS toleration APAR OA39466 must be installed and active on all z/OS 1.12 and z/OS 1.13 systems before bringing zFS 2.1into a z/OS UNIX shared file system environment.
- Do not use Version 1.5 aggregates until all systems are running z/OS 2.1, and follow these practices.
  - Version 1.5 aggregates can be used only on z/OS 2.1 systems.
  - Use the **`IOEFSUTL converttov4`** command to bring Version 1.5 aggregates back to Version 1.4 for use on systems before z/OS 2.1. Keep in mind the restrictions that apply, as mentioned in “IOEFSUTL converttov4” on page 508.
- In z/OS 2.1, the batch utility IOEAGFMT requires zFS being active.

Changed default values for IOEPRM*xx* options

There are changed default values for IOEPRM*xx* the meta_cache_size and `metaback_cache_size` options in z/OS 2.1.

- If metaback_cache_size is specified and meta_cache_size is not, the default value of meta_cache_size is 64 M.

- If neither is specified, zFS calculates 10% of real storage (named X) during zFS initialization according to Table 38-2.

Table 38-2 Changed defaults for meta_cache_size and metaback_cache_size

If X....	Default value of meta_cache_size	Default value of metaback_cache_size
X < 64M	64M	No metadata back cache created
64M <= X < 100M	X	No metadata back cache created
100M <= X <= (2G +100M)	100M	X - 100M
X > (2G + 100M)	100M	2G

There is a changed default value for the IOEPRM*xx* `user_cache_size` option.

- In z/OS 2.1, zFS calculates 10% of real storage (named X) during zFS initialization. See Table 38-3.

Table 38-3 Changed default value for user_cache_size

If X....	Default user_cache_size
X < 256M	256M
256M <= X <= 2G	X
X > 2G	2G

Tip: Specify or change desired values in an IOEPRM*xx* member if the previous settings are no longer appropriate or the defaults are not acceptable.

38.1.3 Installation

- IOEAGFMT now requires zFS as being active.
- The corresponding PTF for APAR OA39466 needs to be installed on down-level systems that run earlier than z/OS 2.1. This is to tolerate the zFS externals on z/OS 2.1.
- There are changed default values for IOEPRM*xx* `meta_cache_size`, `metaback_cache_size` and `user_cache_size` options. See PARMLIB members IOEPRM*xx* and IOEPRM*yy* for details.

38.2 Improved zFS salvager

The previous zFS salvager utility had several problems and disadvantages:

- It did not always find and fix zFS structure problems.
- Sometimes, it needed to be run several times against the same zFS before it completely resolved the inconsistencies.
- It could not handle a large zFS or file systems with many objects because of the limitation of a 2 G address space.
- Sometimes, it abended when trying to fix a zFS.

- The serviceability characteristics of the salvager were insufficient for problem diagnosis.
- Some salvager messages did not have message numbers.

The improved salvager in z/OS 2.1 provides the following solutions for these shortcomings:

- The salvage utility can process a larger zFS by using storage above 2 GB.
- The repair processing completes without requiring to be run multiple times.
- All messages have message numbers.
- The verify option, **-verifyonly**, replays the log when necessary.

Note: This has several benefits.

- It improves processing, because it runs only once.
- It improves the scalability of the zFS batch salvager, because it can processes large file systems.
- RAS improves by improving the messages that are displayed.
- It results in improved performance.

38.2.1 Invocation and use

We recommend that you use the **IOEFSUTL** batch utility for calling the salvager, as the example in Figure 38-15 shows.

```
ioefsutl salvage -aggregate name [-verifyonly] [-level] [-help]
```

Figure 38-15 Format of the ioefsutl salvage command

Table 38-4 provides a description of the **IOEFSUTL** salvage options.

Table 38-4 Options of ioefsutl salvage

Option	Explanation
-aggregate name	Specifies the name of the aggregate to be verified or salvaged.
-help	Prints the online help for this command. All other valid options that are specified with this option are ignored.
-verifyonly	Specifies that the salvager is to verify the specified aggregate. It should not attempt to repair any damage found. The log is replayed before the verification unless an error occurs during the replay. If this option is omitted, the salvager replays the log, verifies the specified aggregate, and then attempts to repair any damage found.

Figure 38-16 shows sample JCL for performing a salvage verification.

```
//SALVAGE  EXEC   PGM=IOEFSUTL,REGION=0M,
//         PARM=('salvage -aggregate HERING.TEST.RW.ZFS -verifyonly')
//SYSPRINT DD   SYSOUT=*
//STDOUT   DD   SYSOUT=*
//STDERR   DD   SYSOUT=*
//SYSUDUMP DD   SYSOUT=*
//CEEDUMP  DD   SYSOUT=*
```

Figure 38-16 JCL for performing a salvage verify

Figure 38-17 shows the SYSPRINT output of the job shown in Figure 38-16 on page 514.

```
IOEZ00559I zFS IOEFSUTL: Initializing z/OS    zFS
Version 02.01.00 Service Level OA41870 - HZFS410.
Created on Mon May 13 13:38:21 EDT 2013.
Address space asid x3C
IOEZ00760I No IOEZPRM DD specified. Parmlib search being used.
IOEZ00786I Aggregate HERING.TEST.RW.ZFS was not unmounted cleanly.
Last update time stamp=Jun 13 20:35:29 2013 sysname=SC70
IOEZ00707I Log file size 13 8K blocks, verified correct
IOEZ00729I Verification of aggregate HERING.TEST.RW.ZFS started
IOEZ00705I Formatted v4 aggregate size 2250 8K blocks, dataset size 2250 8K blocks
IOEZ00707I Log file size 13 8K blocks, verified correct
IOEZ00709I Bitmap size 1 8K blocks, verified correct
IOEZ00782I Salvage has verified 1 of 12 pages in the anode table.
IOEZ00782I Salvage has verified 2 of 12 pages in the anode table.
IOEZ00782I Salvage has verified 3 of 12 pages in the anode table.
IOEZ00782I Salvage has verified 4 of 12 pages in the anode table.
IOEZ00782I Salvage has verified 5 of 12 pages in the anode table.
IOEZ00782I Salvage has verified 6 of 12 pages in the anode table.
IOEZ00782I Salvage has verified 7 of 12 pages in the anode table.
IOEZ00782I Salvage has verified 8 of 12 pages in the anode table.
IOEZ00782I Salvage has verified 9 of 12 pages in the anode table.
IOEZ00782I Salvage has verified 12 of 12 pages in the anode table.
IOEZ00782I Salvage has verified 6 of 6 directories in the directory tree.
IOEZ00782I Salvage has verified 2 of 2 pages in the partially-free page list.
IOEZ00782I Salvage has verified 1 of 1 pages in the totally free page stack.
IOEZ00722I Primary file system size 21 8K blocks, verified correct
IOEZ00739I Salvage processed 8 directory pages, 325 anodes, 18 indirect blocks and 12
anode table pages.
IOEZ00730I Verification of aggregate HERING.TEST.RW.ZFS completed, no errors found.
```

Figure 38-17 Output of the salvage verify job

Beneath other information, the output shown in Figure 38-17 indicates that the aggregate HERING.TEST.RW.ZFS was not unmounted cleanly.

Note: The reason is that the file system was copied physically (using ADRDSSU or REPRO) while the source file system was kept mounted in read-write mode.

38.2.2 Interactions and dependencies

There are no software and hardware dependencies.

Any users of zFS in z/OS 2.1 can use the functions and improvements.

38.2.3 Migration and coexistence considerations

We recommend that you use the **IOEFSUTL** batch facility to invoke the salvager. However, the **IOEAGSLV** batch utility is still available.

38.3 Enhancements for zFS quiesce and unquiesce processing

Before z/OS 2.1, zFS handled the quiesce processing of a zFS aggregate independent of z/OS UNIX. This caused a lack of information, from the z/OS UNIX point of view for two reasons:

- The **D OMVS,F** operator command and the z/OS UNIX **df** shell command show that the file system as ACTIVE rather than quiesced.
- This makes it harder for system administrators to determine why a job is not making progress when the zFS aggregate is quiesced.

Support for using the z/OS UNIX osictl() interface

In z/OS 2.1, zFS notifies z/OS UNIX before zFS quiesce processing by a call to the z/OS UNIX `osi_ctl()` quiesce API and again after unquiesce by using the z/OS UNIX `osi_ctl()` unquiesce API.

The zFS commands and zFS APIs used to quiesce and unquiesce a zFS are unchanged, but the way that quiesce works internally and the way that the quiesce status is displayed are modified.

When z/OS UNIX receives the `osi_ctl` quiesce call from zFS, z/OS UNIX quiesces and suspends operations in its layer to avoid unnecessary latch contention. This allows zFS aggregate quiesce status to be available by using z/OS UNIX commands.

Note: System administrators can determine when zFS aggregates are quiesced by using standard z/OS UNIX commands.

No external interfaces are changed with this support:

- The **D OMVS,F** and **df** commands display the quiesce status for zFS aggregates.
- The z/OS UNIX `w_getmntent()` API provides a Version 3 MNTE that contains the PFS exception status.
- The zFS **zfsadm aggrinfo –long** command is enhanced to show the quiescing job, system, and time for a zFS quiesced aggregate.
- The zFS List Aggregate Status (Version 2) API is enhanced with those same fields.

38.3.1 Invocation and use

Two different scenarios are reviewed and various commands are run. In the first case, the zFS is mounted Parallel Sysplex-unaware by using the zFS mount parameter **norwshare**. In the second situation, the zFS is mounted sysplex-aware. The quiesce information might be different for some of these commands in these cases.

Figure 38-18 on page 517 shows mounting a zFS sysplex-unaware and then quiescing it, which involves using the following utilities:

- The **rxlsaggr** utility was mentioned in “zFS extended or large directory support” on page 504.
- The **rxdowner** command is another utility that shows owner and status information about file systems and especially about zFS aggregates. It is available with **rxlsaggr**.
- The **cn** utility allows running MVS system commands directly from a UNIX shell.

```
$> sudo /usr/sbin/mount -f HERING.TEST.ZFS -o norwshare /u/hering/test
$> sudo zfsadm quiesce hering.test.zfs
IOEZ00163I Aggregate HERING.TEST.ZFS successfully quiesced
$> rxdowner -a hering.test.zfs
zFS Owner    : SC74     - Aggregate read-only=N, sysplex-aware=N
$> cn "d omvs,f,n=HERING.TEST.ZFS"
 BPXO045I 20.29.47 DISPLAY OMVS 330
 OMVS     0011 ACTIVE             OMVS=(1B)
 TYPENAME   DEVICE ----------STATUS----------- MODE  MOUNTED    LATCHES
 ZFS           203 QUIESCED                    RDWR  06/15/2013  L=67
   NAME=HERING.TEST.ZFS                              15.53.05    Q=67
   PATH=/u/hering/test
   MOUNT PARM=norwshare
   OWNER=SC74     AUTOMOVE=Y CLIENT=N
   QSYSTEM=SC74 QJOBNAME=HERING8 QPID=  16842923
$> df -v /u/hering/test | grep Quiesce
               (HERING.TEST.ZFS)         Not Available             Quiesced
Quiesce Owner     : SC74        Quiesce Jobname : HERING8       Quiesce PID : 16842923
$> zfsadm aggrinfo hering.test.zfs -long | grep Quiesced
Quiesced by job HERING8  on system SC74 on Sat Jun 15 20:28:06.304207 2013
$> rxlsaggr hering.test.zfs | grep Quiesced
HERING.TEST.ZFS                                 SC74      R/W Quiesced
Quiesced by job HERING8 on system SC74 on 2013-06-15 at 20:28:06
$>
```

Figure 38-18 Information about a quiesced Parallel Sysplex-unaware zFS

The various commands all show desired quiesce information, especially the UNIX commands. The **rxlsaggr** utility uses the enhanced zFS List Aggregate Status (Version 2) API.

The second situation, shown in Figure 38-19, handles the case of a zFS that is mounted Parallel Sysplex-aware.

```
$> sudo zfsadm unquiesce hering.test.zfs
IOEZ00166I Aggregate HERING.TEST.ZFS successfully unquiesced
$> sudo /usr/sbin/unmount -f HERING.TEST.ZFS
$> sudo /usr/sbin/mount -f HERING.TEST.ZFS /u/hering/test
$> sudo zfsadm quiesce hering.test.zfs
IOEZ00163I Aggregate HERING.TEST.ZFS successfully quiesced
$> rxdowner -a hering.test.zfs
zFS Owner    : SC74     - Aggregate read-only=N, sysplex-aware=Y
$> zfsadm aggrinfo hering.test.zfs -long | grep Quiesced
Quiesced by job HERING9  on system SC74 on Sat Jun 15 21:44:58.304207 2013
$> rxlsaggr hering.test.zfs | grep Quiesced
HERING.TEST.ZFS                                 SC74      R/W Quiesced
Quiesced by job HERING9 on system SC74 on 2013-06-15 at 21:44:58
$> rxlsaggr -qsd
RXLSA013I There are 1 quiesced aggregates.
HERING.TEST.ZFS
$>
```

Figure 38-19 zFS API -related information about a quiesced sysplex-aware zFS

The zFS API-related commands in Figure 38-19 show the same information as before.

Notes:

- Using the **-qsd** option with the **rxlsaggr** utility lists all quiesced zFSs.
- By using **f axr,rxlsaggr**, you can run **rxlsaggr** as a SYSREXX, for example:

```
f axr,rxlsaggr -qsd
```

This is different with the z/OS UNIX commands, as shown in Figure 38-20.

```
$> df -v /u/hering/test | grep Quiesce
$> cn "d omvs,f,n=HERING.TEST.ZFS"
 BPXO045I 22.04.50 DISPLAY OMVS 389
 OMVS     0011 ACTIVE             OMVS=(1B)
 TYPENAME   DEVICE ----------STATUS----------- MODE  MOUNTED    LATCHES
 ZFS           219 ACTIVE                      RDWR  06/15/2013  L=67
   NAME=HERING.TEST.ZFS                              21.44.47    Q=67
   PATH=/u/hering/test
   OWNER=SC74     AUTOMOVE=Y CLIENT=N
   PFS INFO:
   PFS EXCP: QUIESCED
$>
```

Figure 38-20 UNIX related information about a quiesced sysplex-aware zFS

Although **df** no longer sees the quiesce status, the **D OMVS,F** command still shows a base hint that the file system is quiesced, but it is shown with ACTIVE status as before z/OS 2.1.

38.3.2 Interactions and dependencies

There are no software and hardware dependencies, and any users of zFS in z/OS 2.1 can use the functions and improvements. The following conditions are relevant:

- Notice the slight difference in the output format, based on whether the zFS is mounted sysplex-aware or not.
- The **df** command does not show the quiesced status if the file system is mounted sysplex-aware.
- Read, write, and lookup operations are suspended.

38.4 zFS backup bit

Before z/OS 2.1, zFS did not handle the backup bit as desired for two reasons:

- zFS did set the change activity flag at mount processing, even when no writes occurred.

  Note: When the change activity flag is set, it means that a backup is needed.

- zFS did not set the change activity flag after a backup occurred (and the change activity flag has been reset), even when writes occurred.

In z/OS 2.1, this behavior has been corrected by the following changes:

- zFS does not set the change activity flag at mount processing.

- zFS sets the change activity flag in the following cases:
  - On the first write after mounting the zFS
  - On the first write after a successful reset change activity flag pfsctl() call
  - On log recovery
- There is a zFS Reset Change Activity `pfsctl()` API that can be used by backup programs to reset the backup bit after completion of a backup.

38.5 zFS auditfid default setting

Setting unique auditfids for zFS is controlled by the `auditfid` setting. In z/OS 2.1, the defaults are changed as follows to set this at creation time or convert it during mount processing:

- The default of the IOEPRM*xx* configuration option **`CONVERT_AUDITFID`** is changed from OFF to ON.
- The defaults for **`zfsadm format`** and **`IOEAGFMT`** have been set to use **`–newauditfid`**.
  - A option **`–nonewauditfid`** has been added to allow not to set unique auditfids.
  - The option **`-newauditfid`** existed since z/OS V1R9.
- The **`IOEFSUTL format`** command always formats with a unique auditfid.

The default for `auditfid` is now set to *on*.

38.5.1 Migration and coexistence

The zFS **`convert_auditfid`** option defaults are like IOEPRM*xx*: The **`convert_auditfid`** setting and the use of **`IOEAGFMT`** and the **`zfsadm`** format.

Note: The **`zfsadm configquery -convert_auditfid`** command displays the current `convert_auditfid` setting.

If you accept the auditfd, do one of the following actions.

- Ensure that IOEPRMxx option **`convert_auditfid`** is not specified or is set to ON for the next IPL.
- Use **`zfsadm config -convert_auditfid on`** for dynamic update.

Note: SMF 80 records will have unique auditfids. This might require updates in programs that process SMF 80 records.

If keeping the previous, not unique auditfid, choose one of these options:

- Specify the IOEPRMxx **`convert_auditfid=off`** option for the next IPL.
- Use the **`-nonewauditfid`** option in IOEAGFMT or **`zfsadm format`** for formatting aggregates.

Attention: Do not use the **`IOEFSUTL format`** command, because it defaults to **`-newauditfid`**.

38.6 Elimination of multi-file system aggregates and cloning

Beginning with z/OS 2.1, zFS no longer supports multi-file system aggregates and clones. As a result, the following zfsadm commands are no longer supported:

```
zfsadm clone
zfsadm clonesys
zfsadm create
zfsadm lsquota
zfsadm rename
zfsadm setquota
```

Note: Changes are added to the **zfsadm config**, **zfsadm configquery** and **pfsctl()** subcommands.

38.6.1 Invocation and use

Figure 38-21 shows a failing mount sample with an aggregate that contains a clone. This example demonstrates that it is not too late t deleting a clone after switching to z/OS 2.1.

```
$> sudo /usr/sbin/mount -f HERING.TEST.RW.ZFS test.rw
FOMF0504I mount error: 79 EF096B01
EINVAL: The parameter is incorrect
Description: Mount of .bak or aggregate with .bak not allowed.
$> bpxmtext EF096B01 | grep Action:
Action: Clones are no longer supported.
$> sudo zfsadm attach HERING.TEST.RW.ZFS
IOEZ00117I Aggregate HERING.TEST.RW.ZFS attached successfully
$> zfsadm lsfs HERING.TEST.RW.ZFS -fast
HERING.TEST.RW.ZFS.bak
HERING.TEST.RW.ZFS
$> sudo zfsadm delete HERING.TEST.RW.ZFS.bak
IOEZ00105I File system HERING.TEST.RW.ZFS.bak deleted successfully
$> sudo zfsadm detach HERING.TEST.RW.ZFS
IOEZ00122I Aggregate HERING.TEST.RW.ZFS detached successfully
$> sudo /usr/sbin/mount -f HERING.TEST.RW.ZFS test.rw
$>
```

Figure 38-21 Deleting a clone in z/OS 2.1 and then mounting successfully

38.6.2 Migration and coexistence

Note: If you are using multi-file system aggregates or clones, you must stop using them. Do not attempt to use any multi-file system or clone commands, APIs, or options.

Be sure that you complete the necessary migration actions before migrating to z/OS 2.1. For details, see *z/OS Migration*, GA32-0889 (see 38.9, “Documentation and reference” on page 523 for a link).

38.7 Renamed z/OS file system

In z/OS 2.1, the name of the z/OS file system (zFS) has been changed to *System z file system*. This is mainly a documentation change.

In the output of the **F ZFS,Q,LEVEL** command, the *zSeries File System* text has been replaced by *zFS*, as shown in Figure 38-22.

```
F ZFS,Q,LEVEL
IOEZ00639I zFS kernel: z/OS    zFS
Version 02.01.00 Service Level OA41870 - HZFS410.
Created on Mon May 13 13:38:21 EDT 2013.
sysplex(filesys,rwshare) interface(4)
```

Figure 38-22 Changed output for F ZFS,Q,LEVEL

38.8 Using non-SMS zFS aggregates with extended addressability

In previous versions, a data set was required to be SMS-managed and have extended format (EF) to be extended addressable (EA) and grow beyond 4 GB.

In z/OS 2.1, DFSMS removes the EA dependency on EF and allows a non-SMS-managed VSAM LDS to grow beyond 4 GB.

38.8.1 Invocation and use

The examples that follow define a non-SMS managed zFS aggregate and grow it to become larger than 4 GB. Figure 38-23 on page 522 shows defining the zFS and growing it close to 4 GB.

```
$> zfsadm define hering.nonsms.zfs -cylinders 500 500 -volumes BH5ST3 \
> BH5ST4 BH5ST5
IOEZ00248I VSAM linear dataset hering.nonsms.zfs successfully created.
$> sudo zfsadm format HERING.NONSMS.ZFS
IOEZ00077I HFS-compatibility aggregate HERING.NONSMS.ZFS has been successfully created
$> sudo zfsadm attach HERING.NONSMS.ZFS
IOEZ00117I Aggregate HERING.NONSMS.ZFS attached successfully
$> sudo /usr/local/bin/zfsgrow 12 hering.nonsms.zfs
Performing grow run 1...
IOEZ00173I Aggregate HERING.NONSMS.ZFS successfully grown
HERING.NONSMS.ZFS (R/W MULT): 716215 K free out of total 720000
...
Performing grow run 10...
IOEZ00173I Aggregate HERING.NONSMS.ZFS successfully grown
HERING.NONSMS.ZFS (R/W MULT): 3955767 K free out of total 3960000
Performing grow run 11...
IOEZ00326E Error 133 extending HERING.NONSMS.ZFS

The zfsadm grow was not successful; rc=1
No further grow run will be processed, stopping...
$>
```

Figure 38-23 Defining non-SMS zFS aggregate and grow it up close to 4GB

The error is forced because the size would become larger than 4 GB. Figure 38-24 shows how to alter the zFS to be extended addressable.

```
$> zfsadm aggrinfo hering.nonsms.zfs
HERING.NONSMS.ZFS (R/W MULT): 3955767 K free out of total 3960000
$> rexx 'say "Aggregate total size is" 3960000/(8*90) "cyls."'
Aggregate total size is 5500 cyls.
$> tsocmd "alter 'hering.nonsms.zfs' extaddr"
alter 'hering.nonsms.zfs' extaddr
DATA SET IN USE+
** VSAM CATALOG RETURN CODE IS 184 - REASON CODE IS IGGOCLK8-10
**ENTRY HERING.NONSMS.ZFS NOT ALTERED
LASTCC=8
$> sudo zfsadm detach hering.nonsms.zfs
IOEZ00122I Aggregate HERING.NONSMS.ZFS detached successfully
$> tsocmd "alter 'hering.nonsms.zfs' extaddr"
alter 'hering.nonsms.zfs' extaddr
ENTRY HERING.NONSMS.ZFS ALTERED
$> sudo zfsadm attach hering.nonsms.zfs
IOEZ00117I Aggregate HERING.NONSMS.ZFS attached successfully
$>
```

Figure 38-24 Alter the non-SMS zFS aggregate to be extended addressable

Important: The zFS aggregate must be closed during the ALTER operation.

Figure 38-25 on page 523 shows processing the missing grow runs and growing beyond 4 GB.

```
$> sudo /usr/local/bin/zfsgrow 2 hering.nonsms.zfs
Performing grow run 1...
IOEZ00173I Aggregate HERING.NONSMS.ZFS successfully grown
HERING.NONSMS.ZFS (R/W MULT): 4315719 K free out of total 4320000
Performing grow run 2...
IOEZ00173I Aggregate HERING.NONSMS.ZFS successfully grown
HERING.NONSMS.ZFS (R/W MULT): 4675671 K free out of total 4680000
$> rexx 'say "Aggregate total size is" 4680000/(8*90) "cyls."'
Aggregate total size is 6500 cyls.
$> rexx 'say "4GB are about" 4*1024*1024*1024%(8*1024*90) "cyls."'
4GB are about 5825 cyls.
$> sudo zfsadm detach hering.nonsms.zfs
IOEZ00122I Aggregate HERING.NONSMS.ZFS detached successfully
$>
```

Figure 38-25 Growing the non-SMS zFS aggregate to be larger than 4GB

Notes:

- The **zfsgrow** utility is available with the zFS IBM Redbooks publication. For convenience, it is provided with the "Additional material" of this book in text format (code page ISO8859-1). See Appendix A, "Material available to download" on page 535 for a link to that material.
- The **rexx** z/OS utility is for TSO and UNIX. It is available from the IBM Redbooks tools web page:

  ftp://www.redbooks.ibm.com/redbooks/SG247035/

38.8.2 Interactions and dependencies

There are no software and hardware dependencies.

Users are system programmers who want to migrate their non-SMS version root HFS data sets, which are that are larger than 4 GB, to zFS.

38.9 Documentation and reference

Also see Appendix A, "Material available to download" on page 535 in this book.

For more information, see the following IBM publications:

UNIX System Services z/OS Version 1 Release 7 Implementation, SG24-7035

ftp://www.redbooks.ibm.com/redbooks/SG247035/

- z/OS UNIX Tools web page:

  http://www.ibm.com/systems/z/os/zos/features/unix/bpxa1ty2.html

- *z/OS V2R1.0 DFSMS Access Method Services Commands*, SC23-6846

  https://ibm.biz/BdFEnJ

- z/OS V2R1.0 Information Center, z/OS Distributed File Service section
  https://ibm.biz/BdFEnu
  - *z/OS V2R1.0 Distributed File Service Messages and Codes*, SC23-6885
  - *z/OS V2R1.0 Distributed File Service zFS Administration*, SC23-6887
- z/OS V2R1.0 Migration (GA32-0889)
  http://www.ibm.com/support/docview.wss?uid=pub1ga32088900

39

SMB: Server Message Block

This chapter describes the Server Message Block (SMB) enhancements, functions, and features that were introduced with IBM z/OS Version 2 Release 1 (2.1). It includes these sections:

- 39.1, “Support for Microsoft Windows Server 2008” on page 526
- 39.2, “Documentation and reference” on page 527

© Copyright IBM Corp. 2014. All rights reserved.

39.1 Support for Microsoft Windows Server 2008

In z/OS 2.1, SMB support has been added to allow a Microsoft Windows Server 2008 to act as a domain controller for pass-through authentication.

Before z/OS 2.1, the z/OS SMB server supported only using Windows Server 2003 as a domain controller. Windows Server 2003 reached the end of Microsoft support in July 2010, with Windows Server 2008 as the follow-on release.

With z/OS 2.1, the z/OS SMB server can use a Microsoft Windows Server that acts as a domain controller to authenticate a PC client. The z/OS SMB server does this by passing the authentication packet to the Windows domain controller for authentication. This is called “pass-through authentication.”

Note: This support is being rolled back to z/OS 1.13.

APAR OA41211 has been opened to address this. (However, as of September 2013, there is only a PTF for z/OS 2.1.)

39.1.1 Invocation and use

Follow the Microsoft documentation to install and configure your Windows Server 2008 as a domain controller.

Users of the z/OS SMB server who use pass-through authentication to authenticate their SMB clients can use the function.

39.1.2 Interactions and dependencies

User must install and configure Microsoft Windows Server 2008.

39.1.3 Migration and coexistence

Digital signing is a local security configuration option on both Windows clients and Windows servers. Digital signing works differently on Windows server 2008 than on Windows Server 2003. The following configuration options are supported:

Windows clients:

- The *Microsoft network client - digitally sign communications (always)* policy is set to Disabled.
- The *Microsoft network client - digitally sign communications (if server agrees)* policy can be set to Enabled or Disabled.

Windows Server 2008:

- The *Microsoft network server - digitally sign communications (always)* policy is set to Disabled.
- The *Microsoft network server - digitally sign communications (if client agrees)* policy is set to Disabled.

39.2 Documentation and reference

For more information, see these IBM publications in the z/OS Distributed File Service section of the z/OS V2R1.0 Information Center:

https://ibm.biz/BdFEnu

- *z/OS V2R1Distributed File Service SMB Administration Guide*, SC23-6886
- *z/OS V2R1Distributed File Service Messages and Codes*, SC23-6885

40

Network File System

This chapter describes the Network File System (NFS) enhancements, functions, and features that were introduced with IBM z/OS Version 2 Release 1 (2.1). It includes these two sections and also describes invocation, use, interactions, and dependencies:

- 40.1, “NFS server reregistration with RPCBIND” on page 530
- 40.2, “Documentation and reference” on page 530

© Copyright IBM Corp. 2014. All rights reserved.

40.1 NFS server reregistration with RPCBIND

The RPCBIND service supplies client programs with the universal addresses of Remote Procedure Call (RPC) server programs. RPC server applications do not have well-known port numbers. Instead, they obtain an ephemeral port number, create a universal address, and register that address with RPCBIND. Clients contact server programs by obtaining the server's universal address from RPCBIND and sending messages to the server's universal address.

In previous releases, the problem was that a restart of the RPCBIND service required a restart of the NFS server to establish NFS V2 and NFS V3 mounts. In z/OS 2.1, the NFS server dynamically reregisters with the RPCBIND service when the latter is restarted.

Benefit: This enhances the availability of the NFS server by not requiring the NFS server to be restarted for establishing NFS V2 and NFS V3 mounts, when the RPCBIND service is recycled.

40.1.1 Invocation and use

There are no special requirements.

There are two console messages:

- GFSA1040I *procname* RPCBIND/PORTMAPPER registration is in progress.
- GFSA1041I *procname* RPCBIND/PORTMAPPER registration complete.

40.1.2 Interactions and dependencies

This function depends on the z/OS Communication Server RPCBIND recycle notification support.

System administrators no longer need to restart the NFS server to establish NFS V2 and NFS V3 mounts when the RPCBIND service is recycled.

40.2 Documentation and reference

For more information, see *z/OS 2.1.0 Network File System Guide and Reference*, SC23-6883:

http://ibm.co/1sHujbV

41

z/OS Font Collection

This chapter describes the z/OS Font Collection that was introduced with IBM z/OS Version 2 Release 1 (2.1). It includes these topics:

- 41.1, “Font library” on page 532
- 41.2, “Invocation and use” on page 532

© Copyright IBM Corp. 2014. All rights reserved.

41.1 Font library

In previous releases of z/OS, many fonts needed to be ordered separately. Starting with z/OS 2.1, IBM introduced a feature that contains a majority of those fonts. The fonts include SBCS and DBCS raster and outline fonts plus WorldType fonts, which are TrueType and OpenType fonts.

The z/OS Font Collection introduces a outline font library, SYS1.SFNTILIB, that replaces SYS1.FONTOLN. To use this library, you must replace references to SYS1.FONTOLN with SYS1.SFNTILIB.

The following applications might be affected:

- IBM Print Services Facility™ (PSF)
- IBM Print Transform from AFP to PCL, PDF, and PostScript for Infoprint Server for z/OS
- AFP Conversion and Indexing Facility (ACIF)

41.2 Invocation and use

The only action that you must take to use the feature is to integrate the libraries into your system. Table 41-1 shows the library names by font type.

Table 41-1 Font separation

Font type	Library name
AFP outline fonts	SYS1.SFNTILIB
240-pel AFP raster fonts	SYS1.FONTLIBB
240-pel Chinese, Japanese, Korean(CJK) raster fonts	SYS1.SFONDLIB
300-pel AFP raster fonts	SYS1.FONT300
WorldType fonts and symbolic links	/usr/lpp/fonts/worldtype

41.2.1 Installation

You do not need to install the z/OS Font Collection, because it is included and contains these two options:

- HFNT110 (Base)
- HFNT11J (Chinese, Japanese, Korean)

The z/OS Font Collection comes with a z/OS UNIX mount point, `usr/lpp/fonts`. The direct access storage device space that is needed for this feature is listed in Table 41-3 on page 533.

The collection contains five main font groups:

- The entire AFP Font Collection, Feature 5648-B33
- All the IBM InfoPrint Fonts, Feature 5684-E76
- WorldType Fonts, which were not a product that could be ordered in the past
- Some object features that are listed in Table 41-2 on page 533
- All of the fonts from the PSF Compatability Fonts Feature

The Font Collection contains the object features that are listed in Table 41-2. They are included in the base software, so there is no need to order these features separately.

Table 41-2 Object features of 5648-B33

Order number	Description
5771-ABC	Pi and Specials font object
5771-ADT	Math and Scient fonts objects
5771-ADA	Data1 fonts object
5771-ADB	IBM APL2® Bounded Box fonts object
5755-ADW	Sonoran Serif Headline object
5771-ABA	Sonoran Serif font object
5771-ABB	Sonoran Sans Serif font object
5771-ADX	Sonoran Sans Serif Headline object
5771-AFL	Sonoran Sans Serif Cond object
5771-AFN	Sonoran Sans Serif Expanded R1 object

Because the complete set of fonts is installed, it requires more z/OS UNIX file system space, as shown in Table 41-3.

Table 41-3 Space requirements for the Font Collection

File system type	Space required in 3390 cylinders
HFS	2028
zFS	2048

41.2.2 Migration and coexistence

The z/OS 2.1 Font Collection introduces a outline font library, SYS1.SFNTILIB, that replaces the SYS1.FONTOLN library that was in previous font products. To use this new library, replace references to SYS1.FONTOLN with SYS1.SFNTILIB.

A

Material available to download

Some chapters in this book refer to additional material that can be downloaded from the Internet, as described in the following sections.

Finding the web material

The web material associated with this book is available in softcopy on the Internet from the IBM Redbooks web server:

`ftp://www.redbooks.ibm.com/redbooks/SG248140`

Alternatively, you can go to the IBM Redbooks web page:

ibm.com/redbooks

Select **Additional materials** and open the directory that corresponds with the IBM Redbooks form number SG248140, which is this book.

The additional web material that accompanies this book includes the following file:

`SG248140.zip`

Download that file, which contains these four files:

File name	Description
`defa4all.zip`	Code samples
`zfsgrow.zip`	Code samples
`cn`	TSO REXX
`cn4uss`	UNIX script

Using the web material

Create a subdirectory (folder) on your workstation, and extract the contents of the material that is in the `SG248140.zip` file into your folder.

© Copyright IBM Corp. 2014. All rights reserved.

Related publications

Several of the chapters include topic-specific “Documentation and reference” sections. The additional publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

Note: The official IBM product manuals and publications take precedence over any item documented in this book.

IBM Redbooks

The following IBM Redbooks publications provide more information. Some publications cited in this list might be available only in softcopy, for download:

- *ABCs of z/OS System Programming: Volume 1* (a 13-volume series), SG24-6981

  http://www.redbooks.ibm.com/abstracts/sg246981.html?Open
- IBM z/OS Management Facility:
  - *IBM z/OS Management Facility V2R1*, SG24-7851

    http://www.redbooks.ibm.com/abstracts/sg247851.html?Open
  - *IBM z/OS Management Facility V2R1 Solution Guide*, TIPS1173

    http://www.redbooks.ibm.com/abstracts/tips1173.html?Open
- IBM z/OS 2.1 Communications Server TCI/IP Implementation:
  - *IBM z/OS V2R1 Communications Server TCP/IP Implementation Volume 1: Base Functions, Connectivity, and Routing*, SG24-8096

    http://www.redbooks.ibm.com/abstracts/sg248096.html?Open
  - *IBM z/OS V2R1 Communications Server TCP/IP Implementation Volume 2: Standard Applications*, SG24-8097

    http://www.redbooks.ibm.com/abstracts/sg248097.html?Open
  - *IBM z/OS V2R1 Communications Server TCP/IP Implementation Volume 3: High Availability, Scalability, and Performance*, SG24-8098

    http://www.redbooks.ibm.com/abstracts/sg248098.html?Open
  - *IBM z/OS V2R1 Communications Server TCP/IP Implementation Volume 4: Security and Policy-Based Networking*, SG24-8099

    http://www.redbooks.ibm.com/abstracts/sg248099.html?Open
- *Introduction to the New Mainframe: z/OS Basics, SG24-6366*

  http://www.redbooks.ibm.com/abstracts/sg246366.html?Open
- *z/OS V2.1 DFSMS Technical Update*, SG24-8190

  http://www.redbooks.ibm.com/redpieces/abstracts/sg248190.html?Open

You can search for, view, download, or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, on this website:

ibm.com/redbooks

© Copyright IBM Corp. 2014. All rights reserved.

Documentation

These websites are also important sources of information:

- IBM z/OS V2R1 Information Center

  https://ibm.biz/BdR7Qf
- IBM Knowledge Center:

  https://ibm.biz/BdFdez
- IBM z/OS basic skills education

  https://ibm.biz/BdRWRd

Online resources

The following resources are also helpful:

- z/OS V2R1 Elements and Features Library (PDF files)

  http://www.ibm.com/systems/z/os/zos/library/bkserv/v2r1pdf/
- z/OS Internet Library

  http://www.ibm.com/systems/z/os/zos/library/bkserv/index.html
- IBM Ported Tools for z/OS

  http://www.ibm.com/systems/z/os/zos/features/unix/port_tools.html
- Quick Publication Center, search topic or publication number for the most recent versions:

  https://ibm.biz/BdFdbZ
  - *z/OS V2R1 Collection, Mar 2014* (SK4T-4949-02)
  - *z/OS V2R1 Introduction and Release Guide*, GA32-0887
  - *z/OS V2R1 Migration*, GA32-0889
  - *z/OS V2R1 Planning for Installation*, GA32-0890
  - *z/OS V2R1 DFSMS Using the Function*, SC23-6857
- *z/OS Hot Topics* newsletter

  http://www.ibm.com/systems/z/os/zos/library/hot-topics/hot-topics.html
- IBM Resource Link (requires registration)

  https://www.ibm.com/servers/resourcelink/svc03100.nsf?Opendatabase
- IBM developerWorks articles related to z/OS

  https://ibm.biz/BdFdeu

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Redbooks

IBM z/OS Version 2 Release 1 Technical Updates

(1.0” spine)
0.875”<->1.498”
460 <-> 788 pages

IBM z/OS Version 2 Release 1 Technical Updates

Summary of changes

Examples of use

Migration considerations

This IBM Redbooks publication provides a broad understanding of the changes, features, and functions introduced with IBM z/OS Version 2 Release 1 (2.1).

This version marks a era of z/OS. Version 2 lays the groundwork for the next tier of mainframe computing, enabling you to pursue the innovation to drive highly scalable workloads, including private clouds, support for mobile and social applications, and more. Its unrivaled security infrastructure helps secure vast amounts of data. Its highly optimized availability can help you deliver data analytics solutions. And its continued improvements in management help automate the operations of IBM zEnterprise systems.

With support for IBM zEnterprise EC12 (zEC12, Enterprise Class) and zEnterprise BC12 (zBC12, Business Class) systems, z/OS 2.1 offers unmatched availability, scalability, and security to meet the business challenges of cloud services and data analytics and the security demands of mobile and social network applications. Through its unique design and qualities of service, z/OS provides the foundation that you need to support these demanding workloads alongside your traditional mission-critical applications.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-8140-00 ISBN 0738439576